

Web Engineering for Workflow-based Applications: Models, Systems and Methodologies

zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften

von der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

Dissertation

von

Patrick Freudenstein

aus Ludwigsburg

Tag der mündlichen Prüfung: 14.07.2009

Erster Gutachter: Prof. Dr. Wilfried Juling

Zweiter Gutachter: Prof. Dr. Hartmut Schreck

Acknowledgements

Working towards this thesis, I received manifold support for which I am very grateful and would like to express my appreciation for.

First of all, I would like to thank my supervisor Prof. Dr. Wilfried Juling for giving me the great opportunity to join his research group at the Karlsruhe Institute of Technology's (KIT) Institute of Telematics as well as for leaving me valuable freedom and scope for my research. Furthermore, I would like to thank Prof. Dr. Hartmut Schmeck for his work and assistance as second referee as well as Prof. Dr. Andreas Oberweis and Prof. Dr. Sebastian Abeck for acting as examiners in my disputation.

During my time at the KIT, I had the luck to work with excellent and highly motivated colleagues and students in a friendly and inspiring atmosphere, for which I am particularly grateful. I would like to thank my colleagues Jan Buck, Prof. Dr. Martin Gaedke, Frederic Majer, Dr. Johannes Meinecke and Dr. Martin Nußbaumer from the IT Management and Web Engineering Research Group (MWRG) for the numerous stimulating discussions and fruitful joint work. I would also like to thank Ina Dvorak for assisting me with administrative work and for her cheerful way of resolving problems. I am thankful to Martin N. for managing the group's transition so smoothly as well as for his continuous helpfulness and support, proficient advice and our successful collaboration throughout the years. I am particularly thankful to Frederic who stood by my side throughout all the hard work and made it much more enjoyable. Being my best friend, he has also been a great mentor, always available for discussing questions and a unique source of creativity.

Furthermore, I am thankful to my graduate students Florian Allarding, Christoph Augenstein, Marko Böttger, Jan Buck, Thorsten Höllrigl, Kristina Jochim, Leila Karademir, Nikolay Orozov, Frank Schell, Denny Setiawan and Frédéric Wenzel for their high enthusiasm, commitment and valuable contributions to my research.

The project 'Karlsruhe's Integrated InformationManagement (KIM)' and the people involved therein presented a central pillar of my work at the KIT. As a dynamic and motivated team, we have succeeded numerous ambitious accomplishments, often beyond the team member's actual responsibilities. As representatives for the entire team, I would like to thank Jan Buck, Thorsten Höllrigl, Stefan Link, Lei Liu, Frederic Majer, Axel Maurer, Christof Momm, Daniel Ried, Frank Schell, Wilhelm Sievers and

Marek Šiller for the great time, the joint efforts and the friendly atmosphere in the KIM Labs.

Moreover, I am thankful to all my friends and former colleagues who have supported me in my personal development and encouraged me in my decision to take the path towards doing a PhD. In this regard, I would like to particularly thank Dr. Dietger Bansberg, Dr. Stefan Feyrer and Sabine Schnarrenberger for their great sponsorship, mentoring and deep friendship.

Finally, I would like to express my particular gratitude to my parents Jutta and Gerd for conveying me the values and principles which made my path of life and this thesis possible in the first place. They, together with my brother Moritz and my grandparents, have always and unconditionally supported and encouraged me and enriched my life with kindness and happiness. I have special thanks for my grandfather Werner who sparked my interest in programming already in my early childhood and thus laid the foundation for my development as a computer scientist. I learnt so much from him and he will always be a great role model for me.

Above all, I would like to thank my fiancée Dana for her unique and unlimited understanding during these intense years. She has always supported and encouraged me and infected me with her happiness. Thank you so much, Dana!

Zusammenfassung

Workflowbasierte Web Anwendungen stellen eine eigenständige, stark an Bedeutung gewinnende Generation von Anwendungen dar, die auf den Technologien und Standards des World Wide Web Consortiums (W3C) aufbauen sowie Dienste und Inhalte über den Browser anbieten. Zu deren Evolution haben primär drei Faktoren beigetragen: Erstens, die weitreichende, durch erfolgreiche Standardisierungsbemühungen getriebene Akzeptanz und Etablierung des Konzepts der serviceorientierten Architektur (SOA). Zweitens, die unter dem Schlagwort „Web 2.0“ subsummierte neue Generation hochgradig interaktiver Web Anwendungen und Basistechnologien. Drittens, der aus Sicht von Unternehmen dringende Bedarf einer durchgängigen und flexiblen IT-basierten Geschäftsprozessunterstützung über System- und Organisationsgrenzen hinweg. Vor diesem Hintergrund bieten workflowbasierte Web Anwendungen eine homogene, integrative und plattformunabhängig verfügbare Benutzungsschnittstelle zur effektiven Abwicklung von Geschäftsprozessen.

Aus technischer Sicht ergeben sich hierdurch vielfältige Anforderungen, die zur Erzielung einer effizienten und effektiven Entwicklung workflowbasierter Web Anwendungen zu adressieren sind. Dies betrifft insbesondere die Bereiche der Prozesssteuerung und der Benutzungsschnittstelle sowie weitere, aus den spezifischen Charakteristika von Web Anwendungen resultierende Herausforderungen. Hierzu zählen beispielsweise kurze Entwicklungs- und Evolutionszyklen, die zunehmende Bedeutung von Ästhetik und Benutzungsfreundlichkeit, die Vielfalt an Zugriffskanälen sowie die ausgeprägte Heterogenität der zukünftigen Nutzer und der an der Entwicklung beteiligten Personen (engl. „Stakeholder“). Des Weiteren stellen die Vielzahl unterschiedlicher Entwicklungs-Artefakte sowie die inhärente Dokumentenzentriertheit des Web grundlegende Spezifika von Web Anwendungen dar.

Neben den genannten Problemstellungen kommt aus einer kommunikations- und kollaborationsorientierten Sichtweise der intensiven und effektiven Einbindung von Stakeholdern eine zentrale Bedeutung zu. In verschiedenen Studien wurde die mangelhafte Einbindung und Kommunikation mit allen relevanten Personengruppen als einer der Hauptgründe für das Scheitern von Softwareprojekten identifiziert. Dies gilt insbesondere im Kontext workflowbasierter Web Anwendungen, für die eine effektive Kommunikation bezüglich der abzubildenden Geschäftsprozesse und Aktivitäten

wesentlich ist. Darüber hinaus kann der dabei relevante Personenkreis entsprechend der Komplexität zu betrachtender Geschäftsprozesse hochgradig heterogen sein.

Vor diesem Hintergrund präsentiert die vorliegende Dissertation einen modellgetriebenen Ansatz zur Konstruktion workflowbasierter Web Anwendungen mit besonderem Fokus auf die effiziente und effektive Einbindung von Stakeholdern. Aufbauend auf einer systematischen Analyse der Problemdomäne wurden neuartige Konzepte, Modelle, Systeme und Methodiken entwickelt und evaluiert, die einen wesentlichen Beitrag zum aktuellen Stand der Technik darstellen. Im Einzelnen liefert die Arbeit folgende Beiträge:

Web Engineering DSL Rahmenwerk: Als konzeptionelle Grundlage für eine intensivere und effektivere Zusammenarbeit mit Stakeholdern bei der Entwicklung von Web Anwendungen wird ein neuartiger modellgetriebener Konstruktionsansatz präsentiert, der auf domänenspezifischen Sprachen (DSLs) fußt. Ziel ist es, Stakeholder in die Lage zu versetzen, eigenständig Teile der Lösung validieren, modifizieren und auch spezifizieren zu können. Das Rahmenwerk sieht die Verwendung verschiedener, stark fokussierter DSLs für die einzelnen Aspekte einer Web Anwendung vor. Dabei erlaubt die DSL-Spezifikation die Integration auf bestimmte Stakeholdergruppen und Prozessphasen zugeschnittener Notationen und Werkzeuge. Dadurch kann deren Erlernbarkeit und Anwendbarkeit sowohl für Entwickler als auch für Stakeholder mit geringen IT-Kenntnissen signifikant verbessert werden. Der vorgestellte Ansatz bietet eine neuartige Alternative zu den existierenden schwergewichtigen und monolithischen Modellierungsansätzen im Web Engineering, die primär für die Verwendung innerhalb von Entwicklerteams entworfen wurden. Web Anwendungen werden somit evolutionär durch Komposition spezifischer Softwarekomponenten und deren Konfiguration durch DSL-Programme in Form von Stakeholder-orientierten Modellen konstruiert und weiterentwickelt.

Der „Workflow-DSL“-Ansatz: Eine Anwendung des DSL-Konzepts zur Stakeholder-orientierten und vollständig modellgetriebenen Konstruktion workflowbasierter Web Anwendungen stellt die Workflow DSL dar. Durch den Ansatz wird eine holistische Spezifikation von Workflow-Aspekten und webbasierten Benutzungsschnittstellen für die effiziente und effektive Abwicklung von Workflow-Aufgaben möglich. Das der DSL zugrunde liegende formalisierte Schema stellt eine neuartige, auf weitverbreiteten Standards und deren nativen Erweiterungsmechanismen basierende, konzeptionelle Grundlage zur integrierten Spezifikation dieser Aspekte dar. Die Workflow DSL erlaubt die iterative Modellierung workflowbasierter Web Anwendungen mittels verschiedenster bekannter Notationen und Werkzeuge in Abhängigkeit des Entwicklungsfortschritts und der Bedürfnisse der jeweiligen Stakeholdergruppe. Für die einfache Modellierung einzelner Workflow-Aktivitäten wird ein Katalog webspezifischer Aktivitätsbausteine wie zum Beispiel dialogbasierte Interaktion, Datenpräsentation oder Web Service Kommunikation eingeführt. Für jeden dieser hochgradig wiederverwendbaren Bausteine wird die minimal benötigte, typspezifische Konfigurationsmenge allgemeingültig und implementierungsunabhängig definiert.

Zur Erzielung eines neuartigen Grads an Modellkontinuität und -konsistenz wird ein neues Konzept zur Überwindung der Heterogenität existierender Geschäftsprozess-

modellierungssprachen vorgestellt. Dieses bisher ungelöste Problem wird durch eine wohldefinierte Menge allgemeiner Geschäftsprozess- und Workflow-Konzepte adressiert, die von individuellen Notationen und Sprachen abstrahiert und die Grundlage zur Erzielung semantischer Kongruenz darstellt. Obwohl dadurch nicht alle theoretisch möglichen Konstrukte einbezogen werden, bestätigen umfangreiche empirische Evaluationen eine ausreichende Abdeckung von in der Praxis auftretenden Geschäftsprozessmodellen. Darauf aufbauend wird ein erweiterbares Rahmenwerk für verlustfreie, bilaterale Modelltransformationen präsentiert, wodurch schließlich ein notationsübergreifender Modellierungsansatz erzielt wird.

Die technische Unterstützungsplattform verwirklicht die vollständig automatisierte Konstruktion workflowbasierter Web Anwendungen auf Basis eines gegebenen Workflow DSL-Modells. Das dabei realisierte Architekturkonzept ist dienstorientiert ausgelegt, wodurch die Grundlage zur Realisierung föderativer Szenarien sowie multimodaler Partizipation geschaffen wird. Der automatisierte Konstruktionsprozess, die konsistente Propagierung von Änderungen sowie die Möglichkeit eines detaillierten, DSL-basierten Entwurfs der webbasierten Benutzungsschnittstelle zur Laufzeit fördern eine agile und evolutionsorientierte Entwicklungsmethodik.

Der „Dialog-DSL“-Ansatz: Effektive dialogbasierte Benutzerinteraktion stellt den zentralen Baustein des oben genannten Katalogs zur webbasierten Unterstützung von Geschäftsprozessaktivitäten dar. Die Dialog DSL stellt neuartige Modelle, Werkzeuge und ein Vorgehensmodell zur vollständig modellgestützten Entwicklung komplexer und hochgradig dynamischer Dialogkomponenten bereit. Dabei stehen Aspekte der Benutzungsfreundlichkeit sowie die effektive Einbindung von Stakeholdern im Vordergrund. Während existierende Lösungen immer noch eine traditionelle, überwiegend präsentations- und technikgetriebene Entwicklungsmethodik verfolgen, lenkt die Dialog DSL bereits zur Entwurfszeit den Fokus auf die Berücksichtigung und Integration von Aspekten der Benutzungsfreundlichkeit und insbesondere dynamischem Verhalten. Dadurch werden die Potenziale webbasierter Dialoge effektiver genutzt und eine kognitive Überlastung zukünftiger Nutzer vermieden. Ein weiterer wesentlicher Fortschritt zum gegenwärtigen Stand der Technik stellt die hervorragende Anwendbarkeit des Ansatzes für Stakeholder mit wenig oder keinen IT-Kenntnissen dar, die durch formale empirische Experimente nachgewiesen wurde. In diesem Kontext wurden auch signifikante Effizienzsteigerungen gegenüber existierenden Ansätzen nachgewiesen. Die resultierenden Dialogmodelle sind plattformunabhängig und werden zur Laufzeit dynamisch entsprechend den Charakteristika der anfragenden Clients transformiert. Zur Unterstützung einer agilen und evolutionsorientierten Entwicklungsmethodik unterstützt der Ansatz die automatische Generierung webbasierter Dialoge, beispielsweise entsprechend des W3C XForms-Standards, sowie die durchgängig modellbasierte Spezifikation mit Hilfe eines webbasierten Editors.

Der „Web Engineering Reuse Sphere“-Ansatz: Besonders modell- und komponentenbasierte Ansätze, wie die in dieser Arbeit vorgestellten, können von einer effektiven Wiederverwendungsunterstützung profitieren. Um Wiederverwendungspotenziale nicht wie bisher nur auf eine bestimmte Art von Artefakten oder Entwicklungsmethodik zu beschränken, stellt die Web Engineering Reuse Sphere einen holistischen, artefakt- und methodenübergreifenden Ansatz für die stark heterogene

Web Engineering-Domäne dar. Ein an Prinzipien aus dem Bereich Enterprise Application Integration (EAI) angelehnte Referenzarchitektur unterstützt neben der geplanten auch die spontane Wiederverwendung, wodurch Artefakte während ihres gesamten Lebenszyklus zur Wiederverwendung nutzbar werden. Zur semantischen Verknüpfung und Homogenisierung der verschiedenen Artefakt-Typen und Web Engineering-Methodiken wurde auf Basis von Semantic Web-Standards eine Ontologie für die Problemdomäne entwickelt. Dabei wird die Berücksichtigung von Stakeholder-Fähigkeiten als Schlüsselfaktor für die Effektivität der Wiederverwendung aufgefasst, d.h. die Befähigung wiederverwendbare Artefakte zu verstehen, zu evaluieren und ggf. anzupassen und zu verwenden. Demzufolge werden auf Basis der Ontologie neuartige wissensbasierte Suchstrategien realisiert, die auch Stakeholder ohne Expertenwissen in die Lage versetzen, effizient geeignete Methodiken und Artefakte für gegebene Probleme und in Übereinstimmung mit ihren individuellen Fähigkeiten zu finden. In Anbetracht aktueller Konsolidierungs- und Interoperabilitätsinitiativen im Web Engineering stellt der Ansatz einen wertvollen Beitrag zur tatsächlichen Verwirklichung methodenübergreifender Wiederverwendung dar. Darüber hinaus bildet er auch eine ideale Ergänzung des Web Engineering DSL Rahmenwerks, indem er Stakeholder beim Finden individuell geeigneter DSLs, Modellierungsnotationen, Werkzeuge und assoziierter Artefakte unterstützt.

Die in dieser Dissertation vorgestellten neuartigen Konzepte, Modelle, Systeme und Methodiken stellen einen wesentlichen Fortschritt zur effizienten und effektiven Entwicklung workflowbasierter Web Anwendungen mit Stakeholdern dar. Sie adressieren erfolgreich die identifizierten Anforderungen und bislang ungelösten Probleme. Die vorgestellten Ansätze wurden erfolgreich implementiert und in verschiedenen Szenarien evaluiert. Dies geschah sowohl durch formale empirische Studien und Experimente als auch durch den Einsatz in realen Projekten. In beiden Fällen konnten deutliche Verbesserungen hinsichtlich der Effizienz und Effektivität der Entwicklung sowie der intensiven Einbindung von Stakeholdern festgestellt werden. Darüber hinaus konnten die Ergebnisse durch zahlreiche Publikationen auf internationalen Konferenzen, Workshops und Zeitschriften mit Forschern aus der Web Engineering Disziplin und angrenzenden Gebieten diskutiert werden.

Contents

Acknowledgements	v
Zusammenfassung	vii
Contents	xi
1 Introduction	1
1.1 RESEARCH QUESTIONS AND CONTRIBUTIONS	2
1.2 RESEARCH CONTEXT AND SCOPE	7
1.3 THESIS STRUCTURE	8
2 Problem Scope	11
2.1 STAKEHOLDER COLLABORATION IN THE WEB ENGINEERING FIELD	11
2.2 WORKFLOW-BASED WEB APPLICATIONS.....	12
2.2.1 <i>Technical Challenges</i>	14
2.2.2 <i>Stakeholder Collaboration for Workflow-based Web Applications</i>	17
2.2.3 <i>Requirements Catalog for the Dimension Workflow</i>	21
2.3 WEB-BASED DIALOGS AS PRIMARY INTERACTION MEDIUMS	22
2.3.1 <i>Requirements Catalog for the Dimension Dialog</i>	25
2.4 EFFECTIVE REUSE	26
2.4.1 <i>Requirements Catalog for the Dimension Reuse</i>	28
3 State of the Art	29
3.1 DIMENSION WORKFLOW	29
3.1.1 <i>Object-Oriented Hypermedia Design Method (OOHDM)</i>	29
3.1.2 <i>Web Modeling Language (WebML)</i>	31
3.1.3 <i>UML-based Web Engineering (UWE)</i>	32
3.1.4 <i>IBM WebSphere Suite</i>	34
3.2 DIMENSION DIALOG	37
3.2.1 <i>Object-Oriented Hypermedia Design Method (OOHDM)</i>	37

3.2.2	<i>Web Modeling Language (WebML)</i>	38
3.2.3	<i>UML-based Web Engineering (UWE)</i>	39
3.2.4	<i>IBM Lotus Forms Designer</i>	40
3.3	DIMENSION REUSE	42
3.3.1	<i>Scientific Reuse Approaches for the Web Engineering Domain</i>	42
3.3.2	<i>Commercial Solutions</i>	44
3.4	EVALUATION RESULTS AND CONCLUSION	45
4	Web Engineering for Workflow-based Applications – A DSL Approach	51
4.1	THE WEB ENGINEERING DSL FRAMEWORK	51
4.1.1	<i>DSLs – Evolutionary Web Development for and with Reuse</i>	52
4.1.2	<i>Technical Platform</i>	56
4.2	OVERVIEW OF SOLUTION ELEMENTS	57
5	Constructing Workflow-based Web Applications with Stakeholders	61
5.1	THE WORKFLOW DSL AT A GLANCE	62
5.1.1	<i>Elements of the Workflow DSL</i>	62
5.1.2	<i>Evolutionary Process Model</i>	63
5.2	THE DSM – PROCESS INTERMEDIATE LANGUAGE	65
5.2.1	<i>The XML Process Definition Language as Foundation for the DSM</i>	66
5.2.2	<i>Catalog of Activity Building Blocks (ABBs)</i>	68
5.2.3	<i>Extending XPDL towards Web-specific Concerns</i>	72
5.3	THE DIMS – MULTI-NOTATIONAL MODELING WITH STAKEHOLDERS	76
5.3.1	<i>Simple Sequence Only (SSO) with Microsoft Word</i>	78
5.3.2	<i>Business Process Modeling Notation (BPMN) with Microsoft Visio</i>	79
5.3.3	<i>UML Activity Diagrams with IBM Rational Software Architect</i>	82
5.3.4	<i>Petri Nets with INCOME2010</i>	86
5.4	MODEL TRANSFORMATION FRAMEWORK	91
5.4.1	<i>Strategy for Efficient and Effective Model Transformations</i>	91
5.4.2	<i>The Core Elements Set (CES) Concept</i>	95
5.4.3	<i>Horizontal Model Transformations – The Petri net DIM</i>	99
5.4.4	<i>Vertical Model Transformations – The XOML Workflow Language</i>	106
5.4.5	<i>Complete Catalog of DIM Mappings</i>	113
5.5	TECHNICAL PLATFORM	115
5.5.1	<i>Technical Platform for the Model Transformation Framework</i>	116
5.5.2	<i>Workflow Execution Platform</i>	120
5.5.3	<i>Automated Application Construction: From Modeling to Execution</i>	126
5.5.4	<i>Support for Federative Scenarios</i>	129

5.6	SUMMARY	131
6	Constructing Advanced Web-based Dialogs.....	135
6.1	THE DIALOG DSL AT A GLANCE	135
6.1.1	<i>Elements of the Dialog DSL.....</i>	<i>136</i>
6.1.2	<i>Evolutionary Process Model.....</i>	<i>137</i>
6.2	THE DOMAIN-SPECIFIC MODEL (DSM).....	138
6.3	THE DOMAIN INTERACTION MODEL (DIM).....	140
6.3.1	<i>Partitions & Transitions Modeling Tier.....</i>	<i>141</i>
6.3.2	<i>Appearance Modeling Tier.....</i>	<i>142</i>
6.4	MODEL TRANSFORMATIONS.....	143
6.4.1	<i>User-Agent-related Model Adaptations</i>	<i>143</i>
6.4.2	<i>Model-to-Code Transformations</i>	<i>144</i>
6.5	TECHNICAL PLATFORM	145
6.5.1	<i>The Web-based DIM Editor.....</i>	<i>146</i>
6.5.2	<i>The Solution Building Block (SBB)</i>	<i>149</i>
6.6	SUMMARY	151
7	The Web Engineering Reuse Sphere	153
7.1	THE SPHERE CONCEPT.....	154
7.2	THE SEMANTIC CORE: THE WEB ENGINEERING REUSE ONTOLOGY.....	155
7.2.1	<i>Overview of the Web Engineering Reuse Ontology</i>	<i>156</i>
7.2.2	<i>The Concepts Knowledge and Stakeholders</i>	<i>158</i>
7.2.3	<i>The Concepts Artifact, Methodology, Process and Product.....</i>	<i>159</i>
7.2.4	<i>The Concepts Resolution Strategy, Modeling Technique & Software.....</i>	<i>160</i>
7.3	EFFECTIVE SEARCH AND INTEGRATION	162
7.4	STORING ARTIFACTS WITH RICH METADATA	163
7.5	REFERENCE ARCHITECTURE FRAMEWORK.....	165
7.6	CROSS-METHODOLOGICAL REUSE WITH STAKEHOLDERS IN PRACTICE	168
7.6.1	<i>Finding Stakeholder-Tailored Resolution Strategies and Artifacts</i>	<i>168</i>
7.6.2	<i>Stakeholder-Oriented Facetted Search and Browsing Facilities</i>	<i>171</i>
7.7	SUMMARY	174
8	Evaluation	177
8.1	EMPIRICAL EVALUATION OF WORKFLOW DSL CONCEPTS.....	178
8.1.1	<i>Expressiveness of the CES in Real-World Process Models.....</i>	<i>178</i>
8.1.2	<i>Coverage of Real-World Process Activities by the ABB Catalog.....</i>	<i>182</i>
8.2	WORKFLOW DSL CONCEPTS APPLIED IN THE KIM PROJECT	183

8.2.1	<i>FSM-based Modeling of User Interaction Workflows using ABBS</i>	<i>185</i>
8.2.2	<i>Technical Framework for Executing UI Workflows in Web Portals.....</i>	<i>186</i>
8.2.3	<i>Experiences</i>	<i>188</i>
8.3	FORMAL EMPIRICAL EVALUATION OF THE DIALOG DSL.....	189
8.3.1	<i>Experimental Evaluation of Development and Change Efficiency</i>	<i>189</i>
8.3.2	<i>Survey-based Evaluation of Stakeholder Adequacy.....</i>	<i>196</i>
9	Conclusion and Outlook	203
	List of Figures.....	209
	List of Tables.....	213
	Bibliography	215

1 Introduction

Since its inception in 1991, the World Wide Web has evolved rapidly from a decentralized information medium to a platform for complex and distributed applications. Likewise, the requirements and expectations for Web-based applications have increased substantially. The first genre of Web applications initiated the Web's transition from static hypertext documents (the "Static Web") to dynamically generated Web pages based on content stored in databases (the "Dynamic Web") (Rossi, Pastor, Schwabe et al. 2008). This allowed for simple customization-related features and limited user interactivity, whereby the Web application itself was the only access channel to the information available therein.

The continuous adoption of the Service-Oriented Architecture (SOA) paradigm and Web 2.0 technologies in the last few years resulted in a new class of Web-based applications (O'Reilly 2005; Phifer 2006). Such applications are highly interactive and user-centered, provide a rich user interface and are constructed based on SOA concepts and standards. At the same time, companies started to take advantage of maturing SOA-related standards and technologies in order to realize integrated end-to-end business processes spanning a great diversity of heterogeneous systems, organizations, and job roles. Today's companies vision of increased business agility results in Business Process Management Suites being one of the fastest growing software infrastructure markets with an estimated annual growth rate of 25% (Cantara, Biscotti and Raina 2007). In this regard, key challenges lie not only in the correct design and execution of business processes or the integration of heterogeneous systems and identities, but also in providing an adequate user interface (Hill, Sinur, Flint et al. 2006). The efficiency and effectiveness of human tasks and thus of the overall business process is strongly influenced by the quality of such a user interface (Nielsen 2005; Wroblewski 2008).

To this end, Web-based Enterprise Portals serving as uniform and integrated interfaces to content, business applications and processes have proved to be an ideal foundation of a "high-performance workplace" (Gootzit, Phifer, Valdes et al. 2008). Depending on the scenario, they serve diverse target audiences including employees, customers and business partners. By providing a personalized overview of available and running business processes as well as high-quality user interfaces for completing associated tasks, they promise to empower users to directly contribute to business processes in an efficient and effective way. Several studies underline the current and

future importance of Enterprise Portals to companies as they rank among the top five spending priorities of CIOs (Merrill Lynch 2006; Alter 2007). Forrester found out in a recent study among companies in Europe, Middle East and Africa, that nearly every tenth of the surveyed companies is planning to establish a new Enterprise Portal (Lucas, Adrian, Wang et al. 2007). The worldwide market for Enterprise Portals is estimated to grow approximately nine percent annually for the next several years (Cantara, Biscotti and Raina 2007). The significant attention assigned to this field can be particularly attributed to the strong need for integrating business processes and heterogeneous application landscapes as well as exposing them to larger and external audiences (BEA Systems 2007).

From the technical point of view, such portals represent a unique genre of Web-based applications named *Workflow-based Web Applications* (Kappel, Pröll, Reich et al. 2006). While naturally placing high demands regarding workflow execution, rich user interaction and Web service integration, they also share special characteristics inherent to Web-based applications in contrast to conventional applications (Freudenstein, Buck, Nussbaumer et al. 2007). These comprise, among others, fast evolution cycles, an increased importance of usability and aesthetic aspects, a great variety of user interfaces including mobile devices, shorter development timeframes, strong significance of standards, difficulties in achieving effective reuse, and a great variety of involved stakeholders (Deshpande, Murugesan, Ginige et al. 2002; Mendes and Mosley 2006). Facing the immense complexity resulting from these characteristics, a dedicated engineering approach providing models, systems and methodologies for this new class of workflow-based Web applications is required.

Besides the unique characteristics and challenges mentioned above, one key factor arising from a communication and collaboration perspective deserves particular attention: strong *stakeholder involvement*. Evaluations on reasons of software project failures highlight that factors like stakeholder involvement and communication as well as clear business objectives are crucial for a project's success (The Standish Group International 1994-2008; Charette 2005). This holds true particularly in the context of workflow-based Web applications. As they are constructed according to business processes, which naturally encompass a great variety of stakeholders, efficient and effective communications among all participants including the development team are vital. Consequently, a suitable engineering approach should provide dedicated concepts, methodologies and tools in order to establish successful stakeholder collaboration.

1.1 Research Questions and Contributions

In the following, the research questions addressed by this thesis as well as its corresponding contributions are presented. While this section is only intended as an overview, a detailed analysis of the problem domain is conducted in Chapter 2.

The main question that drove the research presented in this dissertation is:

Main Question: *How can workflow-based Web applications be constructed in close collaboration with stakeholders in an efficient and effective way?*

In this context, *efficiency* addresses the dimension of time which is of interest regarding various aspects like duration of development cycles, preparation or learning times for involved stakeholders or the required time span for adopting changes. On the other hand, *effectiveness* concerns the qualitative dimension focusing on aspects like compliance with stakeholder requirements, reducing the potential for misunderstandings, the usability of user interfaces or enabling successful reuse of existing artifacts by improving search quality.

Obviously the above cited main question is rather abstract and covers a very comprehensive range of distinct research areas and problems. In order to convey an overview of the problem scope addressed by this thesis, it will be refined into several more specific questions. Each of these questions addresses a particular research problem either related to unique characteristics of workflow-based Web applications themselves or the associated development and evolution process.

The first question concerns the construction and evolution of workflow-based Web applications:

Research Question Q1: *How can Web applications supporting the distributed execution of long-running workflows by controlling the process flow and providing adequate user interfaces be efficiently constructed and evolved?*

This comprises several aspects ranging from a dedicated engineering methodology to an adequate architecture model. Moreover, it covers the efficient modeling and development of components controlling, persisting, and managing the process flow as well as the way of linking it to corresponding actions and Web-based user interfaces. Beyond that, natural characteristics of Web-based solutions like short evolution cycles, platform- and device-independency and their high degree of distribution have to be addressed.

To this end, a novel engineering methodology, the *Workflow DSL approach*, combining model-driven with component-based software engineering concepts is presented (Freudenstein, Buck, Nussbaumer et al. 2007). It is founded on a standards-based Domain-Specific Language (DSL) for the continuous and iterative modeling of workflow-based Web applications. The DSL's modeling notations allow for specifying the process flow and for mapping process activities to a catalog of logical Web-specific *Activity Building Blocks*, e.g. Web service communication, rich dialog interaction (cf. *Research Question Q2*) or data presentation (Freudenstein, Nussbaumer, Majer et al. 2007). Each of these building blocks is again realized as a DSL requiring only a minimal configuration set, thereby complementing the strong focus on modeling instead of programming and allowing for rapid evolution cycles. The architectural model of the resulting applications is fully service-oriented, thus supporting federative scenarios. In conclusion, the Workflow DSL approach allows

for the efficient construction and evolution of distributed and platform-independent workflow-based Web applications on a pure model basis.

The second question addresses the important aspect of a workflow-based Web application's user interface which mainly consists of complex Web-based Dialog components:

Research Question Q2: *How can complex Web-based dialogs be efficiently constructed and evolved using a methodology that inherently addresses these dialog's high demands on usability aspects?*

For end-users, advanced dialogs represent the main means for work in the context of workflow-based Web applications. Corresponding to the complexity of the task they are designed for, they are usually based on rather comprehensive data models with complex internal dependencies. As they often replace forms that were originally paper-based, most of today's dialog engineering approaches and tools suggest a design methodology similar to those for paper-based forms. As a consequence, both the potentials and the special requirements of dialogs in the Web are often neglected. This in turn results in cognitive overload, low usability and ultimately in poor task and process performance.

The model-driven *Dialog DSL* approach presented in this thesis provides a modeling notation, a technical support system and an engineering methodology that inherently address these challenges (Freudenstein and Nussbaumer 2008a; Freudenstein, Nussbaumer, Allering et al. 2008). Thereby, complex device-independent dialogs with rich behavior and appearance can be efficiently constructed and evolved. The *empirical evaluation* of the Dialog DSL approach showed that advanced dialogs can be constructed and evolved in considerably less time compared to today's Web development frameworks. In the experiment, the participants had no previous knowledge about the Dialog DSL approach, its notation and editor. Even stakeholders without any technical background were able to successfully employ the approach.

The third question refers to the problem domain of reuse in the Web Engineering discipline in general and regarding the construction of workflow-based Web applications in particular:

Research Question Q3: *How can effective reuse across the wide range of heterogeneous artifacts and methodologies in the Web Engineering domain be realized?*

Particularly model- and component-based engineering approaches like those presented in this thesis can benefit considerably from an adequate reuse support, thus improving, amongst others, development efficiency and software quality. However, the efficient realization of reuse should not be considered exclusively for the problem domain of workflow-based Web applications or even only a particular engineering methodology. In fact, to fully utilize the great potentials of reuse, a more holistic perspective, spanning all kinds of artifacts and unifying the existing variety of heterogeneous Web Engineering methodologies is required.

Facing these challenges, the *Web Engineering Reuse Sphere* approach facilitates the efficient and effective reuse across today's Web Engineering methodologies (Freudenstein, Boettger and Nussbaumer 2008). A well-defined ontology for the Web Engineering domain forms its core and allows for innovative and holistic search strategies. To unfold the full potential of reuse, the approach covers both planned and spontaneous reuse, thereby extending reuse to the complete lifecycle of an artifact. Moreover, the approach represents an important contribution to current consolidation activities in the Web Engineering research community. On the one hand, its ontology of the Web Engineering domain enables the unification of diverse methodologies on a conceptual level. On the other hand, it establishes a shared foundation for real cross-methodological reuse.

The fourth question focuses on the strong involvement of stakeholders throughout the process of constructing and developing workflow-based Web applications. Thus, it represents a *cross-cutting concern* affecting all of the presented research questions and contributions.

Research Question Q4: *How can models and methodologies in the context of workflow-based Web applications be designed stakeholder-oriented, thus allowing for continuous, intensified and much more effective stakeholder collaboration?*

As already mentioned in the previous section, the strong involvement of stakeholders throughout all phases of the development process is crucial for a project's success. This holds true particularly for the construction of workflow-based Web applications as the great variety of future end-users knows the concerned business processes and the tasks performed therein best. Thus, enabling stakeholders to contribute actively or even autonomously to the development effort by understanding, validating and specifying parts of the solution being built, is desirable.

Against this background, a new stakeholder-oriented development approach, named *DSL-based Web Engineering* (Nussbaumer, Freudenstein and Gaedke 2006c), is presented. It is based on the idea of employing distinct Domain-Specific Languages (DSLs) for the model-driven specification of a Web application's various aspects, with each DSL being highly focused on a particular problem domain. Existing model-driven Web Engineering methodologies are usually developer-oriented and provide rather heavy-weight and complex modeling approaches. A DSL, however, is designed in a stakeholder-oriented way with strong emphasis on simplicity and employing well-known concepts, abstractions and notations from the problem domain. By incorporating various graphical notations and accompanying editors, a DSL can be tailored to the characteristics and requirements of individual stakeholder groups (Nussbaumer, Freudenstein and Gaedke 2006a). Thus, DSLs are easy to understand, learn and use – both for developers and stakeholders (Nussbaumer, Freudenstein and Gaedke 2006b). Each DSL provides a dedicated software component being able to interpret the models developed with a DSL at runtime. Hence, Web applications can be rapidly constructed and evolved by composing DSL-specific software components and configuring them with DSL programs in terms of models. The

immediate availability of a running application based on a model presents an additional contribution to the effective collaboration with stakeholders.

The DSL-based engineering approach forms the foundation for all models, systems and methodologies related to the three research questions stated above, thus deeply integrating stakeholder-orientation into each of them.

Consequently, the *Workflow DSL* approach, for example, allows the multi-notational modeling of workflows with an extensible set of established and standardized modeling notations and tools. These include, for example, the Business Process Modeling Notation (BPMN) using Microsoft Visio, UML Activity Diagrams designed with IBM Rational Software Developer, Petri Nets employing INCOME2010 or even simple task lists created with Microsoft Word. Thus, stakeholders can specify workflow-based Web applications by using their own languages, i.e. the notations and tools they know best.

Regarding the *Dialog DSL* approach, stakeholder orientation is expressed in terms of a very simple and intuitive modeling notation for highly dynamic and complex dialogs, a supplemental easy-to-use editor and an engineering methodology enabling rapid evolution cycles. As a result, stakeholders can participate in and contribute to the development effort much more intensively than with traditional engineering approaches.

The *Web Engineering Reuse Sphere* finally closes the gap between the set of available DSLs and reusable artifacts on the one side, and stakeholders in need of performing development tasks on the other side. Therefore, it provides advanced ontology-based search facilities which – due to their simplicity – enable stakeholders to search autonomously for adequate resolution strategies (e.g. a DSL) and related artifacts for a given development task. In contrast to existing reuse approaches, stakeholders can thus find appropriate DSLs, modeling notations and reusable artifacts not only according to conventional search dimensions like the given problem domain or keywords but also corresponding to their individual skills and knowledge.

In conclusion, the solutions presented in this thesis represent a significant advancement for the efficient and effective construction of workflow-based Web applications with stakeholders. They have been successfully implemented and evaluated in different scenarios including both formal experimental evaluations and real-world applications, for example in the context of the project “Karlsruhe Integrated Information Management (KIM)” (Juling 2005). In either case, substantial improvements in terms of efficiency and effectiveness throughout the development process were observed. Furthermore, large parts of this thesis were published in numerous papers at international conferences, workshops and journals and intensely discussed with researches from the Web Engineering discipline and adjacent research areas.

1.2 Research Context and Scope

Workflow-based Web applications are mature and complex software systems, representing an own class of Web applications. Like Web applications in general, they use the World Wide Web, i.e. its technologies, standards and paradigms, both as a development platform and as a user platform, leading to the following definition:

Definition 1.1: *A **Web application** is a software system based on technologies and standards of the World Wide Web Consortium (W3C) that provides Web-specific resources such as content and services through a user interface, the Web browser. (Kappel, Pröll, Reich et al. 2006)*

Compared to traditional software applications, the unique characteristics of Web applications require dedicated approaches, methodologies, tools, techniques and guidelines (Ginige and Murugesan 2001). This particularly affects differences regarding the people involved in the development process, the intrinsic characteristics of Web applications and the audience for which they are developed. The Web Engineering research discipline addresses these needs:

Definition 1.2: ***Web Engineering** is the establishment and use of sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high quality Web-based systems and applications. (Murugesan, Deshpande, Hansen et al. 1999)*

While Web Engineering adopts and encompasses numerous principles and methodologies from the software engineering domain, there are many significant differences posing additional challenges and therefore requiring novel Web-specific approaches (Kappel, Pröll, Reich et al. 2006). Beyond that, Web Engineering is a much more multidisciplinary field and encompasses contributions from distinct disciplines such as hypermedia engineering, requirements engineering, usability engineering, information engineering, graphics design, network management, and of course software engineering (Deshpande, Murugesan, Ginige et al. 2002; Mendes and Mosley 2006).

Considering the research questions given in the previous section, this thesis clearly represents a contribution to the Web Engineering research discipline. It promotes the discipline's current state of the art by novel approaches, models, systems, and methodologies for the investigated problem domains. Most of them were presented and discussed with researchers from the Web Engineering discipline and adjacent research fields on international conferences, e.g. the International Conference on Web Engineering (ICWE) 2006-2008 or the World Wide Web Conference (WWW) 2006-2008.

Due to its focus on Web applications supporting the execution of workflows, the discipline of Business Process Management (BPM) seems to be very close to this

thesis. As defined in (van der Aalst, ter Hofstede and Weske 2003), BPM covers the support of business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information. Certainly, some aspects of this thesis overlap with this area, e.g. the homogenization of diverse business process modeling notations in order to allow for multi-notational modeling or the efficient transformation of process models into software components. However, this thesis considers itself mainly focused on Web-specific research questions related to the Web application itself and its user interface as well as methodologies for their efficient development and evolution.

1.3 Thesis Structure

An overview of the structure of this thesis is given in Figure 1-1. Following this introduction, in Chapter 2, a detailed analysis of the thesis' problem domain based on example scenarios from real-world projects is performed. Thereby, a universally valid set of well-defined requirements for an adequate solution is elaborated. Chapter 3 presents existing scientific and commercial approaches related to the given problem scope. Each of these approaches is evaluated against the requirements catalog defined in the preceding chapter, thus highlighting their benefits and drawbacks for this thesis' problem scope and ultimately underlining the need for novel, innovative approaches and methodologies.

The subsequent five chapters contain the main contributions of this thesis. First of all, Chapter 4 gives an overview of the overall solution presented in this thesis as well as its core pillars and their interrelations. This includes the introduction of the DSL-based Web Engineering framework which represents a key foundation for the following two chapters. Based on this overview, each of the solution elements is presented in a separate chapter.

Chapter 5 introduces the Workflow DSL approach for the model-driven construction of workflow-based Web Applications with stakeholders. In Chapter 6, the Dialog DSL approach covering the model-driven and stakeholder-oriented engineering of advanced Web-based dialogs – whether as integral components of a workflow-based Web application or as stand-alone means of user interaction – is presented. The Web Engineering Reuse Sphere, a holistic and stakeholder-oriented approach for effective reuse in the Web Engineering domain in general and in the context of workflow-based Web applications in particular, is described in Chapter 7.

Chapter 8 addresses the evaluation of the presented solutions based on real-world scenarios and formal empirical studies and experiments. Finally, Chapter 9 summarizes the contributions and results of this thesis and gives an outlook on future research directions.

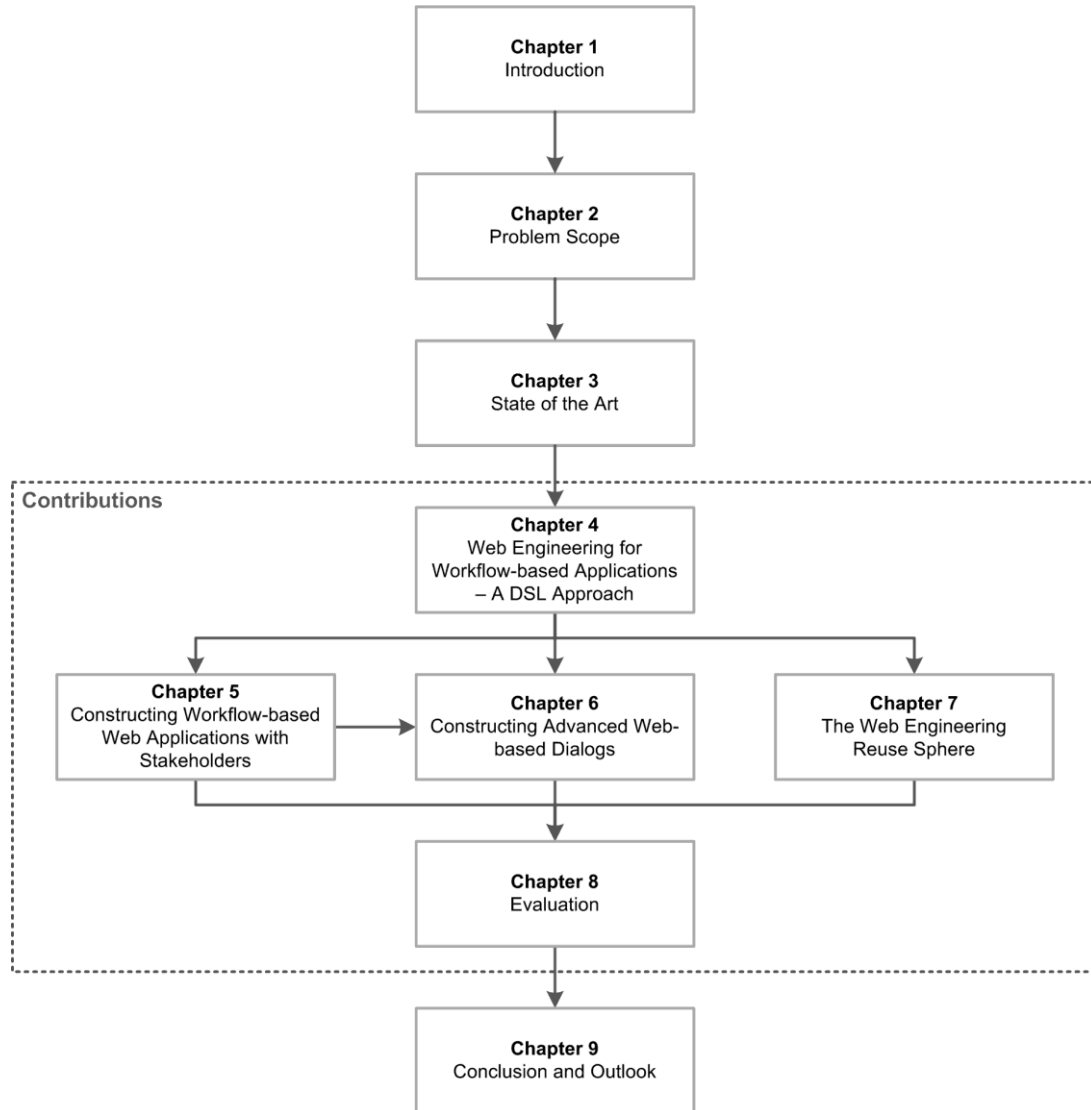


Figure 1-1: Structure of the Thesis

2 Problem Scope

As adumbrated by the research questions in Section 1.1, the problem domain covered by this thesis comprises various aspects. From a technological perspective, workflow-based Web applications impose a great variety of challenges, particularly concerning dimensions like workflow design and execution, application construction and evolution, user interfaces and reuse. On the other hand, significant problems concerning the efficient and effective involvement of stakeholders throughout the development process arise from a communication and collaboration perspective. As they are deeply interrelated with all other, more technological and methodological aspects, they form a cross-cutting requirement dimension.

In this chapter, a systematic and detailed analysis of the problem scope addressed by this thesis will be performed. After a short introduction of general challenges regarding stakeholder collaboration in the Web Engineering domain, the analysis is split into three problem areas, whereby the strong involvement of stakeholders as a cross-cutting concern is examined from a holistic viewpoint for each of these areas. Throughout this chapter, illustrative real-world scenarios are introduced and serve as running examples throughout this thesis. Each section concludes with a summarizing requirements catalog which forms the foundation for the assessment of the current state of the art as well as the design and evaluation of the presented contributions.

2.1 Stakeholder Collaboration in the Web Engineering Field

In the development of complex Web-based solutions, challenges in the communication and collaboration between all involved stakeholders make up a second major problem area besides technical problems and requirements (Nussbaumer, Freudenstein and Gaedke 2006a). Compared to traditional software development projects, both the amount and the heterogeneity of stakeholders in Web application development projects are significantly higher (Escalona and Koch 2004). In the most cases, stakeholders belong to completely diverse departments, have different professional and educational backgrounds and possess various skills and knowledge. Consequently, each group uses its own “language” when talking about aspects of the solution to be built. This becomes obvious especially in the

phases related to requirements management and conceptual design when understanding the various languages of the particular stakeholders is decisive and misunderstandings must be avoided as far as possible. Agreeing on and learning a common language as a potential solution to this, is usually not feasible because of the stakeholders' very limited availability.

Over the last years, a lot of languages and modeling approaches for the (mostly model-based) specification of distributed Web-based solutions have been established (cf. Chapter 3). Most of them attempt to cover their problem domain as exhaustive as possible and therefore include concepts and notations for almost every aspect of the problem domain. This often leads to very expressive and powerful modeling languages, being a good means of conceptual and logical design within the developer team. However, regarding the communication and collaboration with stakeholders, these heavy-weight approaches are too complex and thus not appropriate. For stakeholders being predominantly non-programmers, it would cost too much time and effort to learn these extensive modeling languages and notations.

The crucial influence of strong stakeholder involvement on a project's success was proved in comprehensive studies (The Standish Group International 1994-2008; Charette 2005) and taken on in agile software development methods (Cockburn 2006). Thus, also the Web Engineering discipline needs more stakeholder-oriented approaches intensifying the collaboration and supporting efficient and effective communication throughout all phases of the development process (Nussbaumer, Freudenstein and Gaedke 2006c). The vision should be to provide languages and notations tailored to the characteristics and needs of individual stakeholder groups and thus enabling them to directly contribute to the development effort by understanding, validating, and even autonomously specifying aspects of a Web-based solution.

2.2 Workflow-based Web Applications

Over the last decades, the nature of a company's business processes as well as its IT landscape has transformed tremendously. Whereas previously competitive advantage was often considered from a product-oriented perspective, differentiation in a service-oriented world like today is predominantly based on the uniqueness of services and business processes (Heskett, Sasser and Schlesinger 1997). Likewise, the IT landscape has evolved significantly from a rather centralized and homogeneous to a highly distributed and heterogeneous structure (Bieberstein, Bose, Fiammante et al. 2006).

Consequently, business processes today are more complex, they span a great variety of roles, organizational units and even companies, they suffer from media-discontinuity issues, and they involve diverse heterogeneous and distributed IT systems and applications. Moreover, agility in terms of the flexibility to rapidly meet changed or new business demands represents one of the most valuable assets for

companies. Whereas in the past, business processes changed once a year, they now change once a month or sometimes even once a week (Carter 2007). As a consequence, business process performance often turns out to be rather poor and error-prone.

To this end, companies started to adopt business process management suites for realizing integrated and controlled end-to-end business processes. Such systems allow for modeling, controlling and monitoring business processes in form of workflows across diverse roles, organizations and IT systems. Prominent examples include the IBM WebSphere Suite, the Oracle BPM Suite, or the SAP NetWeaver BPM Suite.

As indicated by the previous sentence, there is a clear separation between the terms “business process” and “workflow”, even though they are often used as synonyms. The Workflow Management Coalition (WfMC) defines these terms as follows (Workflow Management Coalition 1997):

Definition 2.1 - Business Process: *A set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships.*

Definition 2.2 - Workflow: *The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.”*

Hence, the main difference between these terms lies in the fact that the term “workflow” emphasizes the aspect of software-based enactment of a business process which is supported by a “workflow management system”.

Gartner Research predicts that, by 2009, 20% of business processes of Global 2000 companies will be supported by workflow management systems. Unlike prevailing system- and application-oriented processes without human interaction, these processes will predominately involve a considerable amount of human work that differentiate the company from its competitors and that is poorly supported by established IT systems (Hill, Sinur, Flint et al. 2006). This growing importance of human tasks within workflows leads to the question of adequate user interface concepts and dedicated methodological solutions.

To this end, Web-based applications and portals as uniform and integrated access channels for initiating and interacting with such workflows have recently gained significant attention (Gootzit, Phifer, Valdes et al. 2008). Based on the WfMC definition for “workflow application”, i.e. “software programs that interact with a workflow enactment service and handle [...] the processing required to support a particular activity or activities” (Workflow Management Coalition 1997), and considering that they are Web applications in the sense of Definition 1.1, they are termed “*Workflow-based Web Applications*”. As such, they combine the potentials and benefits of both fields.

Consequently, the main benefits which can be realized by workflow-based Web applications – provided that they can be implemented in an efficient and effective way - comprise (Puschmann and Alt 2004; Phifer 2006; BEA Systems 2007; Gootzit, Phifer, Valdes et al. 2008):

- increased efficiency and quality of business processes and individual tasks
- integrated provision of previously siloed applications, systems and processes
- consistent and improved user experience
- replacement of multiple application-specific user interfaces and accounts
- ubiquitous access and availability
- reduction of paper-based or manual processes
- reduced maintenance efforts
- increased business flexibility

Depending on the business processes they support, workflow-based Web applications can serve a great variety of target audiences. These include employees in a Business-to-Employee (B2E) context (often referred to as “Enterprise Portals”), customers in Business-to-Consumer (B2C) portals as well as business partners within Business-to-Business (B2B) Web sites.

These great potentials and expectations towards workflow-based Web applications come along with a significant, multi-faceted complexity regarding their construction and evolution. In the following subsection, specific challenges arising from a technical perspective will be analyzed, whereas Section 2.2.2 examines challenges in the context of collaborating and communicating with stakeholders throughout the development process.

2.2.1 Technical Challenges

In the following, the specific technical challenges arising in the construction and evolution of workflow-based Web applications will be elaborated based on an example scenario. The scenario deals with a workflow-based Web application for the handling of an exemplary “business trip” process which could be found in a company’s intranet. The scenario and its characteristics were chosen in a way that ensures the universal validity of the resulting challenges and requirements an adequate solution should address.

Figure 2-1 shows a simplified excerpt from the current “business trip” process at the Karlsruhe Institute of Technology (KIT) focusing on the reimbursement process after the employee has returned from the trip. In this context, a variety of roles and departments from all over the university is involved, e.g. the employee, the travel department, the traveler’s institute director or manager and the accounts department. In earlier phases of the business process, also external organizations like partner travel agencies could be included. The example starts with the employee creating a travel expense report, which is currently done using a paper-based form (Karlsruhe Institute of Technology 2009). Next, the filled form is sent per internal

mail to the travel department, which processes it, creates a refund statement and sends it back to the employee via internal mail (potential follow-up-inquiries by the travel department and associated replies by the employee were left out in this example). Next, the employee checks the refund statement and could possibly have objections. In this case, an objection handling process, again paper-based and via internal mail, follows. Afterwards, the employee's institute director or manager receives a payment statement in order to approve it (the sub-process in case of a rejection was left out in this example). Subsequently, the accounts department processes the payment and the employee receives a payment notification.

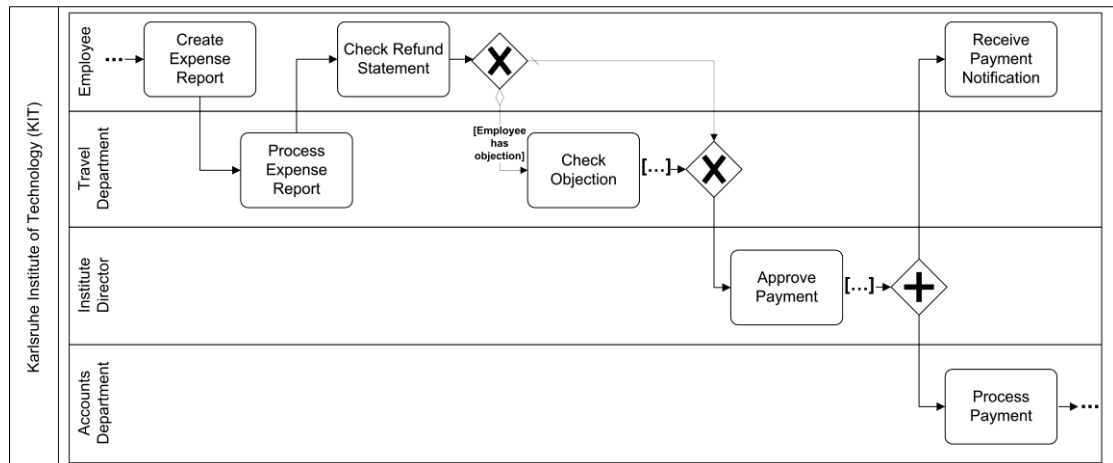


Figure 2-1: Simplified Excerpt from the “Business Trip” Business Process

A major technical challenge in the context of the development of workflow-based Web applications lies in the technical implementation of the business process to be realized. This comprises the construction of a *workflow component* according to the given business process model or an enhanced workflow model respectively. Such a component should allow for instantiating new workflow instances as well as controlling and ensuring the exact process flow and the correct distribution of tasks to roles. Moreover, with respect to long-running workflows with possibly long time spans between activities, it must be capable of making the current state of the workflow persistent and resuming at a later point in time. In addition, the workflow component is in charge of responding to queries about the current status of a workflow instance or the available tasks for a given user as well as supplying and receiving data objects as input or output parameters of tasks.

The implementation of such a workflow component can be supported by a workflow engine, e.g. Microsoft Windows Workflow Foundation (Microsoft Corp. 2006c) or IBM WebSphere MQ Workflow (IBM 2006); however, a significant amount of work and complexity remains for constructing a workflow program as input for the workflow engine as well as integration challenges regarding the Web application as a whole and its user interface in particular – both for the enactment of human tasks as well as for consuming infrastructure services provided by the engine.

Analyzing the particular tasks of the “business trip” example process above, various activity types to be supported by a workflow-based Web application can be identified. A great majority of tasks, e.g. “Create Expense Report”, “Process Expense

Report” or “Check Refund Statement”, requires *rich dialog-based human interaction* (cf. Section 2.3). Others consist of *presenting data* to a user, e.g. “Receive Payment Notification”. Some tasks, e.g. “Process Payment”, rely on data to be retrieved from or stored in heterogeneous backend systems or even services from external providers. Such scenarios are usually realized via *Web service communications*, which can occur either in the context of autonomous system activities or in combination with the mentioned activity types. In advanced workflow scenarios, e.g. in supply chain management, there could also be physical activities like shipping a package or having an in-person meeting. In order to realize an end-to-end support for business processes, these tasks need also to be considered and realized as some kind of completion confirmation dialog.

Besides such tasks requiring only a completion confirmation, there are tasks which also take place outdoors or at least away from a desktop or notebook computer and which could be better supported by one of the aforementioned activity types. In such cases, process participants should be able to collaborate using other devices, e.g. PDAs or smart phones. This raises the requirement for *mobile and device-independent access*, which is a common, primarily user interface-related, requirement for advanced Web-based applications (World Wide Web Consortium 2007).

Beyond that, some tasks are more efficiently conducted in dedicated, task-specific client applications, e.g. a spreadsheet application (Kotoric 2007). Thus, even though trying to offer one integrated, browser-based user interface is a desirable objective, a workflow-based Web application should also supply the technical foundation for completing tasks off the browser. Thus, the architectural model of the application should be designed in a way that allows for *additional workflow interaction channels* besides the Web-based user interface.

This represents also a crucial requirement for workflow-based Web applications in a B2B context as they span not only diverse organizational roles and departments, but also multiple companies. The fact that the workflow application is a Web application already entails the potential to make it available to a global audience. Sometimes, however, in such scenarios, external organizations prefer to have their employees participate in the workflow from within their own portal applications. In this respect, apart from federation-related security challenges (Menzel, Thomas, Wolter et al. 2007), these external portal applications can be considered like non-browser client applications as described before.

For the technical realization of a business process’ particular tasks by a workflow-based Web application, an adequate approach should provide *activity building blocks* for the aforementioned human- and system-oriented activity types. In addition, the workflow-based Web application’s *architecture model* as well as its user interface should be prepared for access using mobile devices as well as internal and external client applications. The demand for a workflow-based Web application’s full coverage of all occurring tasks within a business process holds great benefits in terms of process efficiency and quality as well as solving media-discontinuity issues, but also reveals a great amount of development effort and technical complexity

which in turn underlines the necessity of adequate models, systems and methodologies.

A further problem area concerns the aspects of *agility and evolution*. Web applications in general and workflow-based Web applications in particular underlie a continuous evolution due to frequent changes (Roger S. Pressman 2005), e.g. adjustments in the business process' structure, integration of new partners or changes in dialogs or presentation design. For example, in the presented "business trip" scenario, various types of changes and extensions like changed responsibilities, integration of new travel agencies or external partner services, modifications to business rules, extensions to forms or the integration of new backend systems are imaginable, particularly in the context of the current merger of the University of Karlsruhe (TH) and the Forschungszentrum Karlsruhe towards the Karlsruhe Institute of Technology (KIT) (Karlsruhe Institute of Technology 2007). Furthermore, compared to traditional software projects, the development timeframes of Web applications are in most cases significantly shorter (McDonald and Welland 2001).

Thus, agility in terms of supporting short revision lifecycles and the efficient adoption of such changes is essential. To this end, a model-driven software development approach seems to be a promising option as it allows for comparatively easy changes in the models which then need to be – ideally automatically - propagated to the actual implementation (Stahl and Völter 2006). However, achieving a *continuous model-based approach* throughout all phases of the development process as well as assuring *consistency* between the various models and the implementation represent challenging requirements.

With respect to the great diversity of artifacts produced during the development and evolution of Web applications and particularly workflow-based Web applications as well as facing requirements like strong support for agility and evolution, development efficiency and software quality, effective *reuse* strategies become a key factor. On the one hand, an adequate approach for the construction and evolution of workflow-based Web applications should explore the potentials of *component-based software engineering* (Sommerville 2007b). An analysis of the various task types as outlined above could represent a starting point for the definition of highly generic and reusable software components for the implementation of workflow tasks. On the other hand, the systematic reuse of all kinds of artifacts throughout the development process should be examined (cf. Section 2.4).

2.2.2 Stakeholder Collaboration for Workflow-based Web Applications

Considering the construction and evolution of workflow-based Web applications, particular challenges regarding the efficient and effective collaboration with stakeholders arise. During the specification of the envisioned solution, a multitude of stakeholders has to be involved. For the presented example process excerpt, one or more representatives from each of the participating roles, i.e. traveling employees, the travel department, institute directors and the accounts department, have to be

included. As the example shows a small part of the process only, the amount and heterogeneity of concerned stakeholders are much higher for the complete business process. The effective communication with these groups is crucial, as they know the business process best and, unlike other types of Web applications, there is very few room for specifications based on assumptions.

The specification process for workflow-based Web applications can be differentiated into business process modeling and workflow modeling. The former focuses on the underlying business process, its structure and exact control flow, whereas the latter addresses its technical realization by the workflow-based Web application.

During business process analysis and specification, business process models represent the main design artifact and different stakeholders contribute to different sections of the process. A first step in the requirements analysis usually lies in the elicitation of tasks and corresponding roles of the considered section. According to the presented example scenario, Table 2-1 shows a result of this early elicitation activity in form of a – usually pen and paper-based - table.

Table 2-1: *Table-based Elicitation of Tasks and Roles of a Business Process Excerpt*

Task	Role
Create Expense Report	Employee
Process Expense Report	Travel Department
Check Refund Statement	Employee
Check Objection	Travel Department
Approve Payment	Institute Director
Receive Payment Notification	Employee
Process Payment	Accounts Department

Depending on the background and skills of the interviewed stakeholder, this can be an important pre-stage to analyzing the detailed process flow in order to reduce the complexity and to achieve a step-by-step specification process. The table-based specification also abstracts from a concrete business process modeling notation, thus minimizing the stakeholder's required prior knowledge. For the succeeding specification activities, a member of the development team has to transfer the gathered information into a graphical model which can be rather time-consuming and already leaves room for mistakes.

The subsequent specification of the business process' detailed structure and control reveals additional problems. To date, there is still no widely-accepted standard notation for business process modeling (Hill, Sinur, Flint et al. 2006). This holds true also for business process management suites and related tools where many vendors still employ proprietary notations. However, particularly in such a diversified context, the *adherence to existing standards* – where possible - becomes even more important. Thus, stakeholders might already know or even use a notation and corresponding existing tools, which is not the fact for proprietary approaches.

This heterogeneity further aggravates the collaboration with stakeholders as they are confronted with diverse notations and tools from project to project. Hence, their experiences, knowledge and skills in these fields are likewise very heterogeneous. As a result, when specifying business processes with stakeholders across an organization, e.g. in the project “Karlsruhe’s Integrated Information Management (KIM)” (Juling 2005), some of them prefer Petri nets as a means of communication as they play a major role in their research context (Klink, Li and Oberweis 2008). Others, for example, favor the Business Process Modeling Notation (BPMN) (White 2006) or the Unified Modeling Language (UML) (Object Management Group 2005b) for the same reason or due to prior project experiences with this notation and related tools. And people without any prior experiences or with a non-technical background, e.g. in humanities and social sciences, often like a notation in natural language better. Figure 2-2 illustrates these differences by showing the “business trip” example process excerpt in four different notations and tools: BPMN with Microsoft Visio, Petri Nets with INCOME2010, UML Activity with IBM Rational Software Architect and a text-based notation (cf. Table 2-1) with Microsoft Word.

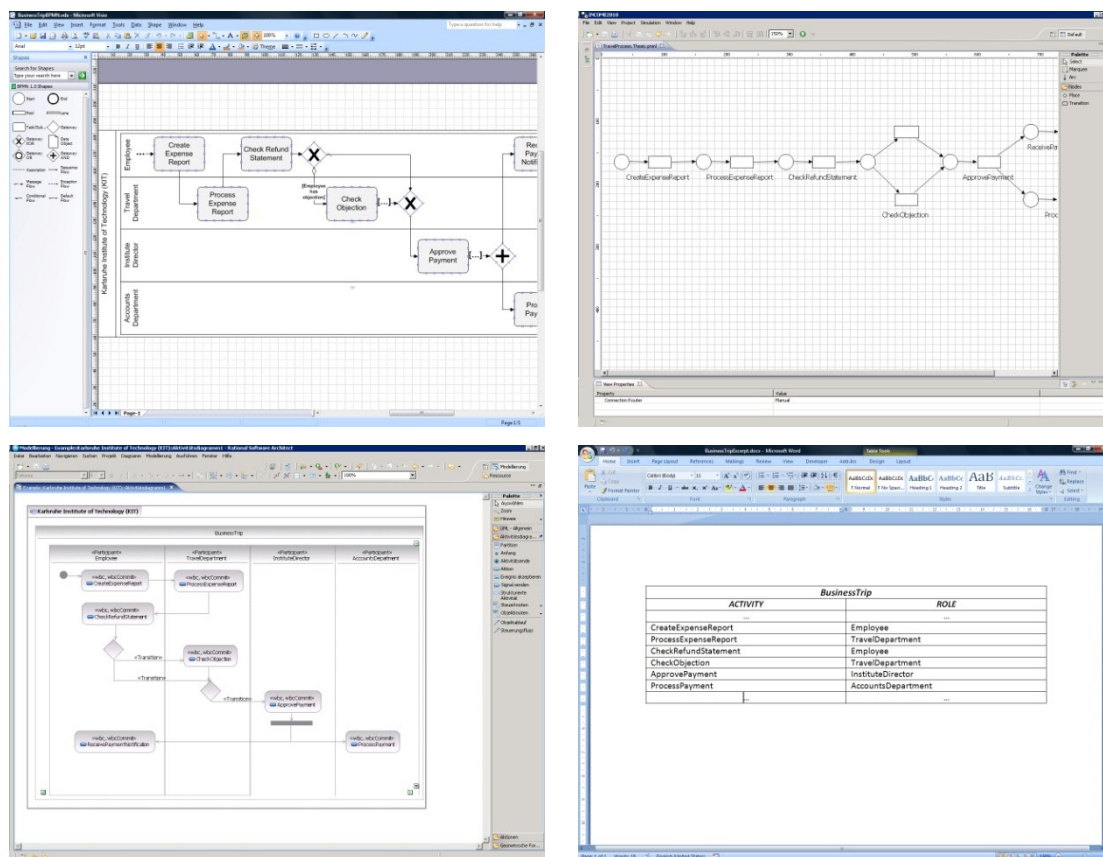


Figure 2-2: Various Business Process Modeling Notations and Tools

Considering the fact that different stakeholders contribute to different sections of one business process model, agreeing on a common language, notation and tool seems inevitable. Some business process modeling notations appear to some extent similar for experienced process analysts and can be classified into process modeling language families (Recker and Dreiling 2007). However, stakeholders are usually rather low experienced and the challenge of multi-notational modeling across these

families still remains. This in turn results in high learning efforts and inefficient communications, raises the probability of misunderstandings as well as hinders long-term efficiency gains. Moreover, the potential of reusing already existing business process models, which were created e.g. for documentation purposes, is significantly constrained to models that were created with the same notation and the same tools.

A consolidated view of the presented challenges indicates the necessity of a *stakeholder-oriented* approach throughout all stages of the business process modeling phase. Such an approach should support stakeholders in collaboratively validating, modifying and even creating shared business process models by allowing each stakeholder to use her *preferred notation and tool*. Likewise, *model consistency* must be persevered across notations, tools and process phases. An ideal approach should not be limited to specific notations or tools, but rather provide well-defined extension points and corresponding processes.

The subsequent workflow specification focuses on the technical implementation of the modeled business process by means of a workflow-based Web application. In this regard, the realization of each task in the business process has to be clarified and designed in close collaboration with the relevant stakeholder groups. Besides a high-level mapping of tasks to technical activity building blocks like e.g. dialog-based interaction, data presentation or Web service communication as described in Section 2.2.1, a detailed design for each of these has to be conducted. In the case of a dialog-based interaction, for example, developers and stakeholders have to collaboratively specify the detailed dialog design in a way which assures a maximum efficiency and effectiveness for future users. The strong technical nature of these specification tasks further aggravates efficient and effective communications and entails the risk of misunderstandings.

Against this background, *hiding technical complexity* becomes a key factor. As the business process model is already known from the business process modeling phase and thus still serves as a reference point for stakeholders, it is desirable to continue to use it as a key artifact also in the workflow design. While usually the logical and physical design of a Web-based solution is performed employing developer-oriented modeling notations including lots of technical details, a stakeholder-oriented approach should aim at *preserving the stakeholder's business perspective*. Thus, it should provide a much *higher abstraction level* and should include technical details only where necessary. For example, when designing the technical implementation of tasks in a business process, it is desirable that one activity from a business perspective can be realized by one technical activity building block and has not to be split up into several activities from a system perspective. Thereby, the business process model's structure can be kept throughout all stages of the development process, easing the understanding and orientation for stakeholders. Furthermore, the potentials of minimizing and hiding as much technical details as possible should be explored thoroughly. By exploiting automation potentials and component-based construction strategies, a stakeholder-oriented approach should strive for reducing the required initial design set to a minimum.

To this end, principles and techniques from the model-driven software development field can be a possible solution (Hailpern and Tarr 2006). On the one hand,

stakeholder and developers can continue to specify the design based on a model of the business process throughout all phases of the development process. On the other hand, models raise the level of abstraction and are a good means for hiding complexity. However, model-driven software development can also pose additional problems if not applied maturely. First of all, a problem termed “model-code gap”, i.e. the semantic distance between a model artifact and its code representation, has to be considered (Warmer and Kleppe 2003). This common disparity requires developers to transfer specifications on a model basis into executable program code, which leaves room for interpretation and misunderstandings as well as poses challenges regarding model redundancy and consistency. This also can result in roundtrip engineering problems occurring in the case of evolution. A further means for supporting efficient and effective stakeholder collaborations is the provision of *evolutionary prototypes* of the application from the very beginning of the development process (Wieggers 2003). For stakeholders, they represent an ideal basis for clarifying and completing requirements, exploring design alternatives and visualizing development progress.

Facing these challenges, a stakeholder-oriented solution should provide adequate modeling support for all phases of the development process and all relevant stakeholder groups. At the same time, it should assure model consistency and preferably achieve a direct, lossless and complete mapping of models into running applications and services.

2.2.3 Requirements Catalog for the Dimension Workflow

The following catalog briefly summarizes the identified requirements for the problem dimension Workflow:

- **R-WF-01 – Workflow Management and Execution:** Workflow-based Web applications should be capable of managing and executing long-running workflows comprising diverse roles.
- **R-WF-02 – Continuous, Rich Web-based User Interface:** Besides the realization of system-oriented process activities, an adequate engineering approach should support the efficient construction of a continuous, rich Web-based user interface for all human workflow tasks.
- **R-WF-03 – Multimodal Participation:** In addition to the Web-based user interface, process participants should be able to collaborate using various client devices and applications across diverse platforms.
- **R-WF-04 – Federation-Enabled and Component-Based Architecture Model:** A workflow-based Web application’s architectural model should be designed with respect to federative scenarios as well as explore the potentials of component-based software engineering for realizing workflow activity building blocks.

- **R-WF-05 – Agility and Evolution:** A suitable approach should support agility in terms of short revision lifecycles as well as efficient adoption of evolutionary changes.
- **R-WF-06 – Multi-Notational Modeling:** Stakeholders should be able to understand, validate, modify and create shared business process and workflow models using various notations and tools at the same time. Extensibility for new notations and tools should be assured in a non-invasive way.
- **R-WF-07 – Complete Model Continuity and Consistency:** Models should be the primary means of analysis and design as well as automate development throughout all phases of the development process. At the same time, continuous model consistency has to be assured.
- **R-WF-08 – Lightweight Modeling Approach:** In contrast to existing heavy-weight modeling approaches, the models, notations and methodology for specifying workflow-based Web applications should be stakeholder-oriented, i.e. hiding technical complexity and focusing on the essentials, thus fostering simplicity and usability.
- **R-WF-09 – Use of Standards:** Existing standards in the fields of modeling notations, process and workflow languages, interchange formats as well as existing tools should be incorporated where possible.

2.3 Web-based Dialogs as Primary Interaction Mediums

Complex dialogs with comprehensive underlying data models as well as a high intensity and complexity of user interaction aspects are gaining increasing importance in today's Web applications (O'Reilly 2005; Phifer, Gootzit, Sholler et al. 2007). Particularly in workflow-based Web applications, Web-based dialogs represent the primary means for work. For example, in the presented "business trip" process excerpt (cf. Figure 2-1), the great majority of tasks, e.g. "Create Expense Report", corresponds to dialog-based user interaction within a workflow-based Web application. Considering the significant complexity of most of these tasks as well as the comprehensive underlying data models, highly dynamic dialogs reducing cognitive overload and offering guidance to the users are required.

In the following, universally valid challenges and requirements for the construction and evolution of such advanced Web-based dialogs will be elaborated based on the "Travel Expense Report" dialog, (e.g. (Karlsruhe Institute of Technology 2009)). The travel expense report includes personal data, the detailed travel itinerary as well as all incurred expenses. Thus, the associated data model is quite comprehensive and includes several context-dependent elements. For example, the required information differs depending on the travel destination (national / international) and the used means of transport (e.g. train, plane, car).

As the Web-based dialog's *usability* is strongly interrelated to the user's efficiency and effectiveness in completing the task (Nielsen 2005), the integration of usability factors in a development methodology becomes a success factor (Matera, Rizzo and Carughi 2006). Beyond that, several studies proved that the general consideration of usability aspects during the development process can lead to significant cost savings by decreasing the amount of later changes (Nielsen and Landauer 1993; Madsen 1999). The most widely adopted definition of the term "Usability" was introduced by Nielsen (Nielsen 1993):

Definition 2.3 - Usability: *Usability is a quality attribute that assesses how easy user interfaces are to use. [...] Usability is defined by five quality components: Learnability, Efficiency, Memorability, Errors, and Satisfaction."*

In this context, *learnability* refers to how easy it is for users to accomplish basic tasks the first time they encounter the design. *Efficiency* addresses how quickly users can perform tasks once they have learned the design. *Memorability* means how easily and quickly occasional users can reestablish proficiency after a period of not using it. *Errors* concerns the number and severity of errors made by users as well as how good the system supports users both in preventing and correcting them. *Satisfaction* covers how pleasant it is for users to use the design.

To this end, guidelines and *best practices* for the usability-oriented design of Web-based dialogs have been developed (Nielsen 2005; Preciado, Linaje, Sanchez et al. 2005; Wroblewski 2008):

- Reducing cognitive overload by semantic partitioning
- Selection-dependent inputs
- In-context help and hints
- Immediate feedback and meaningful error indication
- Consistent form layout
- Clear path to completion
- Visual continuity

However, the implementation of such usability features still remains complex and error-prone. Above all, developers have actually to be aware of such usability principles and best practices. The fact that a large amount of Web sites still ignores well-documented usability guidelines and best practices (Nielsen and Loranger 2006) highlights the necessity of integrating usability aspects already in the dialog design and development. Existing Web Engineering methodologies and commercial form development tools leave the enforcement of usability factors to the attention and perception of designers and developers. Even worse, many of them implicitly suggest a rather static form design known from paper-based forms, thus neglecting the potentials and expectations for Web-based dialogs.

A significant proportion of improvements could already be implemented at development time instead of being recognized in later usability tests. This often results – depending on the time of identification - in additional development cycles or even costly redesign projects, e.g. (Herman 2004). Consequently, models, tools and methodologies should inherently address usability aspects and provide guidance

to the developer. In this regard, user interaction patterns can be helpful (Welie and Trætteberg 2000).

Beyond these usability aspects, Web-based dialogs should be usable *platform- and device-independently*, whereby each device can possess different characteristics such as screen resolution. With respect to the travel expense report, travelers might want to fill out parts of the expense report during their journey using a mobile device (e.g. a PDA). An adequate engineering methodology as well as an associated technical framework should be capable of adapting a dialog to the characteristics of requesting client devices at runtime. In this context, design models should allow for specifying rule-based guidelines on how a dialog may be adapted in order to preserve its usability, e.g. regarding logical partitioning.

Besides these dialog-specific requirements, further challenges arising from a development- and project management perspective exist (Freudenstein and Nussbaumer 2008b). As already mentioned in Section 2.2, Web applications in general and workflow-based Web applications in particular underlie a continuous *evolution* due to frequent changes. This holds true especially for their dialogs, e.g. due to adaptations in the data model, design or layout changes, or completely new requirements (Pressman 2005b). Thus, a suitable dialog engineering approach should be *agile* in terms of supporting short revision cycles and the efficient adoption of changes.

From the technical point of view, the integration of *Web service communication* for retrieving updates of the dialog's data model or for submitting the final user input is a common requirement found in service-oriented and workflow-based Web applications. Regarding the travel expense report example, Web service communication could be required to submit the expense report to a legacy backend system or to provide auto-completion features using external Web services. Hence, data models in form of XML Schema specifications (Thompson, Beech, Maloney et al. 2004) or integrated in Web Service Description Language documents (Christensen, Curbera, Meredith et al. 2001) usually form a starting point for the dialog design. With respect to requirements related to development efficiency and rapid prototyping, the automated generation of running, Web service-enabled dialogs based on such data models is desirable.

An additional requirement arising from a technical perspective concerns the resulting dialog's implementation language. As powerful *standardized markup languages* for the technical implementation of Web-based dialogs have been introduced in the last few years, e.g. XForms by the W3C (Boyer, Dubinko, Leigh L. Klotz et al. 2007) or XAML by Microsoft (MacVittie 2006), their potentials should be leveraged in an engineering approach for advanced Web-based dialogs. Thereby, on the one hand, the extensibility and reusability of the resulting dialogs can be improved. On the other hand, the engineering approach's applicability and utility can be increased as well as the dissemination of these new markup languages can be supported.

Throughout the development processes of the various Web-based dialogs within a workflow-based Web application, a multitude of *stakeholders* from different organizational units with diverse professional backgrounds and skill levels has to be

involved. In order to improve the efficiency and effectiveness of the collaboration, the employed modeling notations have to be as simple and intuitive as possible and should focus on relevant dialog-specific aspects while hiding unwanted complexity. Furthermore, an associated, easy-to-use editor for creating and adapting dialog models is desirable. Such an editor would ideally be Web-based, thus easing location- and platform-independent development. Assuming the existence of effective approval processes, stakeholders could thereby autonomously perform modifications and extensions to existing dialog models or even design new ones.

Evolutionary prototypes or the completely automated transformation of dialog models into running dialogs respectively, can help to achieve a common understanding and to identify discrepancies between requirements and their realization (Wieggers 2003). Design alternatives, e.g. targeting usability aspects, can be explored and misunderstandings can be resolved at an early, yet cost-efficient point of time.

Considering the immense number of Web-based dialogs being developed at a university over time, the strong integration of *reuse* in all phases of the development process is desirable. The systematic reuse of various artifacts, e.g. data models, dialog models (in part or whole), as well as software components contributes particularly to development efficiency and software quality (cf. Section 2.4).

2.3.1 Requirements Catalog for the Dimension Dialog

The following catalog briefly summarizes the identified requirements for the problem dimension Dialog:

- **R-D-01 – Usability:** An adequate engineering methodology should treat usability aspects as vital features of advanced dialogs. Thus, it should provide guidance and strong support for the efficient implementation of highly usable Web-based dialogs.
- **R-D-02 – Device Independence:** The resulting dialogs should be accessible and usable device-independently. Therefore, they should be reasonably adapted to client characteristics at runtime.
- **R-D-03 – Web Service Support:** Web service endpoints should be considered as an important submission channel for completed dialogs. Thus, the automated generation of running Web service-enabled dialogs according to WSDL-based or XML Schema-based data models should be supported.
- **R-D-04 – Agility and Evolution:** A dedicated dialog engineering approach should be agile and evolution-oriented in terms of supporting short revision lifecycles and the efficient adoption of changes. Therefore, it should allow for efficiently constructing and evolving advanced Web-based dialogs on a pure model basis.

- **R-D-05 – Stakeholder Involvement:** The modeling notation should be stakeholder-oriented in terms of harnessing technologically complex dialog aspects with strong emphasis on simplicity. A supplemental editor should be easy to access and use, also for stakeholders without technical background.
- **R-D-06 – Standardized Markup Language:** Regarding the automated transformation of dialog models into executable code, standardized dialog markup languages should be supported.

2.4 Effective Reuse

Reuse has been identified very early as an important software engineering principle being able to significantly improve development efficiency and quality (McIlroy 1968). In fact, reuse can lead to greater schedule and effort savings than any other rapid-development practice – if implemented as a systematic and dedicated long-term strategy and supported by an effective framework (McConnell 1996). Likewise, the preceding analysis mentioned effective reuse strategies as an important requirement an adequate engineering approach for workflow-based Web applications or Web-based dialogs respectively should address.

In the Web Engineering research field, aspects of reuse have primarily been examined in the context of a particular Web Engineering method and focusing on specific artifact types like models or components, e.g. OOHDM (Schwabe, Esmeraldo, Rossi et al. 2001), WebComposition (Gaedke and Rehse 2000) or WebML (Ceri, Fraternali and Matera 2001). While most of the Web Engineering approaches describe their modeling methodology's adequacy for reuse, the efficient and effective realization of reuse when developing Web applications still remains nontrivial.

Beyond that, consolidation efforts like the "Model-Driven Web Engineering Initiative (MDWEnet)" (Vallecillo, Koch, Cachero et al. 2007) or research papers, e.g. (Selmi, Kraiem and Ghezala 2005), strive for achieving interoperability between common Web Engineering methodologies and their tools. Thereby, not only the significance of a unifying reuse approach is emphasized, but also the immense potential of reuse in interoperable, cross-methodological Web Engineering scenarios is underlined. Consequently, a reuse strategy should be *independent* from the development methodology used. An adequate reuse approach should therefore provide positive impact on any Web Engineering methodology and should establish a common basis for *cross-methodological reuse*. Especially in the context of the above-mentioned consolidation efforts, a unifying approach unfolding the power of cross-methodological reuse is desirable.

To date, the integration of *stakeholders* and their specific characteristics and demands have not been considered in reuse-related Web Engineering research yet. However, the goal of enabling stakeholders to directly contribute to the development process also requires their strong consideration and dedicated support

regarding reuse aspects. On the one hand, adapting existing artifacts is in the most cases much easier than creating new artifacts from scratch - especially for people with few technical skills. On the other hand, assuming a cross-methodological context, the choice of the Web Engineering methodology used for the realization of a particular feature depends, amongst others, on the given *stakeholders' skills* and the *qualifications* required by the methodologies. Thus, empowering stakeholders to find reusable artifacts, methodologies and tools suitable both for the given problem and their individual knowledge and skills, is crucial - irrespective from the Web Engineering methodology used.

As the positive effects of reuse are not restricted to particular types of artifacts, a systematic Web Engineering reuse approach should be *generic* in terms of supporting any type of artifact occurring in the development process (Freeman 1983). In this regard, it is desirable to non-invasively build on *existing artifact stores*, e.g. document repositories, model databases, component repositories or version control systems.

When developing with reuse, efficiently and effectively finding suitable reuse assets is crucial (Krueger 1992). The common way of searching on a keyword or full-text basis is usually not sufficient though. In fact, an appropriate search mechanism should strongly incorporate the current context (Tracz 1990), e.g. the project and application type, the given task and process phase, the involved stakeholders, the feature's associated business domain, the Web-specific concern etc. Such complex context-dependent search queries are often not directly resolvable, but rather require knowledge-based resolution strategies. Thus, powerful *semantic inference-enabled search capabilities* tailored to the Web Engineering domain should be provided. Especially for users having little experience in searching for suitable artifacts, it can be difficult to determine good search parameters. To this end, enabling users to browse through the registry space can further increase efficiency and effectiveness (Frakes and Pole 1994).

Having found a suitable artifact, reusing and *integrating* it should be very efficiently. Therefore, finding and retrieving artifacts should be possible within the specific proprietary tools and applications where they are used in. For example, business process models and templates should be directly searchable and retrievable from within the associated business process modeling tool. In the context of reusing software components, it is desirable to have direct installation and integration capabilities at runtime, ideally augmented by safe preview facilities for integration testing (Pressman 2005a).

In large projects or organizations or particularly in a global context, it happens quite often that a particular artifact is needed by several parties but does not exist in the repository. Then, each party individually starts with developing for reuse, which in turn leads to a considerable amount of redundant development effort. Supported by an effective *coordination mechanism* which indicates ongoing development efforts and allows for spontaneous ad-hoc reuse at an early point in time, such parallel developments could be efficiently aligned (Yongbeom and Edward 1998). In this regard, challenges concerning the automated derivation of basic metadata from the current context have to be considered (Boldyreff, Nutter and Rank 2002).

2.4.1 Requirements Catalog for the Dimension Reuse

The following catalog briefly summarizes the identified requirements for the problem dimension Reuse:

- **R-R-01 – Generality and Homogenization:** An adequate reuse approach should be generally applicable to today’s Web Engineering methodologies, artifacts and frameworks and establish a homogenizing basis for cross-methodological reuse.
- **R-R-02 – Stakeholder Orientation:** Stakeholder characteristics should be treated as an important context parameter for storing and finding artifacts. From a usability perspective, performing such core operations should be rather intuitive, requiring only little technical knowledge.
- **R-R-03 – Semantic Search:** Advanced semantic context-dependent search capabilities tailored to the Web Engineering domain should be provided. In addition, facilities for browsing through the registry space should be offered.
- **R-R-04 – Integrative Reference Architecture:** A holistic reuse approach should include a reference architecture providing patterns and guidance to existing Web Engineering approaches on how to integrate their heterogeneous applications and stores.
- **R-R-05 – Coordination:** Besides supporting planned reuse, the approach should provide coordinative support reducing redundant efforts in development for reuse.

3 State of the Art

Since Web Engineering constitutes a comparatively young research discipline, a multitude of approaches has been proposed and, according to new requirements emerging from the rapid evolution of the World Wide Web and its applications, continuously been extended. Consequently, established Web Engineering methodologies have been extended with respect to the new generation of workflow-based Web applications as well as specialized approaches for particular aspects have been introduced. Likewise, commercial solutions, particularly in the fields of business process management, have emerged. Altogether, these scientific and commercial approaches represent the current state of the art for this thesis.

In this chapter, the current state of the art will be evaluated based on the requirements catalog presented in the previous chapter. As an adequate solution could not only consist in a single comprehensive methodology, but also be achieved by a combination of aspect-specific approaches, the analysis is again divided into the three dimensions Workflow, Dialog and Reuse. For each dimension, established scientific and commercial approaches are representatively evaluated, highlighting their strengths and weaknesses for the concerned problem domain. The chapter concludes with an integrated overview of the evaluation results underlining the necessity for novel models, systems, and methodologies for the construction of workflow-based Web applications.

3.1 Dimension Workflow

3.1.1 Object-Oriented Hypermedia Design Method (OOHDM)

The Object-Oriented Hypermedia Design Method (OOHDM) is one of the first model-based approaches to design and develop Web applications (Schwabe, Rossi and Barbosa 1996). OOHDM proposes a five-step development process consisting of requirements gathering, conceptual design, navigational design, abstract interface design and implementation. Each of these steps focuses on a particular design

concern and is supported by a specific modeling approach. As indicated by the methodology's name, the modeling notations are predominantly derived from object-oriented modeling techniques like the Unified Modeling Language (UML). The approach is partly supported by a dedicated technical development framework named HyperDE (Nunes and Schwabe 2006).

In (Rossi, Schmid and Lyardet 2003; Schmid and Rossi 2004), extensions to the OOHDM approach towards workflow-aspects was presented. The authors point out the importance of conceptually differentiating between navigation and stateful business process as well as examining semantics regarding the transitions between those. However, the aspect of business processes in Web applications is mainly considered from a navigational perspective, albeit enriched by state- and context-related semantics. Consequently, important characteristics of workflow-based Web applications like various roles participating in a business process or the integration of backend systems are not considered.

With respect to a suitable modeling-support for processes, activities and control flow, the authors propose to integrate activities in the conceptual and navigational schema models of OOHDM as well as to capture control flow in separate UML activity diagrams. Thus, activities can be reused in various process contexts. However, an abstraction regarding particular generic activity types is not taken into account. Furthermore, the important aspect of model consistency resulting from this threefold separation is not discussed.

The user Interface design is not specifically covered for workflow-based Web applications. Thus, it can be assumed that the general approach of OOHDM for user interface modeling, which is based on Abstract Data Views (Cowan and Lucena 1995), should be applied. Hence, device independency regarding the Web-based user interface is at least considered on a conceptual level by separating the abstract interface from its implementation via one or multiple concrete interface widgets. The interaction via external task-specific applications or portals though is not addressed.

The transition from models to technical implementation is not covered by the authors; they solely refer to a particular state-machine framework, but leave the actually very complex implementation of the model semantics as well as architectural design decisions unconsidered. Thus, according to this approach, the implementation has to be performed manually, which in turn hinders agility and evolution.

The authors clearly state developers as their target audience. The modeling notation adheres mainly to UML, whereby class diagrams serve for specifying OOHDM conceptual and navigational schema models as well as constrained activity diagrams for specifying the control flow. For stakeholders collaborating in the specification of the business process and the realization of its tasks, the distribution of processes and activities to multiple models may be less intuitive. Guidance on how to derive the required OOHDM models from existing business process diagrams is missing. Beyond that, due to the missing abstraction regarding various generic activity types, the concrete design of each process activity requires considerable specification and implementation effort.

3.1.2 Web Modeling Language (WebML)

The Web Modeling Language (WebML) is a model-driven Web Engineering methodology which focuses on so-called data-intensive Web applications (Ceri, Fraternali and Bongio 2000). Since its inception, WebML has been very actively promoted and is today one of the most prominent Web Engineering methodologies. WebML is supplemented by a comprehensive development framework named WebRatio 5, which is fully integrated into the Eclipse framework (Acerbis, Bongio, Brambilla et al. 2007).

To date, several extensions to WebML supporting lightweight Web-enabled workflows have been proposed (Brambilla, Ceri, Comai et al. 2003; Brambilla 2006; Brambilla, Ceri, Fraternali et al. 2006; Brambilla, Comai, Fraternali et al. 2008). Thus, the WebML modeling framework is extended in several ways. First, a business process modeling dimension is introduced in the methodology. Second, the existing data model is augmented with process tracking-relevant objects required for logging and constraints evaluation purposes. Third, the existing hypertext model is extended by constructs for specifying the business activity boundaries and workflow-dependent navigation links. Beyond that, a workflow-driven hypertext generator for transforming business process models into WebML hypertext skeletons was developed.

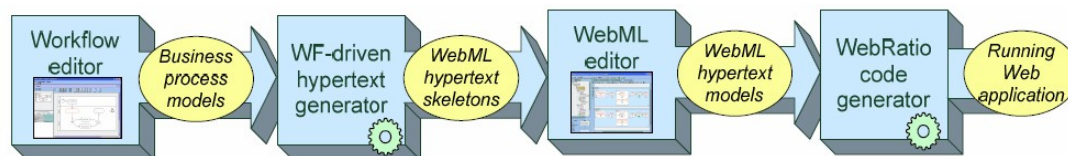


Figure 3-1: Overview of Steps, Tools and Results of the WebML Methodology for Lightweight Web-enabled Workflows. Taken from: (Brambilla 2006)

The resulting WebML development process for lightweight Web-enabled workflows is outlined in Figure 3-1. A proprietary Eclipse-based visual business process modeling tool serves for modeling the business process based on the Business Process Modeling Notation (BPMN). The resulting business process model is translated into skeletons of the WebML hypertext model, data model and workflow metadata, providing support for major control flow patterns, i.e. sequence, AND-, OR-, XOR-splits and joins as well as basic loops. Based on the skeletons, members of the development teams have to perform a comprehensive and detailed design of the so far empty activity structures as well as specify data queries related to workflow constraints. Thus, the user interface and the business logic have to be manually designed using the general WebML modeling methodology for data-intensive Web applications and the WebRatio tool support. With respect to short development lifecycles, considerable modeling effort remains before a first version of a running application is available.

The architectural model of the WebRatio runtime environment uses a component-based approach for realizing so-called “WebML units” which represent common data-oriented operations and which form the core elements for modeling pages.

Even though there is no conceptualization or componentization of typical generic workflow activity types, such activity types could be realized by a combination of existing units and newly-developed units. The requirement of device-independency is only covered at implementation level by providing different page templates for different markup languages. The page-orientation at design level aggravates this problem.

Beyond that, the WebML approach weaves the workflow enactment logic into its various models. The process control flow, for example, is translated into navigation and associated constraints. Thus, there is no distinct separation of workflow enactment from other aspects of the Web application, both on the model layer and the implementation. Although WebML addresses the publishing of modeled operation chains via Web service interfaces, this intermixture hinders the exposure of workflow enactment-related services for federative workflow participation scenarios. The realization of federative process scenarios though is not completely impossible with WebML; however, a significant amount of implementation-oriented modeling and development work is required which could be avoided by inherently addressing this requirement on an architectural level.

Furthermore, this missing separation aggravates the adoption of changes which in turn also impedes an iterative development approach. There is no reverse transformation from the refined WebML data and hypertext models back to the business process model. Once the forward transformation has been performed, changes to the process structure within the hypertext model are not propagated back. Moreover, the pure process structure embodied in the hypertext model is hardly visible as it is modeled as some type of enriched navigation and mixed up with page and navigation design.

Since the first version of WebML, a lot of appealing and relevant extensions have been introduced, resulting in a very comprehensive and expressive modeling approach for a developer audience. However, as a downside of the resulting small set of very comprehensive models, the approach became increasingly heavy-weight with lots of details and special symbols. Furthermore, except the business process modeling via BPMN, WebML defines completely proprietary notations. Regarding business process modeling and workflow implementation aspects, no technical standards are employed.

3.1.3 UML-based Web Engineering (UWE)

The UML-based Web Engineering (UWE) methodology (Hennicker and Koch 2000) is a model-driven Web Engineering approach which particularly stands out due to its strong foundation on the Unified Modeling Language (UML) as well as its extensive incorporation of related standards. UWE proposes at least one dedicated UML diagram type according to various development stages, Web application concerns and structural versus behavioral views. The UWE methodology aims at a continuous model-driven development approach based on the Object Management Group's

Model-Driven Architecture approach (Mukerji and Miller 2003). The comprehensive and strongly formalized metamodel underlying the UWE approach is defined as an extension to the UML metamodel (Koch and Kraus 2003) and can be represented by an UML profile. Consequently, the UWE development framework ArgoUWE (Knapp, Koch, Moser et al. 2003) is based on a general open-source UML modeling tool.

The UWE methodology is continuously evolving and proposed various extensions concerning the modeling of business processes in Web applications (Koch, Kraus, Cachero et al. 2003; Knapp, Koch, Zhang et al. 2004; Kraus, Knapp and Koch 2007). Thereby, the UWE approach is extended by additional models, development steps, model transformations and a technical workflow interpreter component. To date, however, full support is only provided for single-person, non-persistent processes, which are often referred to as “page flows”. Hence, concepts like workflow persistence or task assignment are not covered yet.

The conceptual design regarding process aspects starts with the specification of the business process in form of an UML activity diagram. Then, the process is integrated as an entity in the UWE navigation model and connected with other navigation classes to model entry and exit points from and to other navigational entities. Thereby, the process model itself and the navigation model remain separated. Afterwards, a so-called “structural process model” containing process-related data objects in terms of an UML class diagram has to be developed. Furthermore, the conceptual process model from the beginning has to be translated and enriched into a so-called “process flow model”. For each user interaction step in the process, a separate presentation model has to be developed. The method supports only structural presentation aspects whereas more detailed layout and design aspects as well as special characteristics of mobile devices remain unconsidered. Throughout the complete modeling process, the designer is supported by automatically generated model skeletons as well as consistency checks.

The model-based generation of a platform-specific Web application is based both on a transformational and an interpretational approach. Regarding the content, navigation and presentation models, model-to-code transformations are applied, whereas the platform-independent process flow diagram is translated into a configuration document which in turn is executed by a proprietary interpreter. This interpreter offers a limited coverage of possible control flow constructs only. The fact that the current design and integration of this interpreter component in the UWE runtime platform allows at most one active process per user session underlines the focus on “page flows” instead of real workflows.

In a model-driven development approach like UWE, architectural decisions can either be integrated in the transformations or explicitly modeled in dedicated architecture models. To date, UWE mainly follows the former approach but mentions the possibility of supplementing the current transformation concept by architecture models. Presently, neither component-based nor federation- or service-related concepts have been incorporated yet. Regarding the latter, neither the integration of external Web services nor the exposure of an application’s data or business logic via Web service-enabled endpoints is covered by UWE. Due to this as

well as considering the “page flow”-oriented view of business processes in UWE, no additional access channels beyond the Web browser are envisioned.

UWE’s long-term vision of achieving a purely model-based approach and thus solving consistency issues between platform-specific models and code has not yet been realized. Due to the existing model-code gap and missing reverse-engineering concepts, manual changes on the implementation-level get lost when the code is regenerated due to changes on the model-level. This represents a major roadblock to the efficient adoption of changes and thus to iterative development approaches. Beyond that, UWE still requires a lot of modeling effort besides the business process model and is not capable to generate a running prototype without manual coding work.

Considering factors like the required modeling effort, the partly unintuitive modeling guidelines imposed by the strong adherence to UML, or the necessity of manual implementation work, UWE represents a methodology which is primarily tailored to a developer audience.

3.1.4 IBM WebSphere Suite

The IBM WebSphere suite offers various applications and server products for modeling, implementing and monitoring end-to-end business processes (Wahli, Avula, Macleod et al. 2007). Figure 3-2 shows the WebSphere suite’s various products tailored to different aspects of business process management, whereby only the first three correspond to this thesis’ problem scope. According to a recent study of the Butler Group, these IBM solutions belong to the top three suites for SOA-based business process management (Hailstone, Illsley, Jones et al. 2007). Thus, they constitute representative candidates for the evaluation of the state of the art from a commercial perspective.

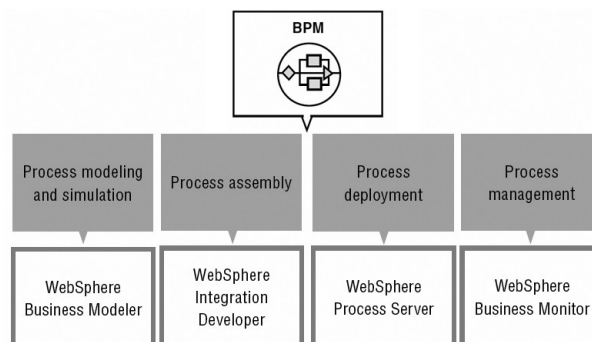


Figure 3-2: Overview of IBM Business Process Management Products.
Taken from: (IBM Corp. 2006)

Within the IBM WebSphere product family, the WebSphere Process Server constitutes the runtime platform for workflows, thereby providing process orchestration, business rules execution, human task management and process state

management. Thus, it naturally provides support for long-running workflows comprising diverse roles and systems.

The WebSphere Integration Developer (WID), an Eclipse-based integrated development environment, is used for the model-based development of workflows. Using the WID, developers can visually design the workflow's control flow as well as assign predefined or custom developed service components to each process activity. Besides components for realizing system-oriented tasks, e.g. Web Service invocations, a dedicated component can be used to indicate human tasks. The construction of rich Web-based user interfaces supporting the completion of such human tasks, however, is only poorly supported. On the one hand, the Web Sphere Process Server is capable of generating a very simple task interface which shows the input message for the current task and allows for entering a corresponding output message. Such an interface can only serve developers for testing purposes. On the other hand, custom developed Java Server Pages (JSPs) can be manually implemented and attached to human tasks. To this end, WID provides developers with the automatically generated JSPs which can then be refined. The realization of device-independently accessible user interfaces is likewise left to the developer. This comparatively low methodological and technical support for the efficient construction of continuous and rich Web-based user interfaces is typical for business process management suites. To date, such suites focus primarily on workflow modeling, integration, enactment and monitoring and consider the construction and provision of Web-based user interfaces solely as add-ons.

The IBM WebSphere Process Server offers comprehensive interfaces for integration and federation scenarios. Thereby, custom client applications can access and interact with the workflow engine. Furthermore, the Web Sphere Process Server provides wide-ranging support for workflows comprising multiple systems via diverse ways of integration as well as for distributed participation scenarios. The so-called "Service Component Architecture (SCA)" approach forms the foundation for constructing composite workflows by wiring process flow with service components. Regarding human tasks, only a single, very abstract "human task" service component is available. A more detailed specification with respect to various activity types would be desirable.

Figure 3-3 depicts an overview of the suggested development process based on the IBM software development platform and its various products. The process starts with creating a business process model using the WebSphere Business Process Modeler. Then, using the WebSphere Integration Developer, the business process' technical realization in form of a workflow is modeled. Simultaneously, the development of new components as well as the Web-based user interface is performed based on the Rational Software Architect platform. Finally, both the workflow model and the developed components are deployed and executed on the WebSphere Process Server. While, according to this process model, the modeling of workflows turns out to be rather efficiently and agile, the construction of workflow-based Web applications still requires a considerable amount of manual implementation effort. For most real-world Web applications, none of the generated user interface components can be actually used. This in turn presents a major roadblock to the requirement of short development timeframes.

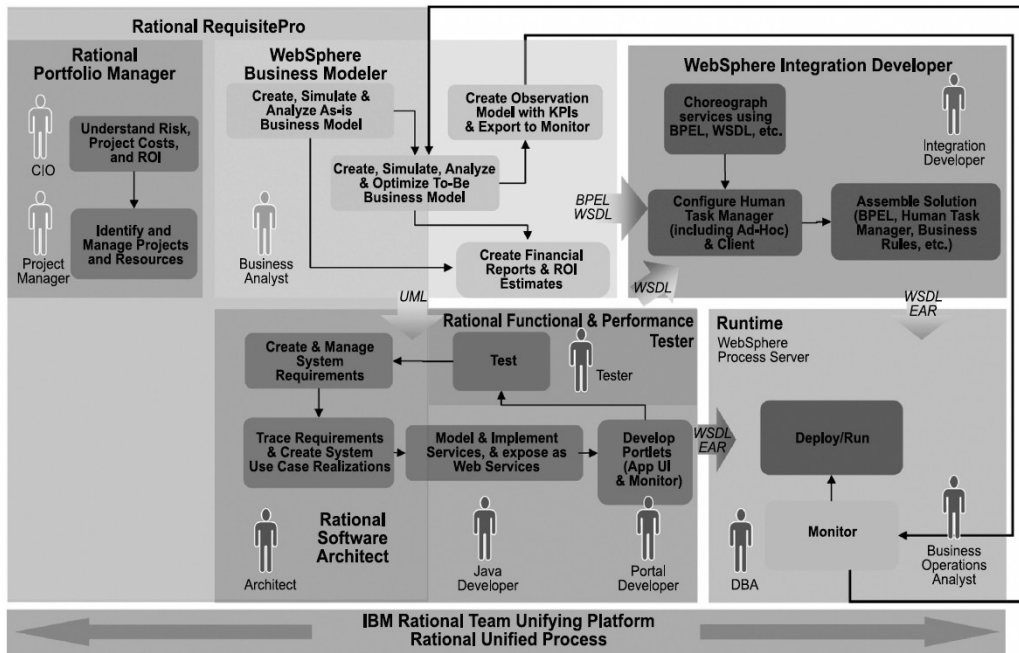


Figure 3-3: Overview of the Development Process based on the IBM Platform.
 Taken from: (Brown, Johnston, Larsen et al. 2005)

While the intention of providing various tools for different audiences seems reasonable, the multitude of tool-specific models aggravates model exchange and consistency, which is crucial for an iterative, evolution-oriented development process. While there is a forward propagation of changes from the Business Modeler to the Integration Developer (Fasbinder 2007b), the backward direction is only supported by indicating changes within the Business Modeler. Their adoption on the business process model has to be performed manually (Fasbinder 2007c). Regarding the transition from the Business Modeler to the Rational Software Architect (RSA), changes to the initial process model cannot be propagated to the RSA platform. To this end, one shared model, at least between the Business Modeler and the Integration Developer, would be advantageous.

IBM positions the WebSphere Business Modeler as a business process modeling tool primarily tailored to a role termed “business analyst” which resides within the project team. The employed graphical modeling notation is similar to the Business Process Modeling Notation (BPMN). Although, multi-notational modeling is not supported, plugins for importing process diagrams from Microsoft PowerPoint, ARIS or Microsoft Visio are available. This highlights that also industry has recognized the need for multi-notational modeling and is undertaking first steps towards this direction. However, there is neither the possibility to work on a shared process model using various notations nor concepts for assuring consistency between various notations once a model has been imported.

The WebSphere Business Modeler supports the design of business processes using multiple hierarchical layers and two different levels of detail. While this contributes to a rather lightweight modeling of business processes, there is no model-based support at all for Web-related aspects, particularly regarding the user interface.

IBM pursues, unlike most of its competitors, the integration of and adherence to standards very actively and continuously. For example, the WebSphere products adopt standards in the field of business process management like the Business Process Modeling Notation (BPMN), the XML Process Definition Language (XPDL) or the Business Process Execution Language (BPEL) (Fasbinder 2007a). Furthermore, the IBM Rational Software Architect, which is used for the development of new services and components, relies strongly on UML and related standards, e.g. the XML Metadata Interchange (XMI) format.

3.2 Dimension Dialog

3.2.1 Object-Oriented Hypermedia Design Method (OOHDM)

The Object-Oriented Hypermedia Design Method (OOHDM) originally employs Abstract Data View (ADV) models for the specification of dialogs and their dynamic behavior (Cowan and Lucena 1995). In the context of their Semantic Web-oriented version of OOHDM named “Semantic Hypermedia Design Method (SHDM)”, a revised approach for presentation design was presented (Moura and Schwabe 2004). This approach introduces a so-called “Abstract Widget Ontology” which defines abstract interface widgets representing various types of functionalities a user interface element can embody (cf. Figure 3-4).

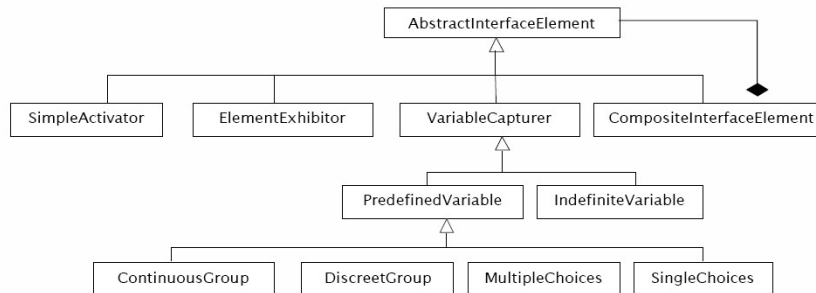


Figure 3-4: Abstract Widget Ontology of OOHDM/SHDM.
Taken from: (Moura and Schwabe 2004)

Based on this vocabulary, a software designer specifies the abstract interface design. At first, only functional aspects are regarded, whereas the concrete user interface is specified in a second step. Therefore, based on a “Concrete Widget Ontology” for a particular runtime environment, each abstract interface widget has to be mapped on a concrete interface widget, e.g. text box, radio buttons, check boxes or buttons.

The presented approach covers only the abstract and concrete interface design regarding structural aspects of a dialog. The concrete implementation concerning design and usability aspects for a specific platform, however, is not addressed. Device-independent access can be supported by providing various concrete widget ontologies for various target platforms. Web service support is neither considered by

OOHDM in general nor with respect to the construction of dialogs in particular. The missing model continuity aggravates short iteration cycles as well as the efficient adoption of changes. Beyond that, the methodology is fully developer-oriented; the first phase of abstract interface design already requires a software designer with an understanding of the functional logic associated with the employed terms. The concrete interface widget is translated into a raw HTML-based serialization. Thus, the potentials of advanced markup formats are not leveraged.

Recently, the OOHDM group proposed an interesting approach towards enriching hypermedia application interfaces by animating navigational transitions and thereby emphasizing semantically important information (Fialho and Schwabe 2007). Although this idea is not particularly tailored to the characteristics of advanced Web-based dialogs, it represents an interesting direction towards improved Web usability.

3.2.2 Web Modeling Language (WebML)

The Web Modeling Language (WebML) recently presented dedicated extensions towards Rich Internet Applications (RIA), augmenting the existing WebML modeling notation by means for modeling RIA-specific data, business logic and particularly presentation aspects (Bozzon, Comai, Fraternali et al. 2006; Preciado, Linaje, Comai et al. 2007). Similarly to OOHDM, the dialog design is differentiated into abstract and concrete interface design. The abstract interface specification comprises device- and platform-independent aspects like data mapping, abstract media elements, e.g. text, image or video, as well as logical views for grouping simultaneously visible elements. The concrete interface design focuses on specific user interfaces for particular devices concerning spatial, temporal and interaction concerns. Based on the concrete interface specification, the final interface for a specific platform can be generated. Therefore, predefined mappings from concrete interface elements to platform-specific components are evaluated. Presently, the model serialization into HTML supported by various JavaScript-based AJAX frameworks is described, whereby more powerful markup languages like XAML are mentioned as future candidates.

While this separation enables the manual modeling of device-specific interfaces, automated adaptations according to the requesting client device and their instrumentation are not addressed. Beyond that, it is questionable if the specification of simultaneously visible elements already in the abstract interface design is reasonable.

To date, as WebML is a methodology addressing particularly data-intensive Web applications, the focus of the presented RIA-oriented extensions consequently lies on common requirements of data-intensive RIAs, e.g. like dynamic filtering or ordering of data in response to a user's input. However, due to the proposed combination of WebML and the Rich User Experience (RUX) Method, some usability patterns of advanced Web-based dialogs, e.g. the separation into multiple views, can be efficiently realized (Linaje, Preciado and Sánchez-Figueroa 2007). A particular

consideration of dialog-specific characteristics and challenges as well as design time assistance regarding related usability aspects has not been covered so far.

While WebML provides modeling techniques regarding the communication with Web services, the construction of rich Web service-enabled dialogs starting from data schemas or Web service specifications still requires considerable manual modeling effort. The presented WebML development process for RIAs can almost completely be performed on a model-basis. However, the modeling process comprises multiple tools, is rather time-consuming and does not provide support for rapid prototyping. Beyond that, some dialogs might require changes or extensions to the generated markup or JavaScript code. To this end, the approach presently does not provide concepts for assuring backward consistency between the platform-specific code and the platform-independent models. This in turn aggravates an iterative, evolution-oriented development approach.

The RUX tool supporting the presentation modeling is tailored to a Web designer or developer audience. The modeling notation both for the abstract interface design and the concrete interface design seem rather unintuitive and confusing to stakeholders having a dialog's final structure, behavior and appearance in mind. An increased separation of concerns supplemented by corresponding, more simple and stakeholder-oriented views would be advantageous.

3.2.3 UML-based Web Engineering (UWE)

The UML-based Web Engineering methodology (UWE) allows for the model-driven construction of Web-based dialogs using dedicated UML stereotypes (Hennicker and Koch 2001). Regarding the modeling of a dialog's dynamic behavior, first ideas based on UML state charts were proposed (Baumeister, Koch and Mandel 1999), but seem not to have been pursued in more detail so far. Further usability aspects are not explicitly addressed and the efficient realization of usability-improving patterns is not adequately addressed either.

Due to the implementation-independent modeling approach, dialogs for diverse platforms could be modeled and generated via dedicated model transformations; how exactly this could be realized remains open though. Moreover, runtime adaptations of dialog models according to requesting devices as well as their model-based instrumentation have not been addressed yet as well. UWE presentation models are serialized into Java Server Pages or HTML; advanced markup languages have not been considered yet. Furthermore, as mentioned in the Workflow dimension-related analysis of UWE, the approach does not provide support for modeling Web service communications. Thus, the realization of Web service-enabled dialogs requires comprehensive manual implementation effort.

Concerning the requirement of agility, i.e. short evolution cycles and flexible adoption of changes, UWE presentation models cannot be directly transformed into running applications as there is still manual source code development required.

Besides model consistency issues, this also hinders stakeholders without development skills to perform lightweight modifications on existing dialogs.

The UWE notation for structural presentation modeling appears rather intuitive; however, the resulting multitude of distinct UML state-chart models for each dynamic behavior might be rather confusing. Beyond that, UWE omits modeling support concerning the concrete and final user interface design.

Recently, a combination of the UWE and RUX methodologies targeting the construction of Rich Internet Applications (RIA) was proposed (Preciado, Linaje, Morales-Chaparro et al. 2008). Thereby, similar to the combined approach of WebML and RUX analyzed above, at least some usability patterns can be realized more efficiently. Beyond that, the aspect of device independency becomes more manifest, integrating advanced markup languages more tangible and manual development effort is reduced.

3.2.4 IBM Lotus Forms Designer

IBM Lotus Forms Designer 3.5, released at the end of 2008, is a visual design tool for the construction of electronic (Web) forms (IBM Corp. 2008a). As Forrester Research rated IBM as a leading vendor in the e-Forms market (Murphy 2006), this product denotes a representative candidate for evaluating the commercial state of the art with respect to the development of Web-based dialogs.

The IBM Lotus Forms Designer stores forms based on the Extensible Forms Description Language (XFDL) format (Boyer, Bray and Gordon 1998). Thus, for accessing them from a Web browser, either a dedicated browser plugin or a supplemental product named “Lotus Forms Server 3.5 – Webform Server”, acting as a translator from XFDL to HTML and Java Script, is required (IBM Corp. 2008b).

Figure 3-5 depicts a screenshot of the Lotus Forms Designer application which is based on the Eclipse platform. The user interface is similar to today’s integrated development environments and comprises a visual design pane, a toolbox containing dialog components, a comprehensive property editor as well as further windows providing detailed information concerning various aspects. The user interface is available in a standard view and an advanced view, whereas the former does not cover all tasks required for the construction of usability-oriented Web-based dialogs.

Concerning the important factor of dialog usability and related user interface patterns, Lotus Forms Designer offers means for their rather efficient implementation. While some of these features can be realized on a drag-and-drop basis, others, e.g. the selection-dependent visibility of form areas, require the manual creation of formulas. Explicit guidance towards the consideration and incorporation of usability aspects is missing though. An important usability pattern addressing the problem of cognitive overload advises to show only relevant fields to a user by incorporating selection-dependent visibility of form areas. In complex business forms, a multitude of such selection-dependent inputs is usually required. While Lotus Forms Designer allows developers to implement them based on

formulas (cf. Figure 3-5), these dependencies are not perceivable in the visual design pane. Thus, the actual dialog behavior is not visible to both developers and stakeholders, which in turn hinders the collaborative design process. The same perception issues arise in the distribution of a dialog on multiple pages, whereby the dynamic transitions between these pages lack a visual representation. Such problems can probably be attributed to the inherent paper-oriented design approach pursued by the great majority of commercial form design applications.

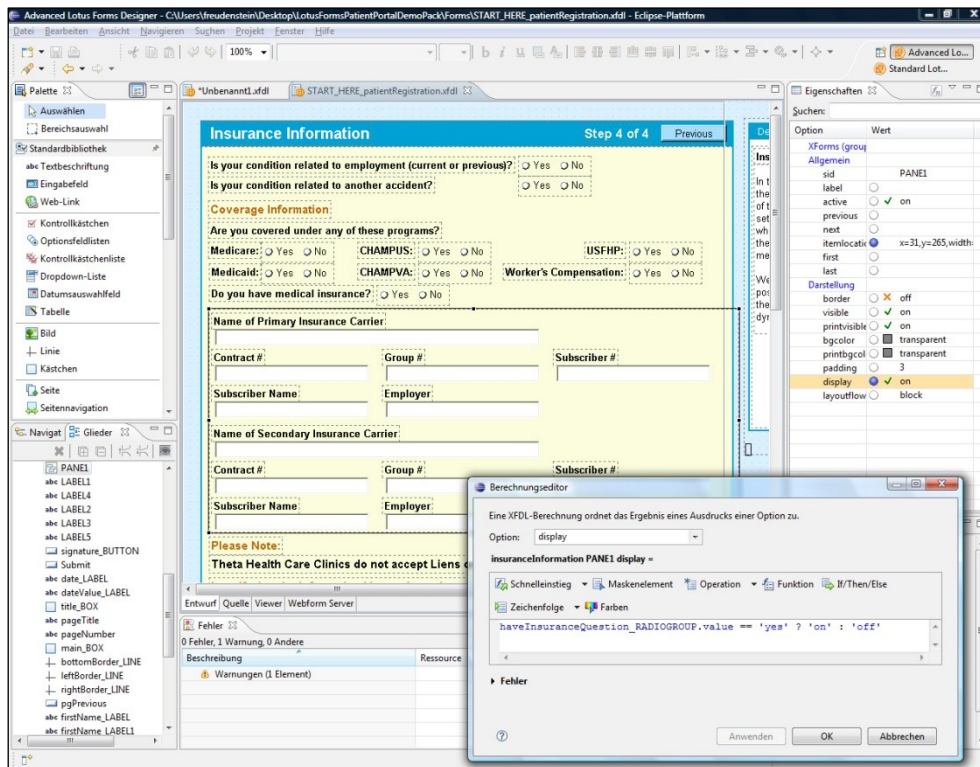


Figure 3-5: Screenshot of IBM Lotus Forms Designer

Regarding device-independent access and use of forms, the IBM Lotus Forms suite does not provide native support. However, by extending the Lotus Forms Webforms Server by a dedicated, custom developed component, runtime adoptions according to various requesting devices could be achieved. The partial foundation of forms on the device-independent XForms standard provides at least a starting point for that.

Lotus Forms Designer allows the specification of Web service invocations. However, this requires a rather comprehensive procedure, a sound technical understanding and even XForms programming skills. A more abstract, ideally model-based support for integrating Web service invocations would be preferable. Furthermore, Lotus Forms Designer is capable of parsing XML Schema and WSDL documents and supports form composition via drag-and-drop operations on elements from the derived XML Schema tree.

The IBM Lotus Forms Designer allows for a rather agile and evolution-oriented development process. Potentials for improvement lie in the initial generation of dialogs. When creating a new form, only PDF documents can be supplied for the generation of a form. Thus, in most cases designers will start with an empty page. To

this end, the direct generation of form prototypes based on a given XML Schema or a combination of a WSDL document and an operation identifier, would be desirable.

With respect to the requirement of strong stakeholder involvement, Lotus Forms designer rather addresses a developer audience. On the one hand, the what-you-see-is-what-you-get-based design mode is rather beneficial to stakeholder understanding and communication in terms of design and layout aspects. On the other hand, however, the lack of a visual illustration of a dialog's dynamic behavior mentioned before presents an obstacle. Beyond that, a process model guiding stakeholders in how to analyze, design and implement a rich Web-based dialog, particularly regarding early stages, is missing.

Recently, IBM released a dedicated Web-based form design tool for non-technical users named "Lotus Forms Turbo" (IBM Corp. 2008c). Lotus Forms Turbo supports both the design and deployment of Web forms as well as provides analysis features for evaluating submission results. The form's underlying data model is implicitly defined by the created form elements and cannot be modified; neither can a given data model be used as a starting point. While Lotus Forms Turbo definitely represents a considerable step towards stakeholder empowerment, the tool presently supports only very basic forms which are not sufficient for the considered problem scope.

As highlighted in Section 3.1.4, IBM generally places emphasis on the incorporation of open standards where possible. Consequently, also the Lotus Forms suite adheres to various standards including XForms and XFDL. While XFDL is used for declaring presentational aspects, XForms is employed for specifying a dialog's data model, validations, and form control types. Although this separation generally seems reasonable, it was conducted in an inapt way at some points. As XFDL has not been actively pursued by the W3C since 1998, employing means from the XForms standard where possible and only transferring actually uncovered aspects to a presentation-oriented language would have been a preferred way. The XForms export function from Lotus Forms Designer underlines this problem, as too much information that is actually covered by the XForms standard gets lost.

3.3 Dimension Reuse

3.3.1 Scientific Reuse Approaches for the Web Engineering Domain

In the Web Engineering discipline, reuse-related research primarily focuses on the adequacy of models and software components for reuse. Due to the overall poor coverage of Web Engineering-specific reuse aspects, the analysis of the state of the art regarding this dimension is not performed separately for particular methodologies. A rather holistic evaluation comprising various relevant approaches and reaching also beyond methodology-specific concepts and solutions better satisfies the comprehensive nature of reuse.

As an example for methodology-specific reuse concepts, the OOHDM research group introduces the concept of “OOHDM Frames”, i.e. Web design frameworks for specifying common design schemas and their variation points, thus fostering reuse on design level (Schwabe, Esmeraldo, Rossi et al. 2001). A similar idea called ‘WebML skeletons’ is presented in (Ceri, Fraternali and Matera 2001). Such skeletons specify abstract and simplified versions of recurring structural and hypertext schemas for being instantiated and reused. In conclusion, these approaches provide interesting insights on how reuse on a model level could be improved by identifying and modeling recurring abstractions and reusing them by instantiation for a particular application. While the similar goals of both approaches awake the desire for a unifying, cross-methodological reuse approach for models, this has not been addressed yet.

With respect to reusing Web components and their code, the WebComposition approach presents its dedicated WebComposition Repository in (Gaedke and Rehse 2000). It aims at facilitating the storage and retrieval of components, thereby allowing for incorporating various metadata representation methods as postulated by Frakes and Pole in (Frakes and Pole 1994). Efficiently finding reusable components and code is a key factor, not only for Component-based Web Engineering, but also for other Web Engineering methodologies. However, a generalization of the WebComposition reuse approach establishing a basis for reuse both on model and component level has not been pursued.

Similarly, the realization of reuse support on a technical level has only been taken into account in the context of method-specific development environments such as WebML’s WebRatio (Acerbis, Bongio, Brambilla et al. 2007). Some Web Engineering frameworks integrate generic version control systems, thus allowing for storing diverse artifact types across methodologies. However, this only covers pure data storage and disregards metadata aspects.

In (De Medeiros, Schwabe and Feijo 2005), the authors present the “Kuaba Ontology” - an ontology-based approach for reusing Design Rationales, i.e. the reasons and justifications for design decision, and associated artifacts. Although the approach addresses a different problem domain, the idea of establishing a unifying, methodology-independent foundation in form of an ontology seems to be a promising approach for the reuse domain as well.

Beyond that, the Web Engineering community presently strives for realizing the hitherto untapped potential of interoperability and model interchange across today’s Web Engineering methodologies. In (Selmi, Kraiem and Ghezala 2005), a generic framework defining a common denominator and enabling the comparability of these methods is proposed. Such research forms a vital initial step towards achieving interoperability and thus also presents an important input for a cross-methodological reuse approach. Beyond that, consolidation efforts like the Model-Driven Web Engineering initiative MDWEnet (Vallecillo, Koch, Cachero et al. 2007) strive for achieving practical interoperability between common model-driven Web Engineering methodologies. Thus, the potential of a unifying reuse approach becomes even more obvious, as it is not only applicable across today’s Web Engineering methods, but also enables real cross-methodological reuse. As these

consolidation efforts are still in an early stage, reuse-related research in this direction has not been presented yet.

Similarly, reuse-related stakeholder-orientation and coordinative support concerning development for reuse have not been addressed yet. While the former can be attributed to the poor consideration of stakeholder aspects already in the development process, the latter should be aligned with concepts for planned reuse which therefore have to be addressed first. The only coordinative support offered by advanced development environments so far consists in the use of centralized artifact stores in addition to the local file system.

3.3.2 Commercial Solutions

Internal software reuse programs have been ongoing in IBM for over twenty years (Yglesias 1998). Consequently, IBM offers a mature product named “Rational Asset Manager 7.1” supporting reuse in the software engineering domain (IBM Corp. 2008d). The product is part of IBM’s federated metadata management strategy (Schmidt and Larsen 2007). Gartner Research considers IBM’s strategy to be “visionary” and denotes IBM as an industry leader in this area (Blechar 2007). Thus, the IBM strategy in general and the Rational Asset Manager in particular constitute representative candidates for evaluating the commercial state of the art for the dimension Reuse.

Over the last years, IBM has introduced three tool- and community-specific repositories: The “Tivoli Change and Configuration Management Database” addresses system-related assets. Assets concerning data architecture, data warehousing or enterprise information integration are covered by the “WebSphere Metadata Server”. SOA-related assets are addressed by the “WebSphere Service Registry and Repository”. With Rational Asset Manager, IBM introduces a fourth, more generic repository for all further kinds of assets. The product offers both a Web-based and an Eclipse-based user interface. Rational Asset Manager employs OMG’s Reusable Asset Specification standard (Object Management Group 2005a) as metadata format and allows for configuring custom artifact types and corresponding metadata schemas based on the standard’s extension mechanisms. Thus, specific metadata schemas and categorization taxonomies tailored to the Web Engineering domain and its artifacts could be realized; however, in order to realize sophisticated cross-methodological reuse support, more powerful semantic description facilities are required.

IBM states the goal of supporting heterogeneous and globally dispersed communities and their specific stores and tools. This can be valued as a first step towards stakeholder-orientation; however, stakeholder characteristics, e.g. skills or knowledge, have not been addressed as a context parameter for storing and finding artifacts yet.

While IBM pursues the long-term vision of a fully federated metadata strategy across diverse registries and repositories, the current focus lies on the pragmatic

integration of the Rational Asset Manager and the WebSphere Service Registry and Repository. Federated searches across repositories, presently only spanning these two products, are a core feature of Rational Asset Manager. This is achieved by synchronizing asset metadata, which can lead to problems regarding coordination and integrity though. Thus, federated searches are actually executed on a single, combined metadata store and are limited to name-, keyword-, tag- and category-based queries. More powerful semantic search capabilities are not provided. However, filtering and browsing facilities along asset relationships are offered.

The need for integrating further, already existing repositories was recognized by IBM and included in its long-term strategy for federated metadata management. So far, however, this is not covered yet. Coordinative support for aligning redundant development efforts is not explicitly addressed. The possibility of integrating multiple WebSphere repositories and registries, thus covering not only production-oriented instances, as well as their transparent background synchronization into Rational Asset Manager present first steps towards this direction.

3.4 Evaluation Results and Conclusion

The results of the evaluation of the state of the art conducted in this chapter are summarized in Table 3-1 for the dimensions Workflow and Dialog and in Table 3-2 for the dimension Reuse. The leading acronyms in each row stand for the requirements elaborated in the previous chapter and recapitulated in Table 3-4. A legend explaining the employed rating symbols is provided by Table 3-3.

Based on the performed evaluation, the following major problem areas of current scientific and commercial approaches can be pointed out:

- **No holistic consideration of workflow and user interface aspects:** Today's established Web Engineering approaches still consider workflows as some kind of advanced navigation, thus predominantly covering at most two out of six relevant workflow perspectives (Weske, Vossen and Puhmann 2005). Consequently, with respect to workflow modeling and execution, they neglect important characteristics and requirements of long-running workflows comprising multiple roles and systems. Furthermore, the dedicated support for constructing workflow-based Web applications does in the most cases not reach beyond basic Web process modeling and concepts for weaving workflow models into existing, usually navigation- and data-related, models. Hence, important architectural concerns, like support for multimodal participation and federative scenarios as well as leveraging the potentials of component orientation, remain unconsidered. On the other hand, commercial solutions provide comprehensive support for workflow specification and execution but disregard Web-related requirements. Therefore, an adequate Web Engineering approach should pursue a holistic perception of both Web- and workflow-related requirements. As mature commercial workflow platforms exist, such an approach should allow for their

smooth integration, thereby delegating the technical workflow execution and rather focusing on their adequate instrumentation.

- **Insufficient support for advanced Web-based dialogs:** Although present Web Engineering approaches naturally address interaction design, their dialog design methodologies are limited to rather basic dialogs. Thus, they neglect the specific requirements of advanced, highly interactive dialogs, particularly regarding usability-oriented design and efficient implementation of usability-related features. Similarly, only WebML provides modeling support for the important aspect of Web service integration. Even though powerful user interface-related markup languages like XForms have emerged over the last years, today's Web Engineering methodologies still ignore their benefits and rather transform dialog models into an HTML-based markup representation. In contrast to model-driven Web Engineering methodologies, commercial approaches like IBM's Lotus Forms are characterized by what-you-see-is-what-you-get-oriented dialog editors. As such, they enable a predominantly visual, paper-like form design supplemented by technical-oriented property editors. However, due to their constraining focus on direct editing of the final dialog, they lack more abstract modeling support as well as a systematic engineering process. Besides hindering an overall design of a dialog's dynamic behavior, this missing abstraction also aggravates device-independency.
- **Constricted, proprietary reuse approaches:** Both scientific Web Engineering approaches and commercial solutions provide reuse support only within the boundaries of their specific methodologies and products. However, both have recognized the need for more generic, cross-methodological reuse strategies. Adequate concepts and methodologies for integrating heterogeneous repositories and registries along with their diverse metadata schemas are still in their infancies though. Similarly, advanced semantic search capabilities have, if at all, been covered only peripherally in other scientific, non-reuse-related contexts. This is probably due to the fact that the Semantic Web constitutes a comparatively young field of research. Thus, how to adopt related standards and technologies in order to provide sophisticated solutions for domain-specific real-world problems still remains non-trivial and of great interest also for other research communities (Dean and Paolucci 2008; Feigenbaum and Heath 2009).
- **Heavy-weight, inflexible development methodologies:** Although the model-driven nature of today's Web Engineering methodologies actually fosters agility and the efficient adoption of changes, they still suffer from model-code gaps and resulting consistency problems. Thus, with respect to agile characteristics like short, iterative development cycles, continuous evolution and the early availability of running versions of the envisioned solution, their suitability is rather limited. This holds true in a similar way for commercial solutions, particularly in the context of model-based approaches like the IBM WebSphere suite. As an exception, Lotus Forms supports a rather agile and evolution-oriented development process. This can be attributed to its low abstraction level as it directly manipulates the final dialog's code. While such

an approach inherently prevents such problems like model inconsistency, its missing abstraction implies severe weaknesses as mentioned above, e.g. missing capability of modeling overall dialog aspects or supporting device-independency by separating abstract, concrete and final dialog design.

- **Restrictive developer-centricity:** Concerning their suitability for strongly involving stakeholders throughout all phases of the development process, both scientific and commercial approaches have turned out to be - either implicitly or explicitly - tailored to a developer audience. Regarding business processes or workflow modeling, for example, all approaches require the use of a fixed, rather developer-centric set of notations and corresponding development environments. Only the IBM WebSphere suite allows for initially importing models from stakeholder-oriented tools like Microsoft Visio or PowerPoint. Beyond that, in order to achieve a running application or to adopt changes, all approaches require a considerable amount of technical-oriented modeling and development effort, which presents a further roadblock towards effective stakeholder engagement. These problems apply similarly both to the construction of workflow-based Web applications in general and advanced Web-based dialogs in particular. As an alternative to today's established Web Engineering methodologies, first approaches towards End-User Development (EUD) in the Web Engineering domain, e.g. (Rode, Rosson and Quinones 2006; Silva and Ginige 2007) are emerging. While aiming at universal and effective stakeholder enablement, they are presently still in their beginnings and suffice only for rather basic Web applications, far from this thesis' problem scope. The consideration of stakeholders, i.e. their characteristics, knowledge and tools, in scientific and commercial reuse approaches has received little attention so far. However, in order to achieve a continuous and effective involvement of stakeholders, their ability to find resolution strategies and related artifacts in accordance with their individual characteristics and skills presents a crucial first step.

Table 3-1: Evaluation of State of the Art Approaches against the Presented Requirements Catalog – Dimensions Workflow and Dialog

	OOHDM	WebML	UWE	IBM
R-WF-01	-	++	-	++
R-WF-02	+	+	+	-
R-WF-03	+	-	--	+
R-WF-04	--	-	--	+
R-WF-05	-	-	-	-
R-WF-06	--	--	--	-
R-WF-07	-	+	+	-
R-WF-08	--	-	-	-
R-WF-09	+	-	++	++
R-D-01	-	-	-	-
R-D-02	+	+	+	-
R-D-03	--	+	--	+
R-D-04	-	-	-	+
R-D-05	--	--	-	-
R-D-06	--	-	-	+

Table 3-2: Evaluation of the State of the Art against the Presented Requirements Catalog – Dimension Reuse

	Scientific WebE Approaches	Commercial Solutions (IBM)
R-R-01	+	+
R-R-02	--	-
R-R-03	-	-
R-R-04	-	-
R-R-05	-	-

Table 3-3: Legend of Rating Symbols

Symbol	Meaning
++	Requirement satisfied / addressed
+	Requirement rather satisfied / rather addressed
-	Requirement rather not satisfied / rather not addressed
--	Requirement not satisfied / not addressed

Table 3-4: Overview of the Requirements Catalog from Chapter 2

Req.-Abbr.	Name
R-WF-01	Workflow Management and Execution
R-WF-02	Continuous, Rich User Interface
R-WF-03	Multimodal Participation
R-WF-04	Federation-Enabled and Component-Based Architecture
R-WF-05	Agility and Evolution
R-WF-06	Multi-Notational Modeling
R-WF-07	Complete Model Continuity and Consistency
R-WF-08	Lightweight Modeling Approach
R-WF-09	Use of Standards
R-D-01	Usability
R-D-02	Device Independence
R-D-03	Web Service Support
R-D-04	Agility and Evolution
R-D-05	Stakeholder Involvement
R-D-06	Standardized Markup Language
R-R-01	Generality and Homogenization
R-R-02	Stakeholder Orientation
R-R-03	Semantic Search
R-R-04	Integrative Reference Architecture
R-R-05	Coordination

4 Web Engineering for Workflow-based Applications – A DSL Approach

The preceding evaluation of the current state of the art arrived at the conclusion that neither a single existing approach nor a combination achieves a sufficient fulfillment of the stated requirements. Consequently, novel solutions for the construction of workflow-based Web applications which satisfy the elaborated requirements and overcome the problems discovered in the current state of the art are required. To this end, this thesis presents several solutions which both individually and combined present significant contributions to the Web Engineering research discipline.

In this chapter, a brief overview of the particular solution elements and their interplay for the construction of workflow-based Web applications in close collaboration with stakeholders is given. First of all, in Section 4.1, a novel approach for the evolutionary construction of Web applications based on Domain-specific Languages (DSLs) is introduced. This framework establishes the foundation for the domain-specific languages, namely the Workflow DSL and the Dialog DSL, presented in this thesis. Afterwards, Section 4.2 conveys an overview of this thesis' contributions and outlines their interrelations and cooperation in the course of the construction of workflow-based Web applications with stakeholders. A detailed description of each solution element follows in the subsequent chapters.

4.1 The Web Engineering DSL Framework¹

The overall vision behind the requirement of strong and continuous stakeholder involvement is to enable domain experts to directly contribute to the development effort by understanding, validating, modifying and even autonomously specifying parts of the solution. As discussed in Section 2.1, this requires specification languages and corresponding notations to be easy to learn, understand and use, both for developers and stakeholders. Consequently, simplicity presents a key factor

¹ Parts of this section have been published in (Nussbaumer, Freudenstein and Gaedke 2006a)

to a language's usability and effectiveness. The inclusion of a problem domain's high-level abstractions and concepts further eases learning, understanding and using a language for domain experts. Against this background, Domain-specific Languages (DSLs) present an interesting alternative to existing, heavy-weight modeling approaches in the Web Engineering domain. A DSL can be defined as follows:

Definition 4.1 - Domain-Specific Language (DSL): *A programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain. (Deursen, Klint and Visser 2000).*

Due to their limited scope and their level of abstraction tailored to the problem domain, DSLs are easy to understand and use, especially for domain experts and non-programmers. By the use of various graphical notations and accompanying editors, each of them being as intuitive as possible for a particular stakeholder group, the usability of a DSL can be further improved (Fowler 2005). Like programs developed with general purpose languages, e.g. Java or C#, DSL programs can also be transformed into executable code or interpreted using a dedicated DSL compiler or interpreter respectively. The advantages of using DSLs do not only affect domain experts and non-programmers; they also comprise potentials for increased productivity, reliability and maintainability (Kieburz, McKinney, Bell et al. 1996) as well as efficient reuse (Batory, Lofaso and Smaragdakis 1998).

In contrast to today's complex and monolithic Web modeling languages, the Web Engineering DSL Framework suggests the use of a multitude of DSLs for the various aspects of a Web application. Thereby, each DSL is tailored to a small, clear problem domain and provides abstractions and notations dedicated to the individual characteristics of relevant stakeholder groups. In Section 4.1.1, the different components making up a DSL as well as their evolutionary character in the course of development for and with reuse are described. Furthermore, the approach's cornerstones addressing the systematic evolution of the emerging variety of DSLs are presented. Following that, Section 4.1.2 briefly introduces the approach's underlying technical platform.

4.1.1 DSLs – Evolutionary Web Development for and with Reuse

Figure 4-1 depicts the elements of the DSL-based Web Engineering approach which is based on the principles of evolution and reuse (Nussbaumer, Freudenstein and Gaedke 2006d). The approach differentiates between two phases in the course of a DSL's continuous evolution: *Development for Reuse* comprises the design and development of a DSL and *Development with Reuse* covers the usage of a DSL for the specification and development of a particular aspect of a Web application. The experiences gained in using the DSL as well as the continuous evolution of the Web, its standards and technologies serve as input for adapting or improving the DSL during the next evolution cycle.

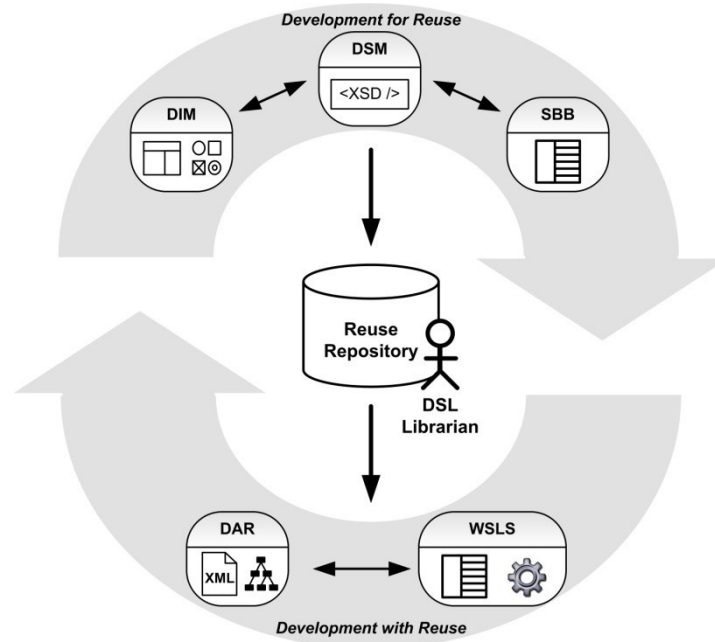


Figure 4-1: Overview of the Evolutionary and Reuse-Oriented DSL-based Web Engineering Approach

4.1.1.1 Development for Reuse

According to the Web Engineering DSL Framework presented here, a DSL consists of three components which are initially developed during the first *Development for Reuse* phase:

- **Domain-Specific Model (DSM):** The DSM embodies the conceptualization of the DSL's respective problem domain in terms of a formalized schema. Usually, the DSM is specified in form of an XML Schema Document (XSD) (Thompson, Beech, Maloney et al. 2004); however, the approach poses no limitations so that other formats are viable as well. As the DSM presents the formal foundation for all solutions that can be specified with the DSL, it has to be designed in strong accordance with the concerned problem domain, thereby taking into account the characteristics of already existing solutions and collaborating with domain experts. To this end, existing approaches from the research fields domain analysis and domain engineering can be adopted, e.g. (Neighbors 1984; Arango 1989; Czarnecki and Eisenecker 2000)
- **Domain Interaction Model (DIM):** A DSL can provide one or more (graphical) notations, termed DIM, whereby each of them is designed as intuitive and appropriate as possible for a particular stakeholder group. By using a DIM, stakeholders can actually employ the DSL, i.e. understand, validate, modify and even create DSL programs, without being confronted with complicated source code. Instead, the DIM should provide concepts and notations derived from the problem domain and thereby should be easy to understand and use. The usability and effectiveness of a DIM can be further improved by providing accompanying DIM-specific editing tools. To this end, preferably tools already known and used by the respective stakeholders should be leveraged. As the

great majority of today's tools and applications provide XML-based export facilities, their integration can be achieved in a straightforward way. The mapping between DIMs and the schema defined by the DSM is realized by dedicated model transformations. In this regard, particularly in the case of multiple DIMs, assuring semantic integrity and consistency is crucial. To this end, Section 5.4 presents an adequate model transformation framework.

A DIM is tightly coupled to the DSM; however, it needs not to cover all of its aspects. The DSM can be projected onto various DIMs, each of them corresponding to a particular level of abstraction and a specific stakeholder audience (cf. Figure 4-3). Hence, for a given problem domain, a taxonomy of DIMs can be spanned, thus providing adequate support for all audiences and process stages. For example, in the figure, DIM₁ represents a very basic, cross-audience DIM for early requirements engineering activities, thus covering only a small fraction of the DSM. DIM₂ and DIM₃ could be notations which cover most of the problem domain and are tailored to different stakeholder groups. Regarding, for example, a DSL addressing workflow aspects, each of these two DIMs could adhere to a different business process modeling notation. Finally, DIM₄ offers full coverage of the problem domain and thus, due to its expressiveness and the resulting complexity, rather addresses a developer audience.

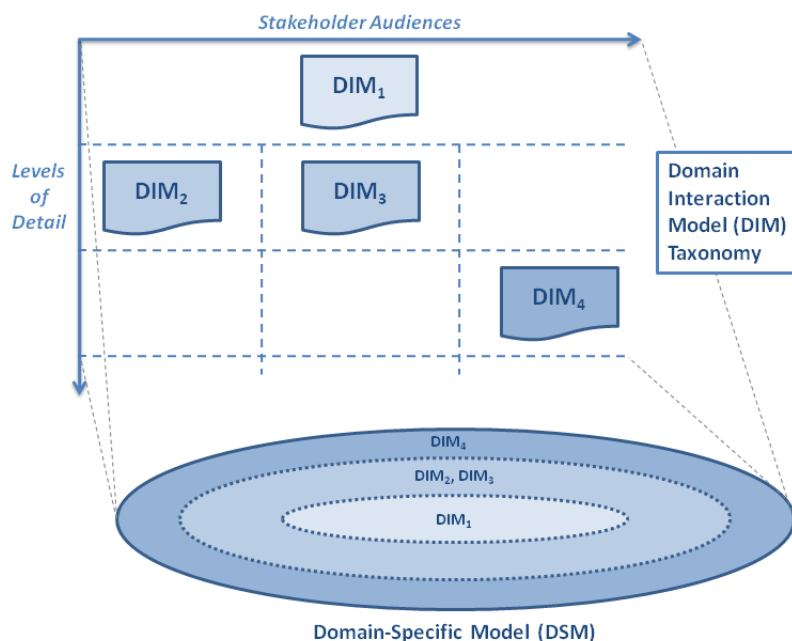


Figure 4-2: Projection of the Domain-Specific Model (DSM) onto the Domain Interaction Model (DIM) Taxonomy

- **Solution Building Block (SBB):** For the execution of DSL programs developed using one or more DIMs and based on the DSM, each DSL provides a dedicated SBB. Unlike the majority of today's model-driven software development approaches, the Web Engineering DSL Framework does not pursue a model-to-code transformation conception, but rather conceives DSL programs as instrumentation for a SBB. Consequently, a SBB is considered as

a domain-specific software component which is capable of adapting its behavior in accordance with a given, usually XML-based, DSL program.

4.1.1.2 Development with Reuse

Once a DSL comprising a DSM, one or more DIMs and a SBB has been initially developed and stored in the reuse repository, it can be retrieved and employed in the *Development with Reuse* phase. In this phase, a DSL program termed *Domain Abstract Representation (DAR)* specifying a concrete solution within the DSL's problem domain is developed. Consequently, it is based on the DSM and created by using one or more DIMs. As the DSM is usually specified in terms of an XML Schema, the DAR is likewise serialized and stored in form of an XML document based on the DSM. Ideally, and in contrast to today's integrated development environments, the editing process using DIM notations is not performed on this serialized form. Modifications are rather carried out directly on the abstract model itself. Thus, DSL programs can be edited in a more powerful way than it would be possible if interacting with the DAR's serialized form.

After having developed a DAR, it is passed – usually in form of an XML document - to an instance of the DSL's associated SBB which in turn adapts its behavior accordingly and thus implicitly executes it. SBBs run on a technical platform which allows their composition and configuration with DSL programs at runtime. In this regard, the Web Engineering DSL Framework approach could be basically applied to most of today's portal platforms, e.g. Microsoft Office SharePoint Server or IBM WebSphere Portal Server. The minimum requirements such a portal system must fulfill lie in supporting the integration of custom Web components and their configuration at runtime. In this thesis' research context, the technical implementation was performed based on the *WebComposition Service Linking System (WSLS)* and is briefly described in Section 4.1.2.

Summarizing the *Development for Reuse* phase, Web application development can be performed in an evolutionary manner by composing SBBs and configuring them with DARs. These are in turn developed in strong collaboration with stakeholders using one or more DIMs and supplementing DIM editors.

4.1.1.3 Systematic Evolution

Domain-Specific Languages are subject to continuous evolution. Their lifecycle starts with the identification of the need for a DSL for a particular, presently uncovered problem domain. This is followed by the specification of a new DSL, consisting of a Domain-Specific Model, one or more Domain Interaction Models and dedicated DIM-specific editor support. The new DSL is then employed in various scenarios and in collaboration with different kinds of stakeholders. Thereby, new experiences are gained permanently and eventually result in requests for change. For example, such improvements could concern the modification of a DIM, the development of a new DIM for a particular stakeholder group or extensions to the DSM due to hitherto uncovered aspects of the problem domain. Thereupon, the DSL enters a new evolution cycle and the required adaptations and extensions are performed in the *Development for Reuse* phase.

Hence, the set of available DSLs for building Web applications underlies a continuous evolution through variation and selection: new DSLs are added, existing DSLs are improved, and DSLs that have turned out to be dispensable are removed. Beyond that, also the focus on a multitude of highly-focused DSLs for concise problem domains requires an efficient reuse management approach. To this end, the Web Engineering DSL Framework suggests two main concepts: A *DSL Reuse Repository* for the systematic management of DSLs from the technical point of view and a team role called *DSL Librarian* being responsible for their efficient management and usage from the process perspective.

The DSL Reuse Repository is the central place for organizing, storing, managing and accessing DSLs and their components as well as associated metadata. Moreover, the repository should provide versioning features in order to cope with the continuous evolution of the stored components. With respect to assuring efficient storage and retrieval of DSLs, a sophisticated classification scheme supporting context-based searches is necessary. For example, it should be possible to find DSLs according to parameters like the problem domain, the application type, the kind of stakeholders etc. Against this background, Chapter 7 presents an adequate reuse repository approach termed *The Web Engineering Reuse Sphere*. Even though the Web Engineering Reuse Sphere has a much more comprehensive and generic focus than required in this context, it presents an ideal supplement providing excellent support for the DSL-based Web Engineering framework.

Facing challenges like effectively creating and maintaining such a repository as well as psycho-social impediments to software reuse, e.g. the “not invented here” syndrome (Sommerville 2007a), the explicit promotion of an reuse-oriented approach like the Web Engineering DSL Framework also from an organizational perspective becomes important. To this end, the DSL Librarian team role accompanies the entire DSL lifecycle and promotes their usage. During the specification of new DSLs, it advocates the project team and is responsible for the avoidance of duplicate or badly reusable DSLs. Similarly, it fosters the reuse of existing DSLs and related artifacts where possible and therefore supports the team in finding and evaluating appropriate DSLs. Furthermore, the role is in charge of effective maintenance of the repository which includes tasks like adding and removing components, updating the classification scheme as well as monitoring successful and unsuccessful searches and adoptions.

4.1.2 Technical Platform

The Web Engineering DSL Framework approach can be adequately supplemented by the *WebComposition Service Linking System (WSLS)* (Gaedke, Nussbaumer and Meinecke 2005) which emerged in the context of Martin Nussbaumer’s PhD thesis (Nussbaumer 2008) and serves as technical platform for SBBs. Similarly, other commercial portal systems could be adopted as technical platforms for the Web Engineering DSL Framework.

WSLS aims at facilitating the systematic evolution of Web applications by reusing software artifacts, particularly software components, and emphasizing the “configuration instead of programming” paradigm. The WSLS framework suggests a two-layered perception of Web applications and their construction by differentiating between an *application level* and a *configuration level*. On the application level, a Web application’s various sites and their structure are defined by referencing and composing atomic units termed *domains*. Domains are the primary conceptual units which serve for specifying a Web application’s structure and composition. On the configuration level, a domain is assigned with a desired behavior type from a repository of available behavioral building blocks. Such behavioral building blocks can be, amongst others, SBBs as defined by the Web Engineering Framework approach. The concrete behavior of such a building block in the context of a particular domain is configured by a dedicated configuration set. The configuration is based on typed name-value pairs, termed *properties*, which represent very fine-grained atomic units of configuration and which are categorized into six Web-specific concern dimensions, i.e. data, navigation, interaction, presentation, process and communication.

In contrast to the available multitude of very fine-grained properties, a DSL program represents a more coarse-grained configuration document, influencing multiple properties from various concern dimensions. Hence, from a conceptual perspective, configuring a SBB with a DSL program equals the configuration of multiple properties across the various concern dimensions. WSLS allows for evolving a Web application both on the application and the configuration level at runtime. Thus, it is perfectly suitable for a highly agile and evolutionary DSL-based construction approach: Web applications are built and evolved by assembling SBBs for their various concerns and configuring them with DSL programs at runtime. These DSL programs in turn are developed and evolved in strong collaboration with stakeholders using (graphical) DIM notations and related editors. Thus, in the most cases, no manual development of source code is required. In conclusion, the Web Engineering DSL Framework approach and the WSLS platform thereby realize the “modeling instead of programming” paradigm.

4.2 Overview of Solution Elements

As pointed out in the previous section, the Web Engineering DSL Framework is adequately supplemented by two additional approaches, namely the Web Engineering Reuse Sphere and a dedicated Model Transformation Framework. With regard to this thesis’ problem scope, i.e. the efficient and effective construction of workflow-based Web applications in strong collaboration with stakeholders, further solution elements based on this foundation are required. In this regard, Figure 4-3 illustrates the various elements and contributions for the stakeholder-oriented construction of workflow-based Web applications which are presented in this thesis. It inherently combines the visualization of the individual solution elements and their interrelations with an assignment to the high-level process model as suggested by

the Workflow DSL. In the following, each solution element is briefly introduced, thereby pointing out interrelations with other elements and conveying an impression of their interplay in the course of constructing workflow-based Web applications.

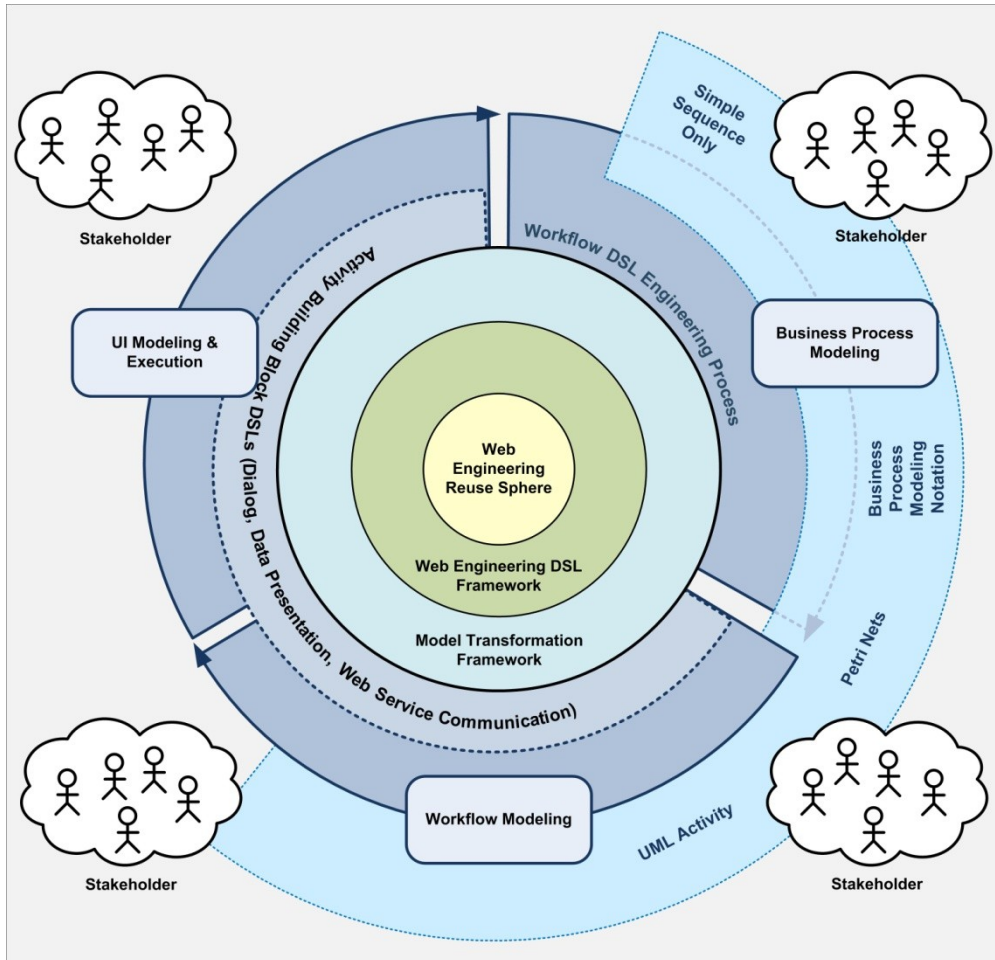


Figure 4-3: Complete Overview of the Presented Contributions for the Stakeholder-Oriented Construction of Workflow-based Web Applications

- **The Workflow DSL:** The Workflow DSL presents the central solution element for the model-driven construction of workflow-based Web applications. It is realized as a Domain-specific Language following the specification introduced by the *Web Engineering DSL Framework*. The Workflow DSL suggests an evolutionary, high-level process model which ranges from initial requirements engineering activities to obtaining a fully functional workflow-based Web application from the constructed models and consists of three phases: *Business Process Modeling*, *Workflow Modeling* and *UI Modeling and Execution*. The DSL adopts various modeling notations as Domain Interaction Models (DIMs) and supports common tools, thus covering various stakeholder groups and levels of detail. These notations include a simple text-based notation for early requirements engineering activities termed *Simple Sequence Only*, and more detailed notations like the *Business Process Modeling Notation (BPMN)*, *Petri Nets* or *UML Activity Diagrams*. Beyond that, the DSL defines systematic extension facilities for incorporating further

notations and tools. The DSL's Domain-Specific Model (DSM) serves as *Process Intermediate Language*, capturing all relevant concepts of the respected problem domain and thus representing the formalized foundation for the fully model-based construction process as well as for integrating the variety of modeling notations. To this end, a dedicated *Model Transformation Framework* supports the systematic development and execution of bilateral model transformations between (almost) arbitrary modeling notations (DIMs) and the Process Intermediate Language (DSM). With respect to the technical realization of workflow activities, the Workflow DSL employs a set of so-called *Activity Building Blocks (ABBs)*. The Workflow DSL is described in detail in Chapter 5.

- **Activity Building Blocks (ABBs):** ABBs are autonomous Domain-Specific Languages that are loosely integrated by the Workflow DSL for realizing various common workflow activities like *dialog-based user interaction*, *data presentation* or *Web Service communication*. Thus, an ABB type is assigned to each workflow activity and configured with a minimal set of properties. Thereupon, due to their advanced automation concepts, ABBs are already capable of implementing the desired behavior type. Like any DSL based on the Web Engineering DSL Framework, they allow for conducting a more comprehensive, detailed design at runtime and in strong collaboration with stakeholders. In summary, ABBs foster the efficient Web-based realization of workflow activities and contribute to keeping the focus on the business process or workflow model respectively instead of losing the overview in a multitude of models. Besides being employed by the Workflow DSL, ABBs can also be used autonomously in other contexts, e.g. for the sole realization of a Web-based dialog. The catalog of ABBs is introduced in Section 5.2.2
- **The Dialog DSL:** Due to the particular importance of complex Web-based dialogs in workflow-based Web applications, the Dialog DSL presents the main Activity Building Block. Like all DSLs presented in this thesis, it follows the DSL specification of the *Web Engineering DSL Framework*. The Dialog DSL allows for the automated generation of device-independent and Web Service-enabled dialogs from data schemas or Web Service specifications. Furthermore, it provides an intuitive modeling notation focusing on dynamic behavior and usability and includes a supplemental Web-based model editor. Like the Workflow DSL, it is purely model-driven, thus not requiring any manual coding for the great majority of dialogs. Hence, it enables the rapid development and evolution of advanced Web-based dialogs, even by stakeholders without programming skills. This was successfully verified in the course of a comprehensive empirical evaluation described in Section 8.3. An in-depth presentation of the Dialog DSL itself is given in Chapter 6.
- **The Model Transformation Framework:** Particularly in the context of the Workflow DSL, model transformations play an important role. While *horizontal model transformations* address the transformation of a business process or workflow model respectively between various notations and tools, *vertical model transformations* accomplish the transformation of a workflow model to an executable workflow specification used as input for a workflow

engine. In this context, the Workflow DSL's Domain-Specific Model (DSM) acts as intermediate schema, thus fostering extensibility and efficiency with regard to the envisioned multitude of supported modeling notations, tools, execution formats and platforms. With respect to the development and execution of adequate transformations, the Model Transformation Framework assures key factors like semantic integrity and consistency. Its supplemental technical platform allows for specifying, managing and executing model transformations as well as their seamless integration into existing tools and platforms. The Model Transformation Framework is presented in Section 5.4.

- **The Web Engineering Reuse Sphere:** A comprehensive, cross-methodological and stakeholder-oriented reuse framework for the Web Engineering domain forms a fundamental complement to the approaches presented in this thesis. On the one hand, as described in the previous section, it can be adopted as DSL reuse repository as required by the Web Engineering DSL Framework. As such, it enables efficient and effective storage and retrieval of DSLs including their various components as well as DSL programs. On the other hand, it enables stakeholders to find resolution strategies and related artifacts for a given problem and a specific set of individual skills across the great diversity of Web Engineering methodologies. Therefore, the Web Engineering Reuse Sphere defines an ontology which conceptualizes the domain reuse in the Web Engineering discipline based on Semantic Web standards and technologies. Based on this semantic, homogenizing foundation, it provides advanced knowledge-based, cross-methodological search facilities. The technical integration of existing heterogeneous artifact stores is guided by a reference architecture. Thereby, besides explicit reuse repositories, also application-specific stores are covered in order to support both planned and spontaneous reuse strategies. The Web Engineering Reuse Sphere is presented in Chapter 7.

5 Constructing Workflow-based Web Applications with Stakeholders²

In this chapter, the *Workflow DSL* as central solution element for the efficient and effective construction of workflow-based Web applications with stakeholders is presented. The Workflow DSL adheres to the Domain-Specific Language specification as introduced by the *Web Engineering DSL Framework* in the previous chapter and explicitly addresses the requirements identified in Chapter 2. Consequently, after a brief overview of the Workflow DSL, its elements and a complementary evolutionary process model in Section 5.1, a detailed presentation of each DSL element is given. Section 5.2 describes the DSL's *Domain-Specific Model (DSM)* serving as formalized conceptualization of the considered problem domain and thereby acting as *Process Intermediate Language* for the various modeling notations. In this context, the catalog of *Activity Building Block (ABB)* DSLs used for specifying the technical realization of workflow activities is introduced. The various *Domain Interaction Models (DIMs)* enabling multi-notational modeling of workflow-based Web applications in strong collaboration with stakeholders and throughout all phases of the development process are presented in Section 5.3. The *Model Transformation Framework* supporting the systematic development and execution of the required horizontal and vertical model transformations, i.e. between various DIMs and the DSM as well as from the DSM to executable workflow specifications, is described thereafter in Section 5.4. Finally, Section 5.5 contains an in-depth presentation of the Workflow DSL's *Solution Building Block (SBB)*, the underlying technical reference architecture as well as the automated application generation methodology. A brief summary including a short evaluation against the identified requirements from Chapter 2 is given in Section 5.6.

² Parts of this chapter have been published in (Freudenstein, Nussbaumer, Majer et al. 2007)

5.1 The Workflow DSL at a Glance

According to the specification of the Web Engineering DSL Framework, the Workflow DSL is an executable specification language tailored to the domain of Web-based workflow execution and interaction. It allows the simultaneous use of various graphical modeling notations known from the Business Process Modeling field and supplemental editing tools, thereby focusing on process aspects instead of technical details as well as easing continuous involvement and participation of various stakeholder audiences. The Workflow DSL enables an evolutionary, completely model-based development process with short iteration cycles. Based on the DSL's technical framework, service-oriented and federation-enabled workflow-based Web applications can be fully automatically derived from model-based DSL programs.

5.1.1 Elements of the Workflow DSL

Corresponding to the Web Engineering DSL Framework, the Workflow DSL consists of three core elements:

- **Domain-Specific Model (DSM):** The DSM represents the formal schema for all DSL programs or Web-based workflows respectively that can be specified with the DSL. With respect to supporting various business process modeling notations, the DSM can also be seen as *Process Intermediate Language*, embodying the common denominator of the multitude of existing process modeling languages. Besides constructs covering a business process' functional, behavioral, informational or organizational perspective, the DSM also includes dedicated modeling constructs enabling the transition from a pure business process model to a running workflow-based Web application. As basis for the DSM, the standardized XML Process Definition Language (XPDL) (Shapiro, Marin, Brunt et al. 2005) was chosen. XPDL was originally designed both as an interchange format for process definitions and as a definition language for executable workflow specifications. Thus, it forms an ideal foundation for the Workflow DSL. Based on the extensibility mechanisms provided by XPDL, dedicated concepts, primarily concerning Web-specific user interface concerns, were integrated. Due to these well-defined extensions, the expressive power of the XPDL standard was extended towards a full coverage of Web-based workflows, thus making it an ideal DSM for the Workflow DSL.
- **Domain Interaction Models (DIMs):** According to the Web Engineering DSL Framework, multiple DIMs tailored to various stakeholder groups and levels of detail can be defined. With respect to the considered problem domain, a DIM could either be derived from a well-known business process modeling notation or be defined from scratch embodying a custom notation. Accompanying editors, e.g. business process modeling tools as well as

everyday office applications well-known by stakeholders, can be integrated in order to support stakeholders in creating DSL programs based on a DIM notation. In the context of this thesis, various DIMs were designed and realized based on the DSM and according to the following standards: The Business Process Modeling Notation (BPMN), UML Activity Diagrams, Petri Nets as well as a custom text-based notation named Simple Sequence Only. To model DSL programs using these notations, the following tools were exemplarily integrated (same order as notations): Microsoft Visio, IBM Rational Software Architect, INCOME2010, and Microsoft Word. While the latter DIM covers only basic process structure aspects, the three former DIMs comprise also technical, Web-related workflow aspects. Dedicated *model transformations* realize the consistent and semantically lossless horizontal transformation of the shared model between various notations and tools. According to the Web Engineering DSL Framework and based on a unique model transformation concept, the set of available notations and tools can be efficiently and systematically extended.

- **Solution Building Block (SBB):** The Workflow DSL's SBB represents the central component of the approach's technical framework. The SBB can be configured with an XML-based specification of a Web-based workflow, i.e. a Workflow DSL program adhering to the DSM. Thereupon, the SBB constructs an associated Web-based workflow at runtime. Therefore, part of the DSL program is extracted and transformed into an adequate instrumentation for a workflow engine which is in charge of controlling the process flow. On the other hand, with respect to the Web-based user interface and the realization of workflow activities, instances of various SBBs belonging to a catalog of so-called *Activity Building Block DSLs* are composed. These serve for the realization of workflow activities like Web-based dialogs, data presentation or Web service communication and are initialized with a minimum configuration set derived from the DSL program. While this is sufficient for their correct operation, their configuration can be refined at runtime by using their respective DSL's DIMs and associated editors.

5.1.2 Evolutionary Process Model

The process model depicted in Figure 5-1 guides the application of the Workflow DSL within an evolutionary development process for workflow-based Web applications. It consists of three phases in a continuous evolution and involves various roles. Due to its openness, it can be smoothly integrated into existing development methodologies as a concern-specific process model for Web-based workflows. In the following, the process model's three incremental phases as well as the activities performed therein and the involved roles are described. It should be noted that, throughout all phases, one single DSL program is created and consistently evolved, even though various DIMs covering different levels of detail are employed. Beyond that, the process can be mostly performed on a pure model-basis without requiring manual coding.

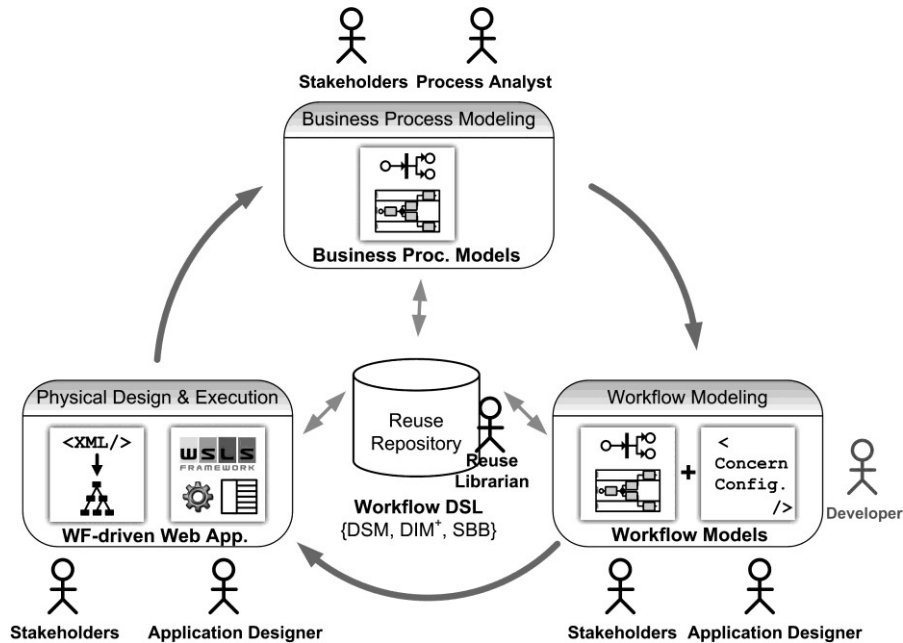


Figure 5-1: Overview of the Evolutionary Workflow DSL Process Model

Business Process Modeling: In this initial phase, the business process to be realized by the workflow-based Web application is modeled using pure business process modeling constructs. While in a first step, the focus usually lies on eliciting and modeling process activities and roles, the model is incrementally refined by more detailed constructs including control and data flows. In doing so, *stakeholders* representing the involved process participants and knowing the business process best as well as a *Process Analyst* role supporting the modeling process itself are involved. Depending on the comprehensiveness of the business process, different stakeholder groups contribute to different sections of the process and on various levels of abstraction. According to the involved stakeholder group and the considered level of detail, various DIM notations and model editors can be employed. Adequate model transformations serve for automatically transforming the shared process model from one DIM into another. The *Reuse Librarian* role advises the modeling team regarding possibilities for reusing existing process models in whole or part as well as assures the effective storage of developed artifacts in the repository (cf. the Web Engineering Reuse Sphere in Chapter 7). A business process model in form of a DSL program represents the output of this phase.

Workflow Modeling: In this phase, the business process model from the previous phase is augmented with information addressing the technical realization of the business process in form of a Web-based workflow. As this particularly implies well-known Web-specific concerns (Schwinger and Koch 2006) like data, presentation, interaction, communication or process, this set is termed *Concern Configuration*. In the course of this phase, first of all, to each activity in the business process model a corresponding *Activity Building Block (ABB)* is assigned. Thereby, the realization of the activity is determined from a conceptual perspective. Subsequently, each ABB is configured with a minimum set of properties according to its respective domain-specific schema. These tasks can be performed in strong collaboration with

stakeholders: On the one hand, the conceptual mapping of activities to ABBs requires a rather low technical understanding. On the other hand, throughout this phase, the focus still remains on the business process model which is solely augmented by the property-based Concern Configuration. Thereby, usually DIMs and accompanying editors known from the previous phase can be used. The difference of these DIMs lies only in an extension in order to specify the Concern Configuration, whereas the DIM's individual notation for the business process itself remains identical. The involved stakeholders are supported by an *Application Designer* role and the *Reuse Librarian* role. The application designer is experienced in workflow modeling and knows the activity building blocks and the associated DSLs. The reuse librarian advises the team concerning the reuse of existing Concern Configurations and Activity Building Blocks from the reuse repository. If no suitable ABB or DSL respectively exists, the *Developer* role initiates the design and implementation of a new one which is not covered any further by this process model. The result of this phase is a valid Workflow DSL program in form of an XML document, wherein process structure information and Concern Configuration are loosely coupled, thus easing reuse and evolution.

Physical Design & Execution: In this phase, the developed DSL program is passed to an instance of the Workflow DSL's SBB, which thereupon fully automatically supplies a corresponding Web-based workflow. Subsequently, the obtained Web-based workflow both be modeled in detail by means of the Activity Building Blocks DSLs and directly be used in production for creating and processing workflow instances. For example, the Activity Building Block used for the realization of Web-based dialogs, i.e. the *Dialog DSL* (cf. Chapter 6), provides a Web-based model editor. Thus, dialogs or their models respectively, which were generated based on the minimum configuration set provided in the Workflow Modeling phase, can be comprehensively refined in-place and at runtime. Similar to the previous phases, stakeholders can strongly participate in this phase, again assisted by the *Application Designer* role.

Evolution: In the case of changing or new requirements, the Workflow DSL provides strong support for adopting changes, either in the business process model or the Concern Configuration or both. Changes in the business process can easily be performed in the Business Process Modeling Phase while keeping the Concern Configuration in the Workflow Modeling phase unchanged. Changes in the Concern Configuration can either be performed by modifying the minimum configuration set in the Workflow Modeling phase or at runtime in the Physical Design & Execution phase by means of the respective DSLs. Both the DSL approach itself and the technical platform preserve model consistency throughout all phases.

5.2 The DSM – Process Intermediate Language

An adequate Domain-Specific Model (DSM) for the Workflow DSL approach should satisfy various requirements. Firstly, considering the multitude of existing standardized specification languages in this domain, it should be preferably based on such a standard. Secondly, besides covering functional, behavioral, informational and

organizational workflow aspects, it should also provide support for specifying (Web-) user interface-related aspects in the context of human tasks. Thirdly, it should provide a sufficient coverage of relevant workflow patterns, i.e. abstract, language-independent patterns which potentially arise from a business perspective and thus should be supported by a workflow specification language (Van der Aalst, ter Hofstede, Kiepuszewski et al. 2003; Russell, ter Hofstede, Van der Aalst et al. 2006). Fourthly, with respect to the goal of supporting various existing process modeling notations in form of Domain Interaction Models (DIMs) based on the DSM, it should be sufficiently universal and embody a common foundation, i.e. a *Process Intermediate Language*. Lastly, it should provide explicit extension points and mechanisms for incorporating missing concepts, e.g. regarding Web-specific concerns. Facing these requirements, the XML Process Definition Language (XPDL) was identified as an ideal foundation for the Workflow DSL's DSM.

5.2.1 The XML Process Definition Language as Foundation for the DSM

The XML Process Definition Language (XPDL) is an XML-based workflow specification language standard by the Workflow Management Coalition (WfMC) (Shapiro, Marin, Brunt et al. 2005). Originally intended as a file format for the visual Business Process Modeling Notation (BPMN) (White 2006), it has been adopted by more than 80 different products yet (Workflow Management Coalition 2009).

In the context of workflow specification languages, often questions regarding the difference between XPDL and the well-known Business Process Execution Language (BPEL) (Jordan and Evdemon 2007) arise. XPDL and BPEL are entirely different yet complimentary standards. BPEL was designed as an execution language for Web services orchestrations, thus focusing only on the executable aspects of a process thereby exclusively dealing with Web services and XML-based data. However, the BPEL standard does not cover aspects concerning the graphical diagram, human oriented processes, sub-process, and many other aspects of a modern business process. Thus, instead of being competing standards, XPDL and BPEL should rather be considered as mutual complements forming a value chain (Palmer 2006).

Figure 5-2 illustrates the formal high-level metamodel for process definitions based on XPDL 2.0 which is also available as detailed specification in form of an XML Schema document. The root element *Workflow Process* comprises a workflow specification including *Type Declarations*, *Data Fields*, *Participants*, *Activity Sets*, *Activities*, and *Applications*. *Transitions* represent sequential control flow between *Activities*, which in turn can be *Tasks* or control flow-related constructs (*Route / Gateway* with different types like AND, OR, XOR etc.), amongst others. To an *Activity*, a *Participant* and an *Application* can be assigned, thereby specifying which activity is to be performed by whom and based on which application definition. *Participants* can be of various types including systems and humans. Similarly, several application types are defined by the standard including both system-oriented and human-oriented application types. Regarding the former, predefined types for calling Enterprise JavaBeans (*EJB*), methods on local Java classes (*POJO*), Extensible

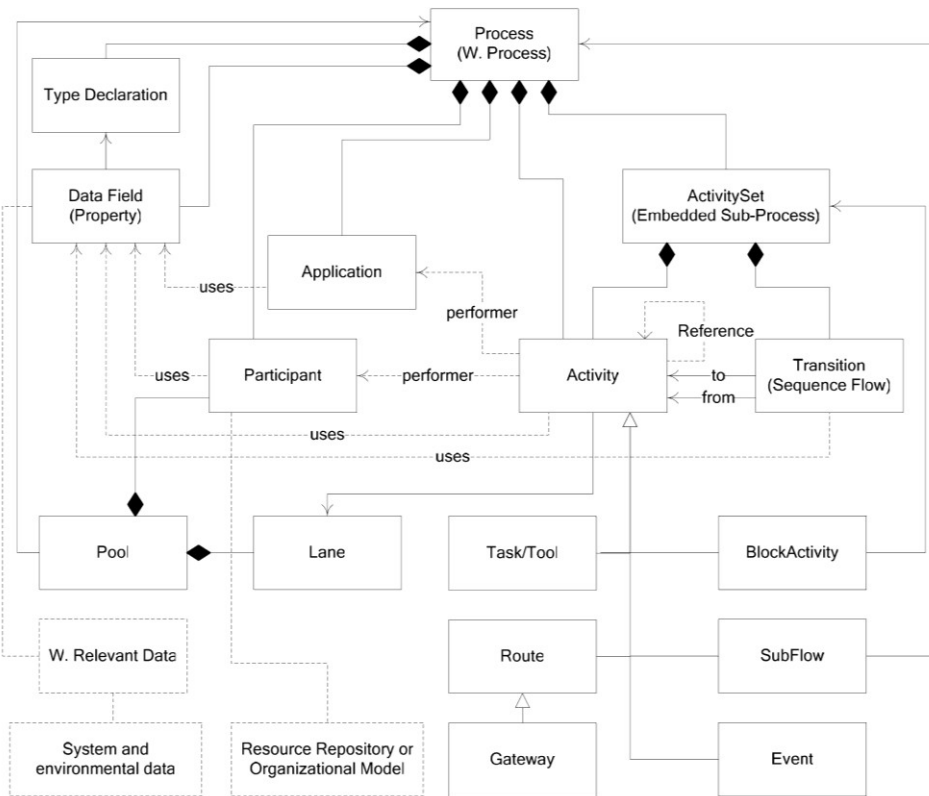


Figure 5-2: Overview of the XPD Process Definition Metamodel.
 Taken from: (Shapiro, Marin, Brunt et al. 2005)

Stylesheet Language Transformations (*XSLT*), expressions (*Script*), Web Services (*WebService*) or invoking business rules (*BusinessRule*) are available. With regard to human-oriented tasks, one predefined application type representing form-based user interaction (*Form*) is provided.

With respect to the above-stated requirements a DSM should satisfy, XPD presents an eminently suited choice: Firstly, as already mentioned, XPD is a widely-adopted standard and is actively advanced by the WfMC. Secondly, it provides basic coverage for human tasks, even though major extensions are required in order to achieve sufficient support for Web-specific concerns and thus for Web-based workflows in general. Thirdly, XPD in its second version offers broad coverage of relevant workflow patterns (Workflow Patterns Initiative 2007). Fourthly, both this broad coverage of workflow patterns, particularly in comparison with common modeling notations, as well as its intention as process interchange format, even though the original vision hereby was restricted to interchanging BPMN models across various tools, qualify XPD as a well-suited *Process Intermediate Language*. Lastly, the XPD specification includes various explicit extension points in its XML Schema definition as well as the possibility to integrate namespace-qualified extensions to all XPD elements.

The *Application* construct and the predefined application types as well as the extension mechanisms provided by XPD form the primary starting point for integrating Web-specific concerns into the DSM or the XPD standard respectively. These extensions mainly address the integration of concepts and constructs in order

to specify the Web-based realization of workflow activities. Thus, first of all, the catalog of Activity Building Blocks (ABBs) is introduced in the next section, whereas Section 5.2.3 describes their formalization and integration into the XPDL standard.

5.2.2 Catalog of Activity Building Blocks (ABBs)³

In the following, a catalog of three major Activity Building Blocks (ABBs) is presented. These ABBs are realized as DSLs according to the DSL specification of the Web Engineering DSL Framework and are used for the Web-based realization of workflow activities. A major design goal for the ABBs was that one activity from a business perspective can be realized by one ABB and must not be split up into several activities from a system perspective. Thus, the business process model's structure can be kept throughout the construction process, easing the collaboration with stakeholders.

A second important goal was to design the ABBs and particularly their respective SBB's in a way that requires *minimal initial configuration* but still yields the desired behavior. After a workflow-based Web application has been fully-automated set up by the Workflow DSL's SBB via composing instances of the respective SBBs and configuring them with this minimum configuration set, a detailed design addressing more fine-grained aspects can be performed at runtime. Therefore, the DIM(s) and associated editor(s) belonging to a workflow activity's respective ABB can be used. In the context of this chapter, the required initial minimum configuration set is of particular interest as it has to be integrated into the Workflow DSL's DSM and DIMs in order to support the specification and configuration of ABBs during the *Workflow Modeling* phase. Consequently, the following presentation of each ABB concentrates on a short description and the *minimal physical configuration aspects* representing this minimum configuration set. This specification for each ABB presents an interesting contribution also beyond the scope of the Workflow DSL: Based thereupon, highly reusable and generic Web-based software components embodying the respective behavior type can be designed and implemented. The applicability of such components extends far beyond the Workflow DSL approach to the development of Web-based solutions in general.

The presented ABBs cover activity types like *Dialog-based User Interaction*, *Data Presentation* and *Web Service Communication*, whereas the Web Service Communication ABB can be used as a supplement to other building blocks. Hence, for example, the Dialog-based User Interaction ABB can submit a filled form to a Web service via delegation to the Web Service Communication ABB.

³ Parts of this section have been published in (Freudenstein, Nussbaumer, Majer et al. 2007)

5.2.2.1 Dialog-based User Interaction – The Dialog DSL

This ABB represents dialog-based user interaction activities via Web forms and is described in detail in Chapter 6. The Dialog DSL's SBB is capable of automatically creating a basic, but fully operational Web-based dialog from a data schema. This can either be provided in form of an XML Schema document (Thompson, Beech, Maloney et al. 2004) or automatically extracted for a particular Web service method from a Web service specification in form of a Web Service Description Language (WSDL) document (Christensen, Curbera, Meredith et al. 2001). In the latter case, the generated dialog is already fully Web service-enabled, thus being capable of submitting the filled dialog to the Web service and processing the result. Beyond that, the SBB is able to perform runtime adaptations on the dialog according to the requesting client device's characteristics, thus fostering device-independent access.

With respect to the interaction of the Dialog DSL's SBB with the workflow, the SBB can receive an instance of its associated data model at runtime whereupon the associated dialog fields are assigned with the corresponding data instance values. Similarly, the SBB returns the submitted form back to the workflow.

Minimal physical configuration aspects:

- *Dialog ID*: Instead of generating a new dialog, an existing dialog from the dialog repository can be referenced via this property.
- *Generation Basis*: Used to specify whether the dialog should be based on an XML Schema document or according to a Web service method and the related specification.
- *XML Schema Document*: In the case of XML Schema-based generation, this property can be used for providing a URL to an XML Schema document.
- *XPath Selector*: Per default, the complete schema is used as generation basis for the dialog whereby the generation starts with the root element's associated type and then recursively parses the schema tree. However, if only a particular type defined in the XML Schema document shall be used as generation basis, a corresponding XPath expression (Berglund, Boag, Chamberlin et al. 2007) can be supplied in this property.
- *Web Service Interface Description*: In the case of Web service-based dialog generation, this property serves for specifying a URL to the Web Service Description Language (WSDL) document.
- *Web Service Operation Name*: In addition to the interface description, the name of the Web service operation the generated dialog should be capable to communicate with has to be supplied in this property. Based on these two properties, the data schema required for the generation can be extracted in form of an XML Schema from the WSDL document.

5.2.2.2 Commit

The Commit ABB presents a specialized form of dialog-based user interaction. It is used to indicate the completion of a physical workflow activity which can actually

not be supported by an application. Examples of such activities could be shipping a package or holding a face-to-face meeting. Such workflow activities are represented by the Commit ABB which renders a Web-based dialog containing solely a checkbox or button enabling the confirmation of the activity's completion. Consequently, it requires no configuration and thus does not define any physical configuration aspects. Regarding the interaction with the workflow, the Commit ABB's SBB receives no input parameters from the workflow and returns a token indicating the commit of the respective workflow activity back to the workflow.

5.2.2.3 Data Presentation

The Web-based presentation of XML-based data to a user is covered by this ABB. This type of ABB is suitable for all workflow activities which require the sole presentation of data without having users to interact with it, e.g. displaying notifications, receipts or confirmations. As already mentioned, by coupling this ABB with the Web Service Communication ABB, also data retrieved from Web services can be rendered. The transformation of XML data into markup in a desired output format is achieved based on appropriate Extensible Stylesheet Language Transformations (XSLT) (Clark 1999) which are applied to the given XML data at runtime.

The ABB's SBB is capable of automatically generating a XSL transformation realizing a basic rendering of the given XML data into XHTML. This is achieved by analyzing and traversing the tree-based structure of the given XML document and performing a pattern-based translation of XML sections into XHTML markup. Besides structural-oriented patterns addressing the presentation layout, also patterns concerning the individual representation of particular elements can be employed. For example, values in the XML document matching a regular expression-based pattern for URLs of pictures or video can be translated into corresponding type-specific markup tags which directly render the image or video instead of displaying the URL. The algorithm can be further improved by reusing and composing existing transformation snippets for specific XML elements which can be identified and assigned based on their XML namespaces. Therefore, the SBB has to be connected to a reuse repository for storing and retrieving XSL snippets.

Beyond that, the pattern-based translation algorithm assigns adequate predefined Cascading Style Sheets (CSS) (Bos, Çelik, Hickson et al. 2007) classes to the XHTML markup. Thus, the foundation for a more detailed design based on CSS is established. To this end, the Data Presentation DSL is ideally supplemented by a browser-based DIM editor named "Lyra" which facilitates a detailed, reuse-oriented presentation design based on CSS at runtime and was developed in the context of Martin Nussbaumer's PhD thesis (Nussbaumer 2008).

With respect to device-independent access, the Data Presentation SBB can perform two types of runtime adaptations: On the one hand, a pagination algorithm can be integrated in the XSL transformation and distribute the data on various interlinked client-specific display units. Such an algorithm is also described in the context of the Dialog DSL in Chapter 6. On the other hand, different CSS definitions, each of them

tailored to a specific class of client devices, can be defined and dynamically linked at runtime.

Concerning the interaction of the Data Presentation ABB's SBB with the workflow, the SBB receives XML-based data to be presented from the workflow and returns a token indicating the completion of the activity back. Hence, as the activity type actually consists in pure data presentation or, from a user's perspective, in data viewing respectively, the SBB provides a dedicated user control allowing the user to confirm the completion of the workflow activity.

Minimal physical configuration aspects:

- *XSL Transformation Document*: This optional property serves for referencing an existing XSLT document from the reuse repository. If no value is provided, the SBB performs a fully automated presentation of the XML data received from the workflow at runtime.

5.2.2.4 Web Service Communication

This ABB represents the communication with a Web service, i.e. sending a request message to a Web service, i.e. invoking an operation of the Web service, and receiving the returned response. The Web Service Communication ABB can be used autonomously for the realization of system-oriented workflow activities, e.g. in order to retrieve data from a legacy system and evaluate it in a subsequent route node, thereby influencing the further control flow of the workflow. On the other hand, the ABB can be combined with other ABB's, thereby adding Web service communication facilities to them. Hence, for example, a workflow activity embodying the presentation of data which is retrieved via a Web service interface can thus be technically implemented by a combination of the Web Service Communication ABB and the Data Presentation ABB.

If required, the communication can be secured based on the WS-Security standard (Lawrence, Kaler, Nadalin et al. 2006) using encryption and digital signatures. The configuration of these security parameters can be achieved via the WS-Policy (Vedamuthu, Orchard, Hirsch et al. 2007) or WS-Security Policy (Lawrence, Kaler, Nadalin et al. 2007) standards respectively. According to a given configuration including at least the Web service endpoint URL, a reference to the Web service's interface description and the name of the operation to be called, the ABB's SBB generates a corresponding SOAP message, sends it to the Web service and receives the response.

With respect to the interaction between the Web Service Communication ABB's SBB with the workflow, the SBB can receive an XML document from the workflow and returns the response received from the Web service in form of an XML document back. The former XML document is optional and serves as basis for extracting values and integrating them as parameter values in the Web service request. Thus, both static and dynamic requests can be realized. The latter XML document does not contain the complete received SOAP message, but rather the unwrapped result as returned by the invoked Web service operation. Thus, it can directly be used in subsequent workflow activities.

Minimal physical configuration aspects:

- *Web Service Endpoint*: This property is used to specify the URL of the Web service endpoint to be called.
- *Web Service Interface Description*: A URL to the Web service's interface description in form of a WSDL document is supplied in this property.
- *Web Service Operation Name*: In this property, the name of the Web service operation to be invoked has to be provided.
- *Input Parameters Mapping Document*: If the operation to be called requires input parameters, this property can be used for specifying corresponding values. These values can either be static or be dynamically extracted from the XML document which is passed from the workflow to the SBB at runtime. Therefore, a mapping has to be defined which assigns to each parameter name as defined by the Web service interface either a static value or an XPath expression referencing a value from the above-mentioned input XML document. The complete mapping is specified in form of a simply-structured XML document and its URL is supplied in this property.
- *Security Policy Source*: If a secured Web service communication is required, this property serves for declaring whether the WS-SecurityPolicy-based specification is contained in the Web service's WSDL document or supplied via the subsequent configuration aspect.
- *Security Policy Document*: If the security policy specification which should be applied to the Web service communication is stored externally, this property allows for supplying a URL to a WS-SecurityPolicy-based specification document.

5.2.3 Extending XPDL towards Web-specific Concerns

The XPDL 2.0 standard provides basic constructs for specifying applications to be used for processing individual workflow activities. Therefore, it defines an *Application* construct and various application types as specializations of an abstract *ApplicationType* concept. However, the specification of the available specializations is rather basic and incomplete. This can be attributed to the fact that in this regard, XPDL has conceived itself only as rudimentary framework leaving the concrete specialization to tool-specific extensions. In order to assure universal validity, generic extensions which can be adopted across various modeling tools and notations are preferable though.

The presented catalog of Activity Building Blocks (ABBs) provides the foundation for extending the XPDL 2.0 standard towards coverage for such generic Web-specific application types. Thus, in the context of this thesis, the XPDL standard was extended in order to capture the presented ABBs and their physical configuration aspects. Therefore, the provided extension mechanisms of XPDL were employed. Figure 5-3 depicts a schematized overview of the relevant XPDL application types

Form, *WebService* and *Xslt* and their attributes as defined by the XPD L standard (indicated by the *xpdl:* namespace tag). In addition, the figure shows the extensions introduced by the Workflow DSL approach which enable the complete specification of fully functional Web-based Activity Building Blocks (indicated by the *wc:* namespace tag).

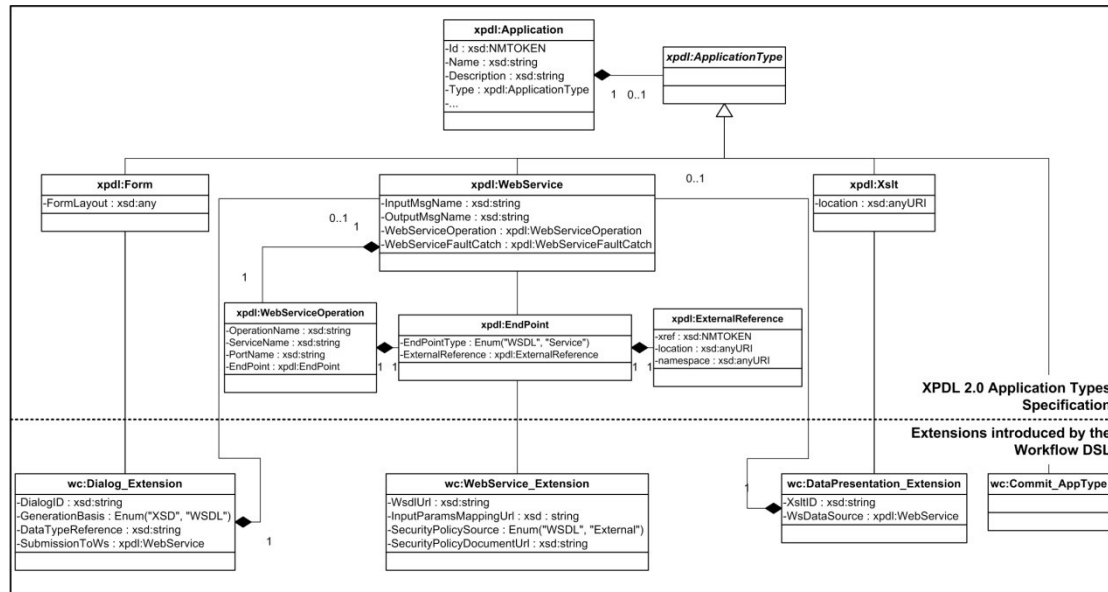


Figure 5-3: Schematized Overview of Relevant XPD L Application Types and the Web-Specific Extensions Introduced by the Workflow DSL

The performed extensions are twofold: On the one hand, a new application type specialization named *wc:Commit_AppType* representing the Commit ABB was introduced. On the other hand, in order to provide full specification coverage for the ABBs *Dialog-based User Interaction*, *Web Service Communication*, and *Data Presentation*, extensions to the existing XPD L application types *Form*, *WebService* and *Xslt* were defined. In doing so, existing XPD L types and attributes were reused where possible. Consequently, the *wc:Dialog_Extension* refers via its attribute *DataTypeReference* to the identifier of a *xpdl:TypeDeclaration* element which in turn specifies the location of the XML Schema and a type selector which were defined as required attributes for the respective ABB in the previous section. Similarly, for the case of a WSDL-based dialog generation and Web service submission of the filled dialogs, the *xpdl:WebService* type and the corresponding *wc:WebService_Extension* are employed to specify the configuration aspects required by the *Dialog-based User Interaction* and *Web Service Communication* ABBs.

The resulting extended XML Schema specification for XPD L serves as *Domain-Specific Model (DSM)* for the Workflow DSL. Based thereupon, the modeling of Web-based workflows can be fully supported both in the *Business Process Modeling* phase and the *Workflow Modeling* phase. The integrated concepts and attributes are sufficient for generating a fully-operational Web-based workflow. The detailed specification of ABBs in the *Physical Design & Execution* phase is performed on the basis of the ABB's individual DSMs which represent a superset of the specification aspects integrated in the Workflow DSL's DSM. In order to still achieve permanent model consistency and

to support smooth roundtrip engineering, these full specifications are referenced within a Workflow DSL program. Thus, the detailed design performed in the *Physical Design & Execution* phase is preserved throughout subsequent evolution cycles.

Due to the specification of the Workflow DSL's DSM in form of an XML Schema document, concrete specifications of Web-based workflows, i.e. Workflow DSL programs, are hence XML documents. For a better understanding of their structure and elements as well as the interplay of the XPDL standard and the introduced extensions, some excerpts are presented. They belong to a Workflow DSL program corresponding to the 'business trip' example process depicted in Figure 2-1 and illustrate particularly the realization of the *Create Expense Report* activity.

Figure 5-4 shows an excerpt containing an *xpdl:TypeDeclaration* for the *ExpenseReportType* based on an external XML Schema document as well as a workflow variable named *ExpenseReport* in form of a *xpdl:DataField* for storing an instance of this type.

```

1 <xpdl:TypeDeclarations>
2   <xpdl:TypeDeclaration Id="ExpenseReportType">
3     <xpdl:ExternalReference location="http://ExpenseReportType.xsd" />
4   </xpdl:TypeDeclaration>
5 </xpdl:TypeDeclarations>
6
7 <xpdl:DataField Id="ExpenseReport">
8   <xpdl:DataType>
9     <xpdl:DeclaredType Id="ExpenseReportType" />
10  </xpdl:DataType>
11 </xpdl:DataField>

```

Figure 5-4: XPDL Type Declaration and Data Field Specification within a Workflow DSL Program

The XPDL-based specification of the *CreateExpenseReport* workflow *Activity* within the Workflow DSL program is depicted in Figure 5-5. It refers to a *TaskApplication* definition named *CreateExpenseReport_App* which embodies the activity's execution at runtime and is described in the next paragraph. Furthermore, a mapping of the application's input and output parameters to workflow variables is defined. In this example, the input parameter is left empty and the value returned from the application is stored in the above-mentioned *ExpenseReport* data field. Finally, the role *Employee* is defined as eligible performer for this activity.

```

1 <xpdl:Activity Id="CreateExpenseReport" Name="CreateExpenseReport">
2   <xpdl:Implementation>
3     <xpdl:Task>
4       <xpdl:TaskApplication Id="CreateExpenseReport_App">
5         <xpdl:ActualParameters>
6           <xpdl:ActualParameter />
7           <xpdl:ActualParameterExpenseReport</xpdl:ActualParameter>
8         </xpdl:ActualParameters>
9       </xpdl:TaskApplication>
10    </xpdl:Task>

```



```

11 </xpdل:Implementation>
12 <xpdل:Performer>Employee</xpdل:Performer>
13 </xpdل:Activity>

```

Figure 5-5: Workflow Activity Definition within a Workflow DSL Program

The most interesting aspect with respect to the introduced Web-specific ABBs and corresponding extensions to the XPDЛ standard is illustrated in Figure 5-6. The excerpt shows an XPDЛ *Application* definition named *CreateExpenseReport_App* which is referenced as *TaskApplication* by the above *CreateExpenseReport* Activity definition. As the workflow activity for which this application shall be used consists in an employee filling out her expense report, the ABB *Dialog-based User Interaction* presents the adequate choice. As shown in Figure 5-3, the introduced extensions related to this ABB are based on the XPDЛ application type *Form*. Hence, in the application definition, at first the application type *xpdл:Form* is declared. Thereupon, the further specification based on the introduced extension container element *wc:Dialog_Extension* follows.

In this example, the dialog shall be generated based on an XML Schema definition (*wc:GenerationBasis*). The corresponding schema was defined in the context of a *xpdл:TypeDeclaration* (cf. Figure 5-4) whose identifier *ExpenseReportType* is referenced via the element *wc:DataTypeReference*. Following the specification of the *Dialog-based User Interaction* ABB, these two statements already suffice as minimum configuration set for obtaining a fully-operational Web-based dialog.

The second part of the application definition addresses the application's interface description enclosed by the *xpdл:FormalParameters* tag. As specified by the ABB, an input and an output parameter of the *ExpenseReportType* type are defined. The former can be used to provide a data instance for the initialization of the dialog whereas the latter serves for returning the filled dialog. These parameter declarations are used by activity definitions (cf. Figure 5-5) to pass and receive values to and from the application. Hence, it is quite intuitive to reference the *ExpenseReportType* also as generation basis (*wc:DataTypeReference*) for the dialog itself.

```

1 <xpdл:Application Id="CreateExpenseReport_App"
  xmlns:wc="http://www.wsls.net/2006/04/workflow.XPDЛ2"
2 <xpdл:Type>
3 <xpdл:Form>
4 <wc:WebCompositionExtensions />
5 <wc:Dialog_Extension>
6 <wc:GenerationBasis>XSD</wc:GenerationBasis>
7 <wc:DataTypeReference>ExpenseReportType</wc:DataTypeReference>
8 </wc:Dialog_Extension>
9 </xpdл:Form>
10 </xpdл:Type>
11 <xpdл:FormalParameters>
12 <xpdл:FormalParameter Id="Send" Name="Send" Mode="IN">
13 <xpdл:DataType>
14 <xpdл:DeclaredType Id="ExpenseReportType" />
15 </xpdл:DataType>

```

```
16     </xpdl:FormalParameter>
17     <xpdl:FormalParameter Id="Receive" Name="Receive" Mode="OUT">
18         <xpdl:DataType>
19             <xpdl:DeclaredType Id="ExpenseReportType" />
20         </xpdl:DataType>
21     </xpdl:FormalParameter>
22 </xpdl:FormalParameters>
23 </xpdl:Application>
```

Figure 5-6: Application Definition for a Web-based Expense Report Dialog within a Workflow DSL Program

5.3 The DIMs – Multi-Notational Modeling with Stakeholders

Enabling stakeholders to use the modeling notation and tools they already know and thereby lowering the threshold for effective collaboration is particularly in the business process modeling domain of great interest (cf. Section 2.2.2). Furthermore, the requirement of continuity, i.e. preserving one shared workflow specification across various notations and throughout the development lifecycle, from initial requirements engineering to the fully-operational workflow-based Web application, exists (Havey 2006).

According to the Web Engineering DSL Framework, various modeling notations for different stakeholder groups and levels of detail can be defined on top of a DSL's DSM. Consequently, in the context of this thesis, multiple business process modeling notations were integrated as *Domain Interaction Models (DIMs)* for the Workflow DSL. Thus, stakeholders can collaborate in the specification of a Web-based workflow, i.e. a Workflow DSL program, by using the business process modeling notation and tool they know best. Thereby, throughout all phases and across all DIMs, one shared DSL program is edited. Thus, problems concerning model consistency as well as misunderstandings or mistakes occurring in the context of manual translation activities can be minimized.

Figure 5-2 illustrates the Workflow DSL's multi-notational modeling approach. It allows the incremental, model-based specification of Workflow DSL programs from initial requirements engineering to business process modeling to workflow modeling and enables stakeholders to use their preferred notations and tools. In the context of this thesis, four DIMs and corresponding tools were exemplarily integrated. For initial requirement engineering activities, a custom, list-based notation termed *Simple Sequence Only (SSO)* was developed and incorporated with Microsoft Word 2007 as editor. The phases *Business Process Modeling* and *Workflow Modeling* from the Workflow DSL's process model are supported by three additional standard business process modeling notations and supplemental tools: The Business Process Modeling Notation (BPMN) and Microsoft Visio, UML 2.0 Activity Diagrams and IBM Rational Software Architect, as well as Petri Nets and INCOME2010.

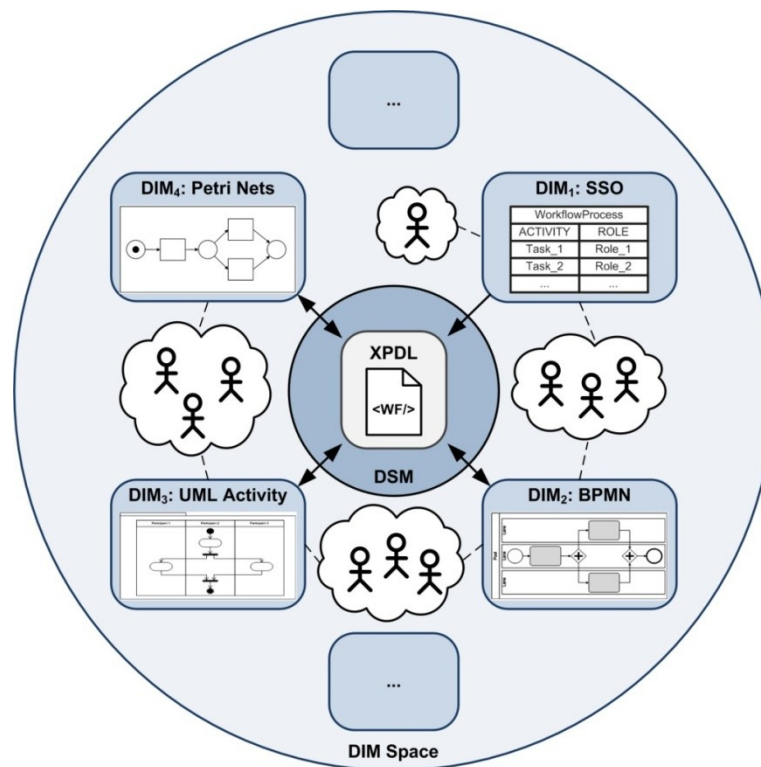


Figure 5-7: Multi-Notational Modeling of a Shared Workflow DSL Program

The model transformation framework presented in Section 5.4 realizes lossless model transformations between all DIM notations and tools on the one hand and the shared Workflow DSL program on the other hand. According to the Web Engineering DSL Framework and the open conception of the model transformation framework, new DIMs can be easily integrated.

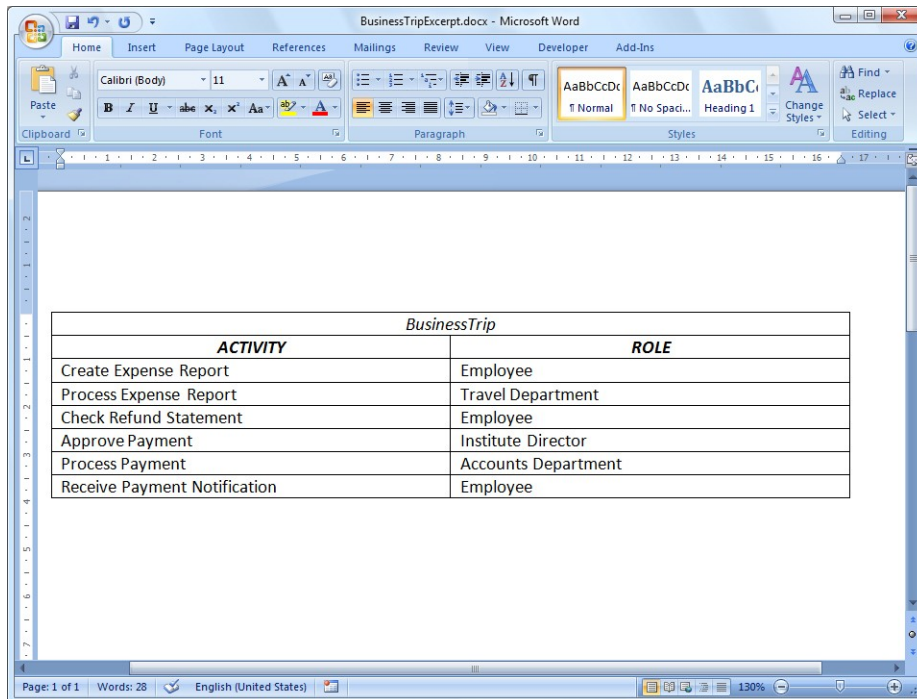
Especially the automated transformation from a very intuitive but coarse-grained DIM notation into a more fine-grained notation, i.e. for example from SSO to BPMN or Petri Nets, helps stakeholders in learning the semantics and symbols. This holds particularly true if stakeholders themselves developed a process model using the former and immediately can obtain a semantically equivalent model in another notation. The backward direction, i.e. from BPMN, UML or Petri Nets back to SSO, is reasonably not intended. This is due to the fact that the former notations cover a much broader part of the DSM than the SSO DIM does. Thus, a Workflow DSL program containing more advanced (control flow-related) constructs than sole tasks and roles cannot be rendered in the SSO notation in a meaningful way. This would rather confuse stakeholders instead of presenting an actual facilitation. As the other three DIMs offer sufficient expressiveness, bilateral transformations are fully supported, independently from the Workflow DSL program's complexity and level of detail.

In the following subsections, the incremental specification of the 'business trip' example process (cf. Chapter 2, Figure 2-1) will be exemplarily described using the various DIM notations and tools. The supporting model transformation framework is subsequently presented in detail in Section 5.4. A complete overview of all DIM elements and their mapping on the DSM concepts can be found in Section 5.4.5.

5.3.1 Simple Sequence Only (SSO) with Microsoft Word

The Simple Sequence Only (SSO) DIM is a custom notation which was designed in order to enable stakeholders without business process modeling skills to autonomously collaborate in initial requirements engineering activities. This intention also resulted in the selection of a word processor, namely Microsoft Word 2007 which is assumed to be available to the great majority of stakeholders, as DIM editor. In addition, the notation shall contribute to the goal of model continuity by extending the model-based development process to these early activities and related specification documents. As SSO is a regular DIM of the Workflow DSL, manual redrawing or translation of initial elicitation documents into more advanced modeling languages can be substituted by automated model transformations.

The SSO DIM consists in a table-based notation as illustrated in Figure 5-8. The table comprises two-columns, whereby the first column serves for listing process activities and the second column contains the respective role. By adding rows, an arbitrary amount of activities and corresponding roles can be easily captured. In order to facilitate the usage of the SSO DIM, a corresponding Word document template containing an initial table with some default values was developed. Thus, stakeholders can start process modeling by creating a new Word document based on the SSO document template and filling out the table.



<i>BusinessTrip</i>	
ACTIVITY	ROLE
Create Expense Report	Employee
Process Expense Report	Travel Department
Check Refund Statement	Employee
Approve Payment	Institute Director
Process Payment	Accounts Department
Receive Payment Notification	Employee

Figure 5-8: Initial Draft of the Business Process using the Simple Sequence Only (SSO) DIM and Microsoft Word 2007

The saved Word 2007 document, which is based on the Office Open XML standard (Ecma International 2006; ISO/IEC 29500:2008 2008), directly serves as source document for model transformations as presented in the next section. Thus, the so

far specified business process can immediately be translated and further elaborated using more advanced DIM notations like BPMN, UML Activity Diagrams or Petri Nets.

Obviously, the coverage of the Workflow DSL's DSM by the DIM notation is rather low as it only provides modeling constructs for the domain concepts *workflow process*, *start node*, *end node*, *sequence flow*, *activity* and *participant* (cf. Section 5.4.5). However, for its intended use, e.g. the gathering of activities during early elicitation sessions where the focus lies exactly on these concepts, the SSO DIM is sufficient. The incorporation of further concepts, e.g. parallel activity flows or XOR routes, represented by indentation, use of key words, cell merging etc. was investigated (Setiawan 2009). However, the evaluation arrived at the conclusion that this would lead to ambiguous and rather unintuitive visual representations. As a result, the original intention and strong focus of simplicity would get lost. Beyond that, the SSO DIM should not be considered as an alternative but rather as a reasonable complement to the existing multitude of business process modeling notations.

5.3.2 Business Process Modeling Notation (BPMN) with Microsoft Visio

This DIM allows stakeholders to specify Workflow DSL programs based on the Business Process Modeling Notation (BPMN) standard (White 2006) and Microsoft Visio. Visio is part of the Microsoft Office product line and supports a broad audience of office workers in creating diverse diagram types from different fields (e.g. business, engineering, facility management, project management, software development etc.). According to a recent study, Visio is also the most commonly used software for BPMN modeling (Recker 2008), even though it does not natively include BPMN shapes in its release version.

The BPMN-based DIM notation was defined in a rather straight-forward way since XPDL, the standard on which the Workflow DSL's DSM is based on, was originally intended as XML-based serialization format for the solely visual BPMN symbols. Thus, the mapping rules between the DSM concepts and the BPMN DIM symbols could be derived from the XPDL specification (Shapiro, Marin, Brunt et al. 2005) and are summarized in Section 5.4.5.

Concerning the incorporation of Microsoft Visio as DIM editor, a so-called Visio *stencil* in accordance with the BPMN standard, i.e. a collection of graphical shapes representing the various BPMN symbols, was developed. Therefore, the graphical symbols defined by BPMN were redrawn with Visio's native drawing tools, named according to the BPMN standard, and saved into the BPMN stencil. Visio allows the definition of *shape data* for a shape, i.e. a set of properties that can be configured individually for each shape instance. This feature was used for enriching the purely visual BPMN shapes by configuration properties in accordance with the Workflow DSL's DSM. Besides few properties required by the XPDL standard, this concerns primarily the Web-specific *physical configuration aspects* which were introduced in the previous section. Based on the resulting BPMN stencil, stakeholders can visually

compose Workflow DSL programs in form of BPMN diagrams in the *Business Process Modeling* phase and configure execution-relevant, Web-specific concerns, i.e. the *Concern Configuration*, in the *Workflow Design* phase.

Figure 5-9 depicts a screenshot of BPMN-based modeling with Microsoft Visio. In the figure, the left panel contains the BPMN 1.0 stencil. On its right, the main drawing area containing the BPMN diagram adjoins. Shapes from the stencil can be dragged and dropped on the drawing area, whereby an instance of the shape is created. Below the drawing area, the *shape data* panel is located. It allows the configuration of instance-specific properties as defined by the type of the currently selected shape.

In the running example, the initial SSO-based draft of business process activities and associated roles as shown in Figure 5-8 was transformed into its BPMN DIM representation. The result is depicted in Figure 5-9 and was achieved by fully automated model transformations which are presented in the next section. The diagram exactly reflects the tasks and roles modeled with the SSO DIM; only the representation of the Workflow DSL program has changed to the BPMN DIM.

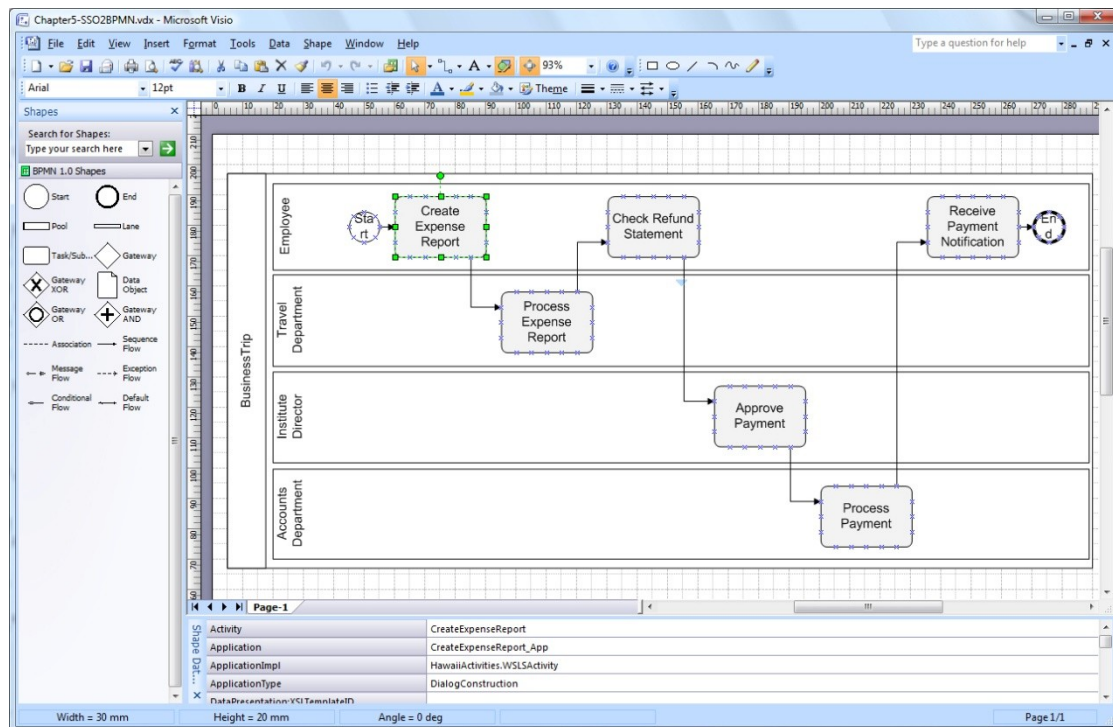


Figure 5-9: Transformed BPMN Representation of Workflow DSL Program

Stakeholders familiar with BPMN and Microsoft Visio can now further refine, extend and modify the business process or Workflow DSL program respectively, i.e. insert activities, add participants, specify the control flow as well as modify or remove existing constructs. Similarly, in the *Workflow Modeling* phase, they can specify the *Concern Configuration* by configuring the shape-specific properties in the *shape data* panel. In the example, the business process was extended with respect to the case that an employee has an objection when checking the refund statement. Therefore, a new activity *Check Objection*, a *XOR-based Split Gateway* and a corresponding *XOR-based Join Gateway* as well as a *Conditional Flow*, a *Default Flow* and two *Sequence*

Flows were inserted. As a result, if the employee indicated in the *Check Refund Statement* activity that she has an objection to the refund statement, the workflow links to the new *Check Objection* activity which is assigned to the *Travel Department* role. Otherwise, the workflow directly links to the *Approve Payment* activity. Figure 5-10 illustrates the inserted process section which is indicated by the dotted line. It also highlights the mapping of the added modeling elements in the BPMN DIM representation to corresponding XML excerpts of the Workflow DSL program which has to be realized by adequate model transformations. The figure also again conveys the *Web Engineering DSL Framework's* general concept of editing a single (Workflow) DSL program using various DIM notations - assuming that adequate model transformations between DIM notations and the DSM exist. Which concrete graphical symbols represent the generic concepts of the DSL's respective domain, e.g. whether a XOR gateway is represented like in BPMN by a rhombus with an 'X' in it or by a completely different symbol, remains only a question of adequate model transformations.

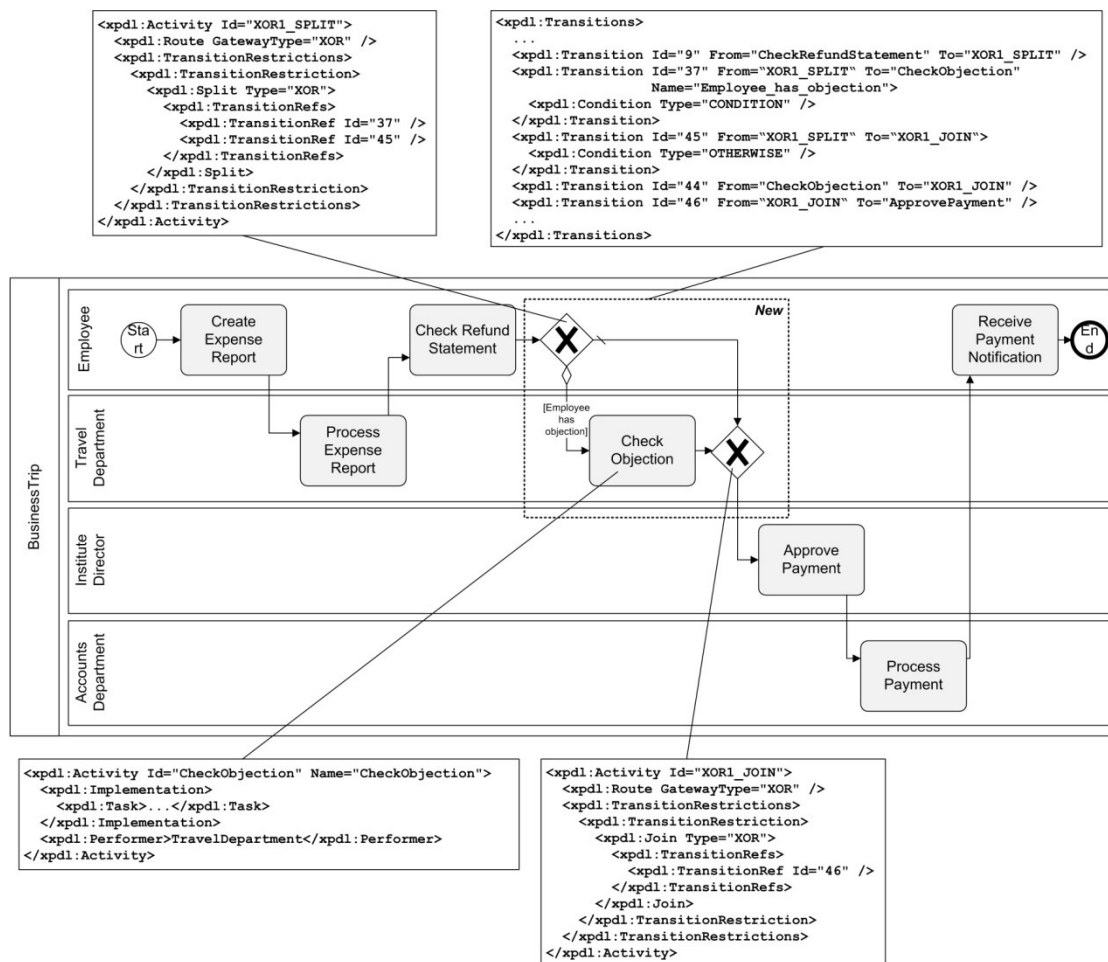


Figure 5-10: Added Process Section in the BPMN DIM Representation and the Corresponding Workflow DSL Program's XML Representation

In this regard, Microsoft Visio provides export and import facilities for an XML-based exchange format named *DatadiagramML* (Microsoft Corp. 2006b). Based thereupon, bilateral model transformations between the BPMN DIM or the *DatadiagramML*

format respectively and the Workflow DSL's DSM were developed (cf. Section 5.4). Thus, with the Workflow DSL's DSM as intermediary format and similar transformations for other DIMs, a Workflow DSL program in its BPMN DIM representation can be transformed into arbitrary DIM representations.

Models created with the BPMN DIM have to comply with some basic modeling rules which assure their well-formedness and, related to that, the correct function of associated model transformations. This has also been identified as a general problem in the business process modeling and transformation field (Murzek and Kramler 2007). The focus of the guidelines lies primarily on enforcing an explicit, formalizable and block-structured, i.e. a well-formed nesting of control flow constructs, modeling approach. Regarding the latter, the XPDL standard as well as most of today's business process modeling notations support a less constraining graph-oriented model structure, whereas a block-oriented structure is inherent to the great majority of today's technical workflow execution languages like BPEL (Jordan and Evdemon 2007) or XOML (Microsoft Corp. 2007). A complete overview of these rules can be found in (Orozov 2008). The evaluation of real-world business process models presented in Section 8.1 showed that these guidelines present no significant restriction in practice. It turned out that, in the most cases, missing adherence can be recovered by simple pattern-based remodeling.

5.3.3 UML Activity Diagrams with IBM Rational Software Architect

The Unified Modeling Language (UML) (Object Management Group 2005b) is increasingly seen as de-facto standard for software modeling and design. In consideration of this level of familiarity among a developer-oriented stakeholder audience, the Workflow DSL consequently provides a DIM notation which allows the use of the UML 2.0 Activity Diagram notation for specifying Workflow DSL programs. Their applicability to the business process modeling domain in a general sense is not immediately evident and was examined in a comprehensive evaluation (Russell, van der Aalst, ter Hofstede et al. 2006). Similar to other business process modeling languages, UML was confirmed sufficient coverage particularly for the control-flow and data perspective, whereas limitations for other perspectives were identified.

Consequently, in order to leverage UML 2.0 Activity Diagrams as DIM for the Workflow DSL, a *UML 2.0 Profile* defining the required extensions was developed and is illustrated in Figure 5-11. A UML Profile is a semantically cohesive group of stereotypes, constraints and tagged values which extends the UML metamodel for a particular modeling domain (Pender 2003). Within UML Profiles, *stereotypes* play an important role. A stereotype is a type which can be applied to any UML element in order to extend its properties and slightly alter its semantics. By applying a stereotype to an UML element, the attributes defined by the stereotype become available for the UML element.

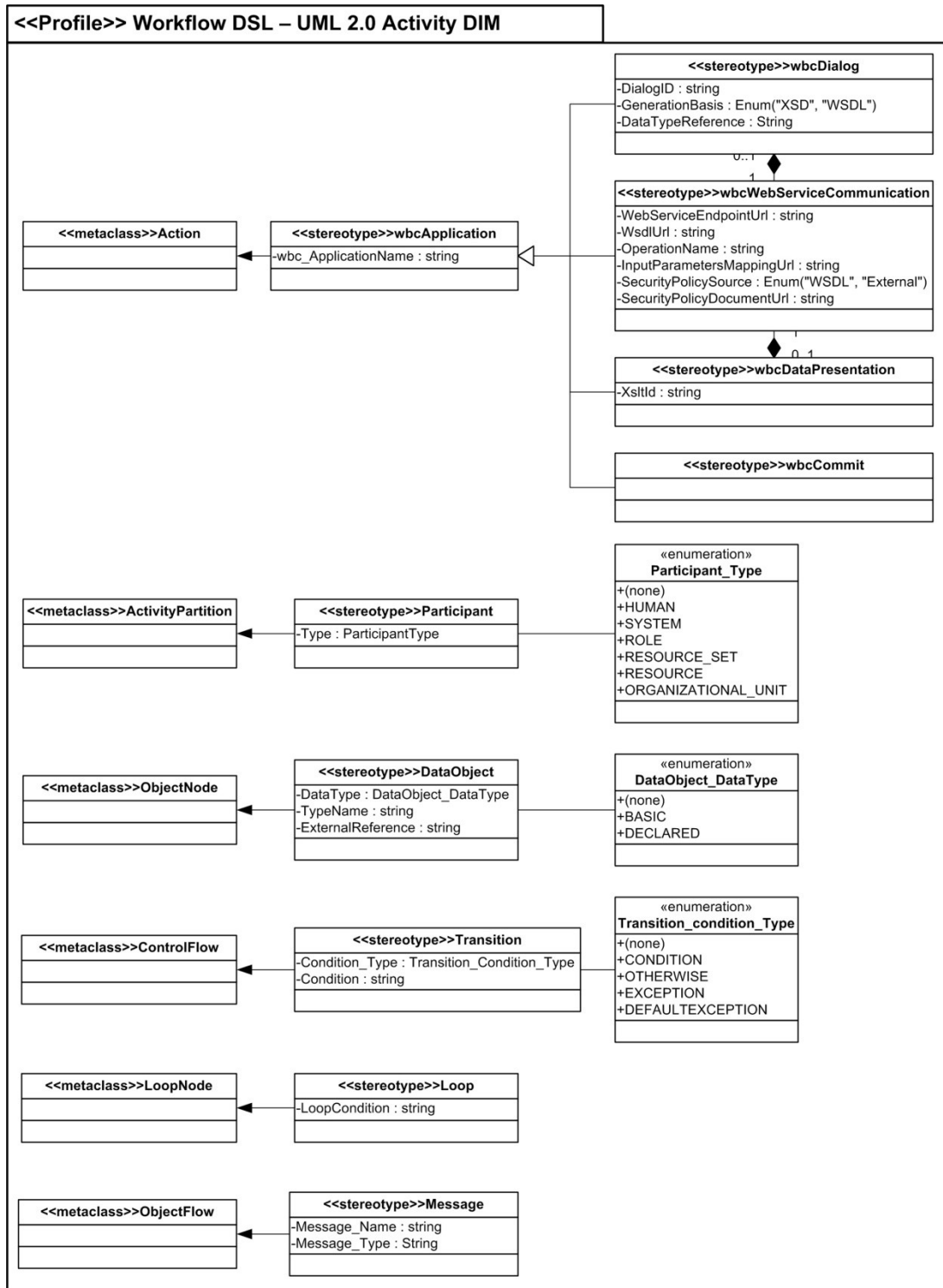


Figure 5-11: UML 2.0 Profile for the Workflow DSL’s UML 2.0 Activity DIM

Within the presented UML Profile, the classes in the left column and marked with the *metaclass* tag present elements from the UML 2.0 Activity Diagram specification. As such, they present the linking points for the introduced extensions. Correspondingly, the associated classes marked with *stereotype* tags represent the introduced extensions. They augment the UML 2.0 Activity Diagram specification by concepts required for achieving a full coverage of the Workflow DSL’s DSM and thus

for specifying Web-based workflows respectively. The introduced extensions are two-fold: On the one hand, the stereotypes *wbcApplication*, *wbcDialog*, *wbcDataPresentation*, *wbcWebServiceCommunication*, and *wbcCommit* support the modeling of Web-specific concerns and applications as described in the context of the Workflow DSL's DSM in Section 5.2.3. On the other hand, extensions for specifying more general workflow concerns in accordance with the Workflow DSL's DSM or the XPD standard respectively are embodied by the stereotypes *Participant*, *DataObject*, *Transition*, *Loop*, and *Message*. An overview of the complete mapping between the concepts of the Workflow DSL's DSM and the UML 2.0 Activity DIM is given in Section 5.4.5.

Due to the specification of the extensions in form of the presented UML Profile, all standard UML tools which support UML profiles, i.e. UML's native extension mechanisms, can be employed as model editors for the UML 2.0 Activity DIM. To this end, the XML Metadata Interchange (XMI) serves as XML-based interchange format across the great variety of today's UML modeling tools, both for exchanging UML Profiles and individual UML models (Object Management Group 2007). Regarding the latter, XMI captures both model information and diagrammatic information, e.g. symbols, layout, colors and fonts. Beyond that, the XMI format presents also the basis for the bilateral model transformations between the Workflow DSL's DSM and the UML 2.0 Activity Diagram DIM which are presented in the next section.

With respect to the running example, the 'business trip' process, the Workflow DSL program in its BPMN DIM notation (cf. Figure 5-10) was transformed by means of these model transformations into its UML 2.0 Activity DIM representation as depicted in Figure 5-12.

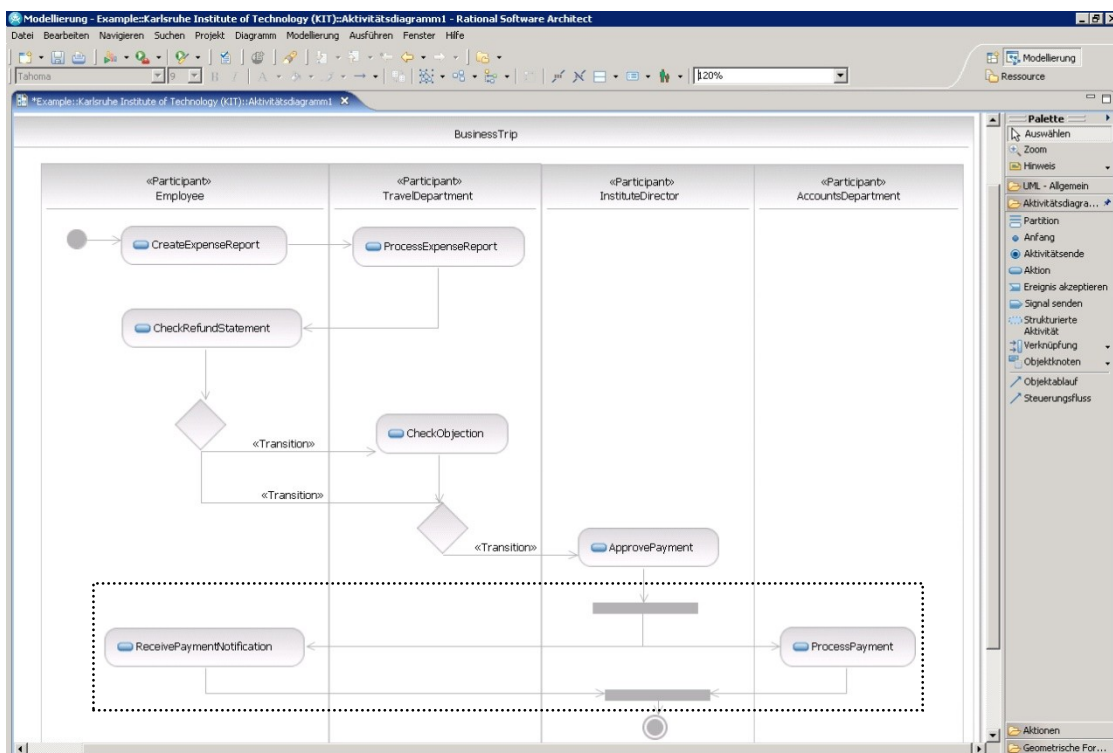


Figure 5-12: Extending the Business Process using the UML 2.0 Activity DIM and IBM Rational Software Architect

Using the UML 2.0 Activity DIM, an *AND*-based Split *Gateway* and a corresponding *AND*-based Join *Gateway*, enclosing the process activities *Receive Payment Notification* and *Process Payment*, were inserted (as indicated by the dotted rectangle). The UML Activity Diagram representations of these gateway DSM concepts are a *Fork Node* and a *Join Node* respectively (cf. the complete mapping overview in Section 5.4.5). Thereby, the parallel execution of these two activities is specified. Beyond that, Figure 5-12 also illustrates the use of the introduced stereotypes *Participant* and *Transition*. Their application to particular elements of the UML 2.0 Activity Diagram is indicated by displaying the stereotype's name enclosed by guillemets next to the respective element.

The specification of the *Concern Configuration* in the *Workflow Modeling* phase can similarly be performed by applying stereotypes and configuring their respective properties. For example, the *wbcApplication* and *wbcDialog* stereotypes could be applied to the *CreateExpenseReport* activity and subsequently be configured with regard to their properties, thereby assigning and configuring an instance of the *Dialog-based User Interaction ABB* to the activity. In the context of this thesis, IBM Rational Software Architect 7.0 (IBM RSA) was used as modeling tool for this DIM. Detailed instructions on constructing Web-based workflows using the UML 2.0 Activity DIM and IBM RSA can be found in the diploma thesis of Denny Setiawan (Setiawan 2008).

The XML excerpts corresponding to the graphical UML *Fork* and *Join Nodes*, to the *Process Payment Action Node* as well as to the *Control Flow* between the *Fork Node* and this *Action Node* are depicted in Figure 5-13.

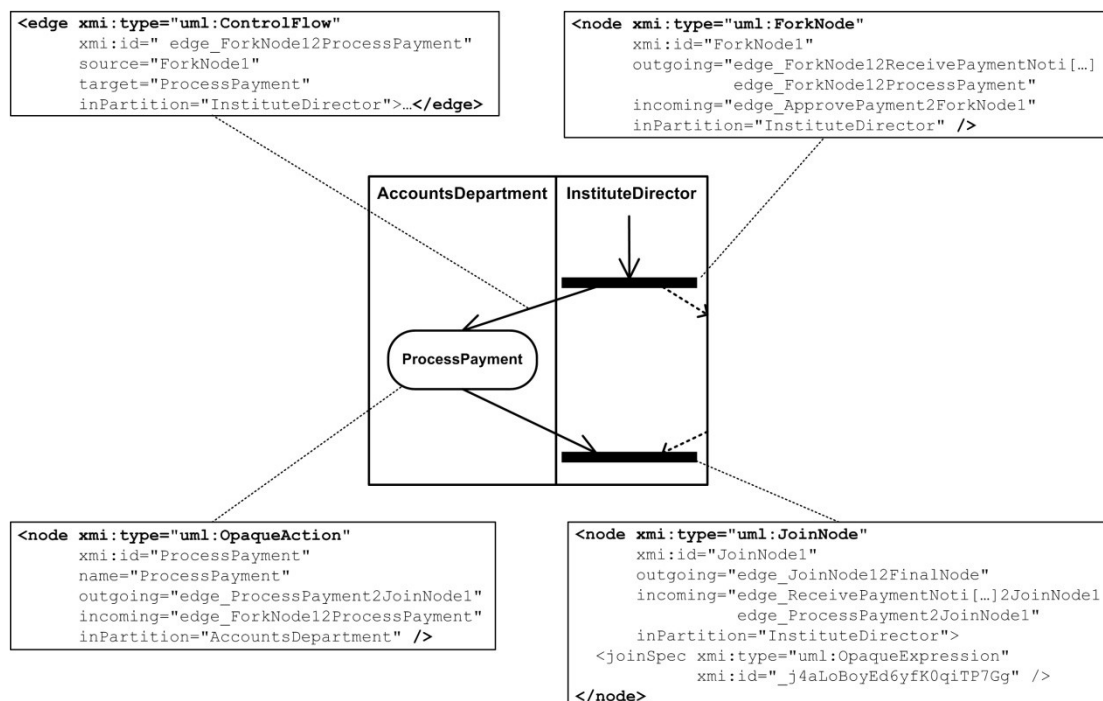


Figure 5-13: Excerpts from the XML-based Serialization corresponding to the Business Process Section Added using the UML 2.0 Activity DIM

On the one hand, as mentioned above, such an XMI-based serialization of a UML 2.0 Activity Diagram can be retrieved via export facilities available in the great majority of UML modeling tools. On the other hand, the XMI markup presents the basis for model transformations between the UML 2.0 Activity DIM and the DSM. The mapping of UML shapes or XMI excerpts respectively to the DSM-based Workflow DSL program is similar to Figure 5-10. An *xpdl:Transition* represents the UML *Control Flow*, two *xpdl:Activity* elements refined as *AND Gateway*-typed *xpdl:Route* stand for the UML *ForkNode* and *JoinNode*, and a *xpdl:Activity* refined as *xpdl:Task* corresponds to the UML *OpaqueAction*.

5.3.4 Petri Nets with INCOME2010

Petri nets were originally invented by Carl Adam Petri as a formal graphical modeling language for the description and analysis of concurrent processes in distributed systems (Petri 1962). To date, due to their formal semantics, graphical nature, high expressiveness, analyzability and vendor-independence, Petri nets have been widely adopted for the domain of business process modeling (Van der Aalst 1998; Klink, Li and Oberweis 2008). Beyond that, Petri nets are also considered as an ideal foundation for analyzing and even mining process models (Van der Aalst 2007). Consequently, the Workflow DSL includes also a Petri net-based DIM allowing stakeholders to specify Web-based workflows based on the Petri net notation.

In contrast to other business process modeling notations, the graphical visualization of Petri nets comprises only three different symbols: *Places*, *transitions* and *directed arcs*. The latter connect places and transitions whereby the direction of the arc indicates whether a place presents a pre- or post-condition for the transition. Consequently, the modeling of common business process constructs has to be performed on a pattern-basis instead of a symbol-basis. For example, while BPMN and UML 2.0 Activity Diagrams individually define particular graphical symbols for typical control-flow constructs like parallel routing, conditional routing or iterative routing, their counterparts in a Petri net notation consist in composite patterns of the above-mentioned three core symbols. Section 5.4.3 shows the complete mapping between the workflow concepts defined by the Workflow DSL's DSM and their Petri net representation.

With respect to enabling model interchange for Petri nets across the variety of existing modeling tools, the XML-based Petri Net Markup Language (PNML) was introduced and is currently in the course of becoming an ISO/EIC standard (Kindler 2006; Kindler 2007). PNML captures both structural and graphical information of a Petri net and provides concepts for tool-specific extensions. These extension mechanisms were used to leverage Petri nets as a DIM for the Workflow DSL. Hence, in order to achieve a full coverage of the Workflow DSL's DSM, dedicated extensions were introduced. The required extensions are reasoned by the DSM concepts *xpdl:Participant*, *xpdl:Application*, *xpdl:Transition*, *xpdl:TypeDeclaration* and *xpdl:DataField*. While the extensions associated to *xpdl:Transition* target the PNML concept *Arc*, all other extensions were assigned to the PNML concept *Transition*.

The introduced extensions are congruent to the extensions presented for the UML 2.0 Activity DIM (cf. Figure 5-11). However, some minor additional aspects, which were natively available in UML 2.0 Activity Diagrams, but were missing in Petri nets or PNML respectively, had to be incorporated: Firstly, the ability to assign roles (*xpdl:Participant*, *xpdl:Performer*) to Petri net *Transitions* had to be introduced. Petri nets originally define no concept for this. Secondly, as Petri nets have no counterpart for the UML *Object Node* or *xpdl:DataField* respectively, the possibility to define data objects was integrated. In the context of this thesis, the XML-based PNML format is not only considered as interchange format for various modeling tools but also serves as basis for bilateral model transformations between the Workflow DSL's Petri Net DIM and the DSM (cf. Section 5.4) Thus, via the DSM as intermediate schema, Web-based workflows can be transformed from arbitrary DIM notations into the Petri Net DIM notation and vice versa.

In the context of this thesis, the Petri net modeling tool INCOME2010 v.0.2.3 was adopted as model editor for the Workflow DSL's Petri Net DIM (Klink, Li and Oberweis 2008; Oberweis 2008). INCOME2010 is still under development and currently comprises no generic extension mechanisms yet. However, the tool's full source code is available under the Eclipse Public License (EPL) whereby the required extensions could be integrated on a code-basis. In this way, extensions to INCOME2010's user interface as well as to the PNML-based diagram import and export were developed. In this regard, existing configuration dialogs were extended in order to support the configuration of Petri net *transitions* and *arcs* according to the extensions described above. Furthermore, the integration and extraction of these configuration properties in the context of the PNML-based export and import features was implemented. In this context, the extensions to PNML were also formalized in form of an XML Schema (Setiawan 2009). In summary, by using the extended version of INCOME2010, stakeholders can employ Petri nets as DIM notation to conveniently model Web-based workflows.

In the running example of the 'business trip' workflow, the Petri Net DIM was used in the *Workflow Modeling* phase for specifying the *Concern Configuration*, i.e. the set of workflow execution-relevant concerns bridging the gap between a business process and its technical realization in form of a Web-based workflow. Figure 5-14 shows the 'business trip' process model in its Petri Net DIM representation and a configuration dialog for the *CreateExpenseReport* activity within INCOME2010. This configuration dialog for Petri net transitions was extended by the sections *Application*, *Data Object*, and *Role*, whereby each section comprises dedicated properties allowing the configuration of these aspects. The *Application* section, for example, was designed according to the minimal configuration sets required by the Web-specific Activity Building Blocks (ABBs) which are part of the Workflow DSL's DSM (cf. Figure 5-3). Thus, as shown in the figure above, the dialog allows selecting an ABB type and entering the corresponding required minimal configuration.

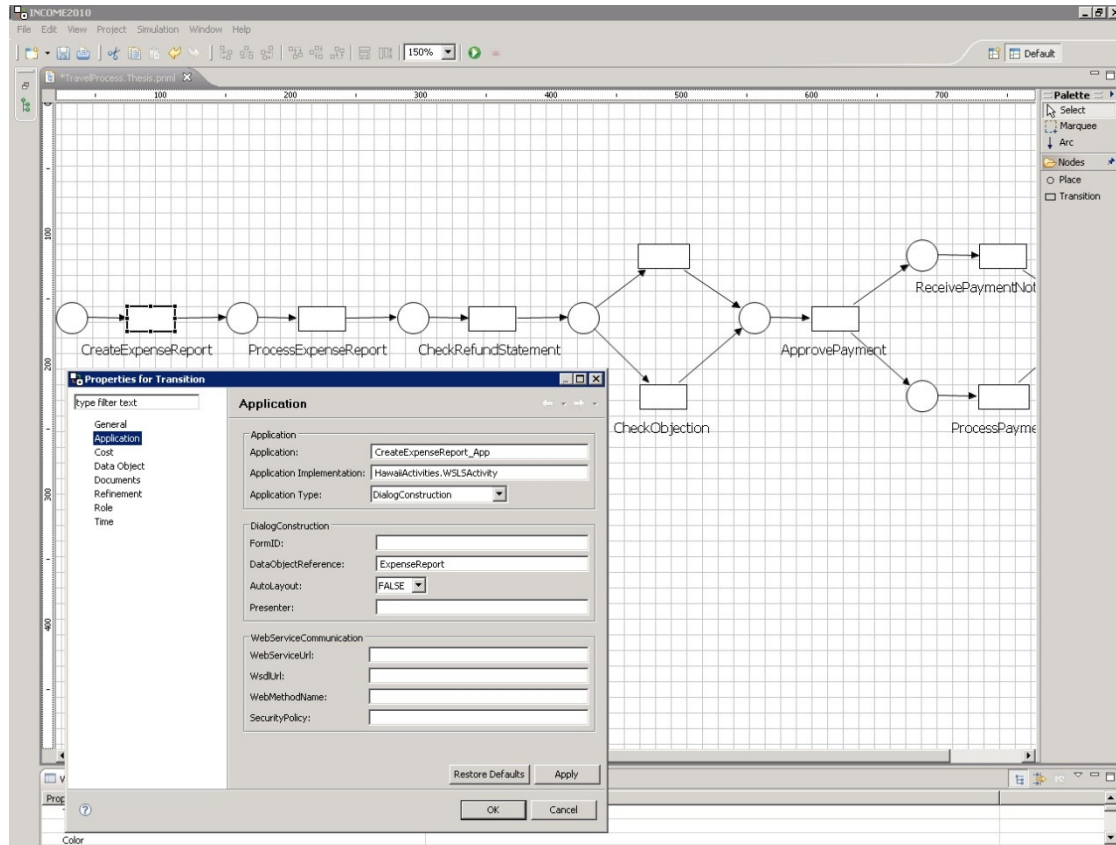


Figure 5-14: Workflow Modeling using the Petri Nets DIM and INCOME 2010

In the example, the *Application Type* “Dialog Construction” corresponding to the *Dialog-based User Interaction ABB*, was selected from the drop-down list. Furthermore, the *ExpenseReport Data Object* was referenced as XSD-based generation basis for the dialog. The resulting PNML representation of this application configuration is shown in lines 6-12 of Figure 5-15, whereby lines 8-11 contain the Dialog ABB-specific information. By means of adequate model transformations between the Workflow DSL’s Petri Net DIM and the DSM, this excerpt is translated into a DSM-based *xpdl:Application* definition as shown in Figure 5-6.

```

1 <transition id="CreateExpenseReport">[...]
2   <toolspecific tool="INCOME2010_Extension" version="v 0.2.3_Extension"
3     xmlns:wbc="http://www.wsls.net/2006/04/workflow.XPD2"
4     <wbc:swimlane>[...]</wbc:swimlane>
5     <wbc:dataObject>[...]</wbc:dataObject>
6     <wbc:wbc xmlns:wbc="http://www.wsls.net/2006/04/workflow.XPD2"
7       <wbc:application>CreateExpenseReport_App</wbc:application>[...]
8       <wbc:WebComp_Dialog_Extension>
9         <wbc:GenerationBasis>XSD</wbc:GenerationBasis>
10        <wbc:DataTypeReference>ExpenseReportType</wbc:DataTypeReference>
11      </wbc:WebComp_Dialog_Extension>
12    </wbc:wbc>
13  </toolspecific>
14 </transition>

```

Figure 5-15: PNML Excerpt resulting from the Application Configuration for the CreateExpenseReport Activity

Beyond configuring an application type and its associated concerns, a complete *Concern Configuration* for a workflow activity also comprises role- and, if necessary, data-specific information. Regarding the former, Figure 5-16 depicts the *Role* section from the transition configuration dialog in INCOME2010. There, the *Role Name* for the selected activity can be entered, a *Role Type* selected and – for the case of cross-organizational workflows with multiple autonomous workflow processes – an *Organization* name be supplied. This set of properties is fully compliant with the XPD L 2.0 standard serving as foundation for the Workflow DSL’s DSM.

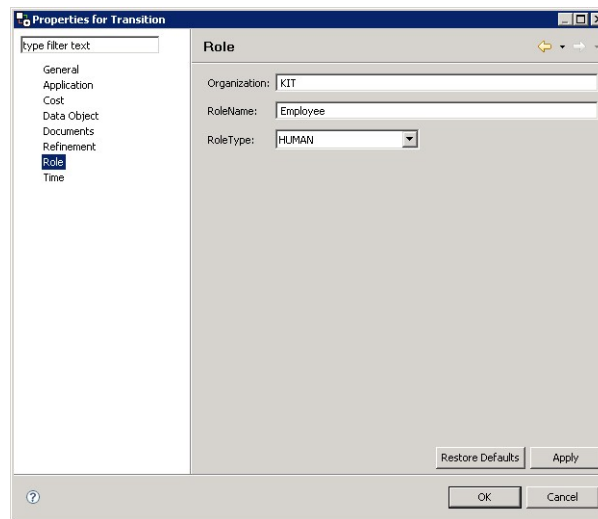


Figure 5-16: Workflow Modeling: Role Assignment in INCOME 2010

Figure 5-17 depicts the corresponding PNML excerpt of the role configuration shown above. The excerpt matches the *wbc:swimlane* tag in line 4 of Figure 5-15. Both the role assignment dialog and the PNML excerpt are completely based on extensions to PNML or INCOME2010 respectively introduced by the Petri Net DIM. This is due to the fact that Petri nets do not provide native support for assigning roles to transitions. In the course of model transformations from the Petri Net DIM to the DSM, the PNML excerpt has to be transformed into the *xpdl:Performer* tag within the *xpdl:Activity* definition (cf. Figure 5-5, line 12) and a global *xpdl:Participant* definition.

```

1 <wbc:swimlane xmlns:wbc="http://www.wsls.net/2006/04/workflow.XPD"2"
2 <wbc:organization>KIT</wbc:organization>
3 <wbc:roleName>Employee</wbc:roleName>
4 <wbc:roleType>HUMAN</wbc:roleType>
5 </wbc:swimlane>

```

Figure 5-17: Role Assignment in PNML

With regard to specifying input- and output parameters of workflow activities and applications as well as defining workflow variables, the *Data Object* section of the transition configuration dialog depicted in Figure 5-18 is used. In the example of the *CreateExpenseReport* activity, an output parameter named *ExpenseReport* and specified in terms of an external schema reference is declared. Internally, the output parameter is mapped to a workflow variable of the same name. Thus, the *ExpenseReport* variable has to be declared only once and can subsequently be

referenced via its name. The set of configurable properties corresponds to the *xpdl:TypeDeclaration* and *xpdl:DataField* specifications which are part of the Workflow DSL's DSM.

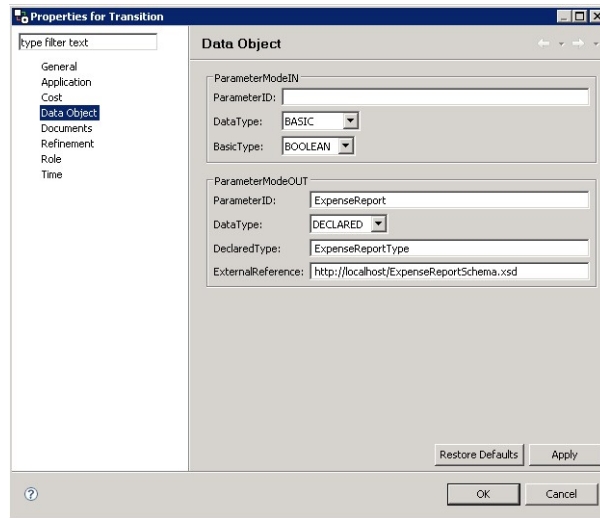


Figure 5-18: Workflow Modeling: Data Object Specification in INCOME 2010

The corresponding PNML excerpt to the data object configuration shown above is depicted in Figure 5-19. It matches the *wbc:dataObject* element in line 5 of Figure 5-15. Again, due to the missing native support of Petri nets for data-related concerns, both the dialog section and the PNML excerpt are completely based on extensions introduced by the Workflow DSL's Petri Net DIM. By means of the model transformations presented in the next section, this excerpt is transformed into a global *xpdl:TypeDeclaration*, a global *xpdl:DataField*, an *xpdl:ActualParameter* element within the *xpdl:Activity* definition and a *xpdl:FormalParameter* definition within the *xpdl:Application* definition. The resulting Workflow DSL program excerpts are depicted in Figure 5-4, Figure 5-5, and Figure 5-6.

```

1 <wbc:dataObject xmlns:wbc="http://www.wsls.net/2006/04/workflow.XPDL2"
2   <wbc:parameterIDOUT>ExpenseReport</wbc:parameterIDOUT>
3   <wbc:dataTypeOUT>DECLARED</wbc:dataTypeOUT>
4   <wbc:declaredTypeOUT>ExpenseReportType</wbc:declaredTypeOUT>
5   <wbc:externalReferenceOUT>
6     http://localhost/ExpenseReportSchema.xsd
7   </wbc:externalReferenceOUT>
8 </wbc:dataObject>

```

Figure 5-19: Data Object Specification in PNML

The *Workflow Modeling* phase is not only supported by the Petri Net DIM, but also by the *BPMN DIM* and the *UML 2.0 Activity DIM*. As described in the respective sections, both DIM notations and their corresponding tools offer the ability to specify the *Concern Configuration* in an analog way as presented here.

Besides the goal of fostering stakeholder involvement, the Petri Net DIM enables formal analysis and simulation of Workflow DSL programs. This can be conducted either based on the simulation features available in INCOME2010 or by importing them via the PNML format into comprehensive analysis and simulation suites like

Woflan (Verbeek and Van der Aalst 2000) or ProM (Van der Aalst, Van Dongen, Günther et al. 2007). Therefore, the Workflow DSL's model transformation framework presented in the next section enables the transformation between arbitrary DIM notations and the Petri Net DIM.

5.4 Model Transformation Framework

In the context of the *Web Engineering DSL Framework*, model transformations inherently play an important role for the mapping between a DSL's *Domain Interaction Models (DIMs)* and the *Domain-Specific Model (DSM)*. This holds equally true for the Workflow DSL and its various DIM notations and associated serialization formats. During the description of the SSO, BPMN, UML 2.0 Activity Diagram and Petri Net DIMs in the preceding section, the necessity of adequate bilateral model transformations between their respective serialization format and the formal schema defined by the DSM was repeatedly pointed out. Besides these *horizontal* model transformations, *vertical* transformations are required in order to realize the mapping to an executable workflow language which in turn enables the adequate instrumentation of a workflow engine.

Against this background, this section presents a novel model transformation framework. Sections 5.4.1 and 5.4.2 introduce the approach's transformation strategy and core concepts. These are exemplified in the course of the design and implementation of a horizontal (cf. Section 5.4.3) and a vertical (cf. Section 5.4.4) model transformation. The section concludes with a complete overview of the mappings between the Workflow DSL's various DIMs and the DSM (cf. Section 5.4.5).

5.4.1 Strategy for Efficient and Effective Model Transformations

From a conceptual perspective, different model transformation strategies between multiple formats are conceivable, e.g. a peer-to-peer strategy, a ring strategy, or a *strategy based on an intermediate format* (cf. Figure 5-20). While the Web Engineering DSL Framework implicitly suggests the latter strategy for transformations between various DIMs, a multi-faceted study of various transformation strategies substantiates this approach (Wüstner, Hotzel and Buxmann 2002). With respect to the requirement of efficiency, the overall costs of a transformation strategy, determined by various cost factors, have to be considered:

$$C_{Strategy} = C_{Implementation} + C_{Transformation} + C_{Errors}$$

While $C_{Implementation}$ comprises the costs for developing the transformations, $C_{Transformation}$ stands for the costs incurred by the execution of the transformations and C_{Errors} represents costs originating from information losses and associated correction efforts.

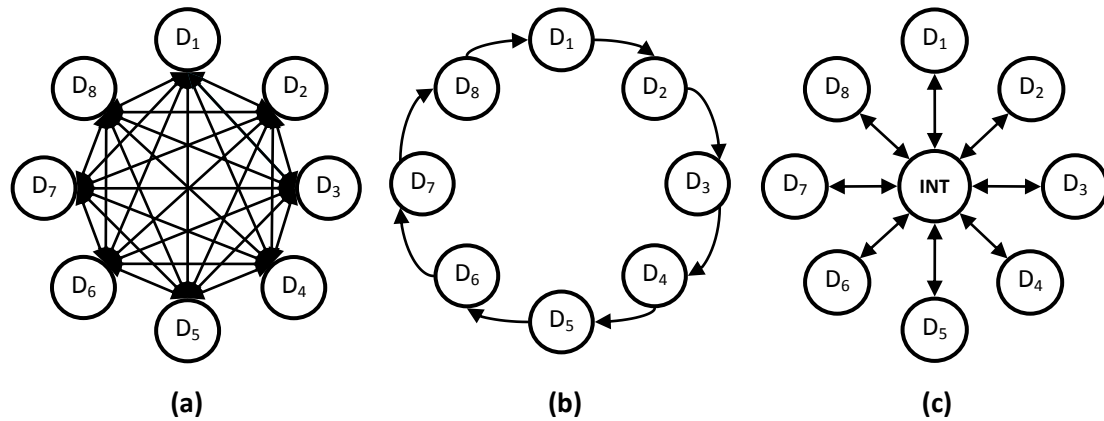


Figure 5-20: Model Transformation Strategies between various DIMs:
 (a) Peer-to-Peer, (b) Ring, (c) Strategy based on an Intermediate Schema

Considering the expected multitude of DIM notations for the Workflow DSL, the third strategy, i.e. *transformation via an intermediate schema*, turns out to be the most efficient. Comparing the transformation implementation costs in this strategy $C_{Implementation} = C_{TransformationImpl} + C_{SchemaImpl}$ with other strategies, additional costs occur for the definition of the intermediate schema. Within the Workflow DSL approach, the DSM forms this intermediate schema and is based on the XPDL 2.0 standard and supplemental Web-specific extensions. Thus, the costs for the intermediate schema consist only in the specification of these extensions. The costs for developing bilateral transformations between the intermediate schema and a DIM are comparable to the other strategies; however, in the case of adding or removing a DIM, no additional costs emerge as existing transformations remain unaffected. Given a set of DIMs $D_1...D_n$, only $2n$ transformations have to be developed.

The costs for the actual transformation process, $C_{Transformation}$, are comparatively low since only two transformation steps, i.e. from the source DIM to the DSM and from the DSM to the target DIM, have to be executed. Furthermore, as the great majority of transformations presented here were implemented as Extensible Stylesheet Language Transformations (XSLT) (Clark 1999), no special transformation applications are required. XSLT is supported by all of today's programming languages and can even be natively executed by established Internet browsers.

The potential costs originating from errors or particularly from information losses, C_{Errors} , are strongly correlated with the heterogeneity of the DIMs and their coverage through the intermediate schema, i.e. the DSM. This issue corresponds to the widely discussed and still unsolved research question of how to integrate heterogeneous business process modeling schemas. In this regard, the model transformation framework proposes a novel approach which is presented in the next subsection. Beyond that and in contrast to the *ring* strategy, this strategy has the advantage that possible information losses do not influence the results of other transformations. This can be of interest if DIM notations shall be introduced which are not capable of sufficiently covering the intermediate schema.

5.4.1.1 Horizontal and Vertical Transformations for the Workflow DSL

Hence, based on the insight that a *transformation strategy via an intermediate schema* is the most efficient for the Workflow DSL, Figure 5-20 gives a corresponding overview of the required model transformations.

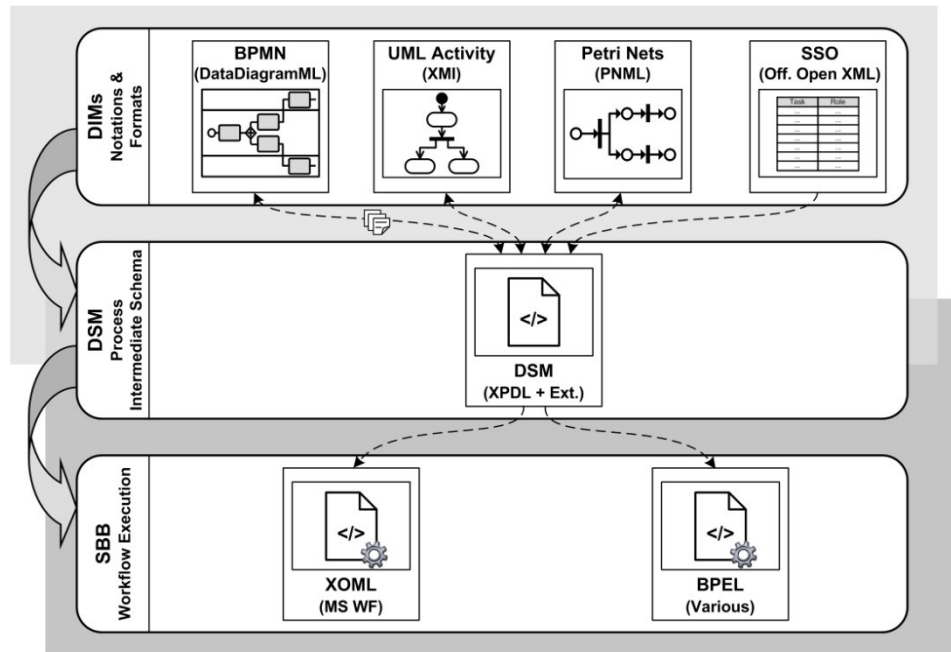


Figure 5-21: Overview of the Workflow DSL's Model Transformations following the Intermediate Schema-based Model Transformation Strategy

The figure shows two types of model transformations occurring in the context of the Workflow DSL: *Horizontal* model transformations between the various DIMs and *vertical* transformations from the DIMs to various workflow execution languages. In this context, the Workflow DSL's DSM acts as *Process Intermediate Schema*. In either case, two transformation steps are required: One transformation from a DIM to the DSM, and one transformation from the DSM either back to another DIM or to a workflow execution language. Thus, in order to enable *cross-notational modeling*, two bilateral transformations to and from the DSM have to be developed for each DIM. In this regard, *effectiveness* in terms of avoiding information loss and thus assuring semantic integrity is crucial. This requirement applies to all DIMs covering a particular level of detail. Thus, in the context of this thesis, lossless, bilateral transformations were specified and developed for the BPMN DIM, the UML 2.0 Activity DIM and the Petri Net DIM (cf. Section 5.4.3). The SSO DIM aims at early requirements engineering activities and consequently covers a much smaller fraction of the problem domain; hence, only a forward transformation from the SSO DIM to the DSM was designed and implemented (Setiawan 2009).

With regard to transformations from the DSM to workflow execution languages, only forward transformations are necessary. These executable workflow specifications are used by the Workflow DSL's *Solution Building Block (SBB)* for the instrumentation of a workflow engine (cf. Section 5.5). As they are not modified on this level at any time, no backward transformations aiming at the preservation of model consistency

are required. In the context of this thesis, a transformation to the XOML format serving as input for the Microsoft Windows Workflow Foundation (WF) was defined and implemented (cf. Section 5.4.4).

5.4.1.2 Determining Composite End-to-End Transformations

Due to the opted transformation strategy based on an intermediate schema, a considerable amount of transformations for the expected multitude of DIMs and workflow execution languages (WLs) emerges. As bilateral transformations are defined between the DIM and the DSM, the set of available transformations between various DIMs is not immediately apparent. Similarly, direct transformations from a DIM to WLs can only be achieved by transitive combinations. However, given a Workflow DSL program in a particular DIM representation, the set of available transformations to other DIMs and WLs is of great interest, e.g. to integrate direct conversion facilities in existing tools.

Thus, in order to foster effective utilization of the set of available transformations, the *transitive closure* of the corresponding transformation graph is calculated. Figure 5-22 depicts an exemplary graph of model transformations for a scenario with bilateral transformations for three DIMs and one forward transformation to a workflow language (WL). By calculating the graph's transitive closure, additional transitive transformations as indicated by the dotted arcs can be determined.

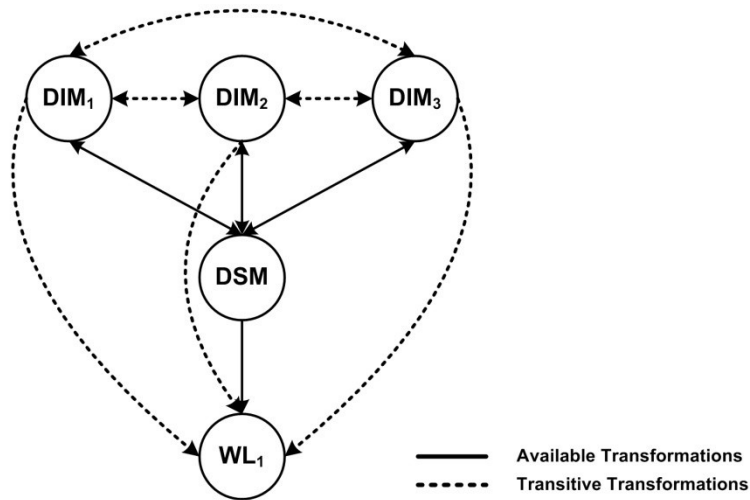


Figure 5-22: Transitive Closure of the Model Transformation Graph

If F is the set of formats, either DIM or WL formats, then the set of implemented transformations T forms a binary relation $T \subseteq F \times F$. The transitive closure T^+ of this relation is accordingly defined as the smallest transitive relation on F that contains T . It can be formally defined as follows:

$$(x, y) \in T^+ \Leftrightarrow (x, y) \in T \vee \exists n \in \mathbb{N}:$$

$$\exists t_1, \dots, t_n: (x, t_1), (t_1, t_2), \dots, (t_{n-1}, t_n), (t_n, y) \in T$$

The obtained set of transitive composite transformations $T^* = T^+ \setminus T$ is of particular interest as it *skips the auxiliary intermediate schema* and contains the *actual*

transformations between individual DIMs as well as from DIMs to workflow execution languages. In the example, this evaluates to:

$$\mathbf{T}^* = \left\{ \begin{array}{l} (DIM_1, DIM_2), (DIM_2, DIM_1), \\ (DIM_1, DIM_3), (DIM_3, DIM_1), \\ (DIM_2, DIM_3), (DIM_3, DIM_2) \\ (DIM_1, WL_1), \\ (DIM_2, WL_1), \\ (DIM_3, WL_1) \end{array} \right\}$$

The transitive closure of the model transformation graph can be calculated by means of the Warshall algorithm which is based on the graph's representation in form of an adjacency matrix (Warshall 1962). In the context of this thesis, the computation of the transformation graph's transitive closure was integrated in the model transformation framework's technical support platform (Orozov 2008). Thus, it is capable to expose and realize the composite transformations contained in the set difference \mathbf{T}^* to external applications and users.

5.4.2 The Core Elements Set (CES) Concept

Achieving *lossless* model transformations for the great variety of conceivable DIMs, e.g. BPMN, UML 2.0 Activity Diagrams, Petri nets, amongst others, is a decisive requirement in order to support multi-notational modeling of Web-based workflows. To this end, several approaches aiming at the definition of an integrated superset for heterogeneous business process modeling languages have been proposed, e.g. (Mendling, Laborda and Zdun 2005; Mendling, Neumann and Nüttgens 2005; Hornung, Koschmider and Mendling 2006). Yet there is doubt whether schema integration as a bottom-up methodology combined with basic refactoring activities is a sufficient solution. The majority of presented integration approaches consider only the case of two schemas to be integrated. This might be due to the fact that the presented methodologies become increasingly complex with the number of integrated schemas. Moreover, guidance on how to define and realize *bilateral* and *lossless* transformations between such an integrated schema and individual business process languages is missing. Similarly, in a bottom-up schema integration-based approach, existing transformations are likely to require modifications when a new schema is added.

Thus, in consideration of the existing multitude of business process modeling languages and notations, a generic, efficiently extensible and more comprehensive solution is required. This holds true particularly in the context of the Workflow DSL in order to enable multi-notational modeling of Web-based workflows. To this end, a novel approach founded on the idea of a *Core Elements Set* concept was developed in the context of this thesis and carried on in several theses and publications (Buck 2007; Freudenstein, Buck, Nussbaumer et al. 2007; Orozov 2008; Setiawan 2008; Setiawan 2009). The *Core Elements Set* (CES) forms a set of common business process and workflow concepts which abstracts from an individual notation or

language. In contrast to the above-mentioned methodologies, the CES is not intended to provide full coverage for all theoretically possible modeling constructs. It rather pursues an approach similar to the Pareto principle, also referred to as “the vital few and the trivial many” (Joran 1954). Accordingly, the CES focuses on few core concepts which establish sufficient support for the great majority of scenarios occurring in practice. At the same time, the introduction of the CES concept enables the adequate resolution of the above-mentioned key challenges, which remain unsolved by alternative approaches.

The restriction of the rather large set of possible concepts to few core concepts included in the CES does actually not present a significant limitation to the Workflow DSL’s applicability in practice. Recently, researchers began to survey and analyze the actual usage distribution of business process modeling constructs in practice, e.g. (Recker 2008; Zur Muehlen and Recker 2008). It turned out that in the most cases, less than 20% of the available modeling constructs are actually used. For example, in the case of BPMN, nine core modeling elements out of fifty elements defined in the BPMN specification proved to be sufficient. Similarly, it was shown that adding further symbols to a well-defined core set adds little expressiveness at the expense of considerably decreased ontological clarity (Zur Muehlen, Recker and Indulska 2007). Although the CES concept was envisioned before these empirical findings, it presents a consequent next step towards their utilization. In the context of this thesis, a comprehensive study of the Core Elements Set’s applicability for real-world process models was conducted and is presented in Section 8.1. The study’s positive results further confirm the assumption that the CES’s restrictive nature does actually pose only very few limitations in practice.

5.4.2.1 CES-based Model-to-Model Transformation Strategy

Figure 5-23 illustrates the *CES-based model transformation concept*. In this context, the CES is considered as *quasi-meta-metamodel*. When defining a model transformation for the Workflow DSL, either the *source* or the *target metamodel* corresponds to the Workflow DSL’s DSM, whereas the other corresponds to a DIM or workflow execution language respectively.

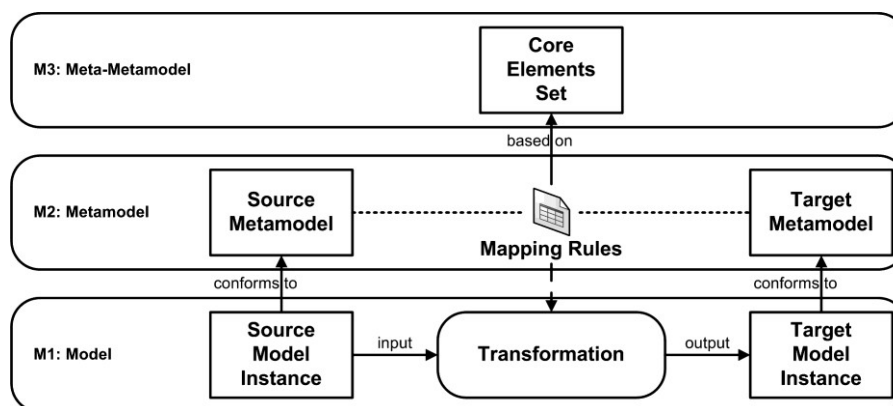


Figure 5-23: CES-based Model-to-Model Transformation Strategy

For each abstract concept in the CES, the respective counterparts from the *source* and *target metamodel* have to be identified. Subsequently, *mapping rules* between the identified metamodel elements are defined. By applying these mapping rules in form of an automated *model transformation*, a *model instance* conforming to the *source metamodel* is transformed into a *model instance* of the *target metamodel*.

In the context of the Workflow DSL, bilateral transformations for DIMs or workflow execution languages are always defined between the respective notation or format and the DSM. The DSM itself can be considered as a formalized XML-based representation of the CES. Once the projection of the CES to corresponding DSM concepts and associated markup has been defined (cf. Section 5.4.5), it remains constant for all newly developed transformations. Thus, when integrating a new DIM or workflow specification language, only the new format's mapping to the CES concepts needs to be identified. To this end, an incremental approach focusing first on the graphical notation and abstracting from its serialization in a particular (XML-based) interchange format is advisable. Having thus identified the mapping on a schematized pattern basis, detailed transformations between each CES concept's markup representation in the new format and its DSM-based counterpart have to be specified and implemented.

In this way, the CES's restriction to core business process and workflow modeling concepts is projected onto the source or target metamodel respectively as well as onto the transformation between them. Consequently, when defining a DIM for the Workflow DSL, the CES provides guidance on which DIM-specific modeling concepts to integrate into the DIM. Due to the thereof resulting semantic congruence between all DIMs, the DSM, and workflow execution languages, *lossless multilateral transformations* are achieved. Furthermore, due to the decoupled arrangement based on the DSM and the CES, the *autonomy of existing transformations* is preserved when new DIMs and workflow execution languages are added.

5.4.2.2 Overview of the Core Elements Set

The CES and its particular concepts categorized along the five workflow perspectives are shown in Table 5-1. While the concepts from the behavioral perspective correspond to selected Workflow Control-Flow Patterns (Russell, ter Hofstede, Van der Aalst et al. 2006), the *Workflow Data* concept corresponds to the Workflow Data Pattern 'Case Data' (Russell, Ter Hofstede, Edmond et al. 2004a) and the *Participant / Role-based Distribution* concept relates to the similarly termed Workflow Resource Pattern (Russell, Ter Hofstede, Edmond et al. 2004b). An overview of the Core Elements Set's mapping to the DSM and the DIMs can be found in Section 5.4.5.

Table 5-1: The Core Elements Set (CES)

Core Elements Set (CES)	
Functional Perspective	
Workflow Process	Constitutes the root container element for the workflow specification.
Activity	The atomic unit of work within a <i>Workflow Process</i> .

<i>Behavioral Perspective</i>	
Concept	Description
Start Node	Marks the start of a <i>Workflow Process</i> .
End Node	Marks the end of a <i>Workflow Process</i> .
Sequence	Defines a sequential execution of <i>Activities</i> .
AND-Split	Divergence of a branch into two or more subsequent branches which are executed concurrently.
AND-Join	Convergence of two or more concurrently executed branches into a single subsequent branch whereby all incoming branches have to be enabled for the transition.
XOR-Split	Divergence of a branch into two or more branches whereby the thread of control is passed to exactly one branch based on an associated condition.
XOR-Join	Convergence of two or more branches into a single subsequent branch whereby only one incoming branch has to be enabled for the transition.
OR-Split	Divergence of a branch into two or more branches whereby the thread of control is passed to one or more outgoing branches based on an associated condition.
OR-Join	Convergence of two or more branches which have been activated by a prior <i>OR-SPLIT</i> whereby all active branches have to be enabled for the transition.
Structured Loop	Encapsulates a set of <i>Activities</i> which shall be executed repeatedly. An associated condition determines whether the loop shall be continued or terminated and is either evaluated at the beginning (<i>While-Do-Loop</i>) or the end (<i>Do-While-Loop</i>) of the loop.
<i>Informational Perspective</i>	
Workflow Data	Typed variables for storing data to be accessible to all components within an instance of the <i>Workflow Process</i> .
<i>Organizational Perspective</i>	
Participant / Role-based Distribution	Serves for defining process participants in form of <i>roles</i> , i.e. groups of resources with similar characteristics, and assign <i>Activities</i> to them.
<i>Operational Perspective</i>	
Application	Specification of an application and its interface which is assigned to one or more <i>Activities</i> and supports or fully automates their processing.

5.4.3 Horizontal Model Transformations – The Petri Net DIM

In this section, the CES-based model transformation concept is exemplified by the Petri Net DIM and can be analogically adopted for other horizontal DIM transformations. For a detailed description of the transformations for the BPMN DIM, please refer to (Orozov 2008). A detailed presentation of the transformations for the UML 2.0 Activity Diagram DIM is available in (Setiawan 2008). The transformations for the SSO DIM can be found in (Setiawan 2009).

For the Petri Net DIM and the related transformations, a block-structured modeling approach is assumed. That is, a Petri net model should be decomposable into non-overlapping blocks. For example, in a branch between an AND-Split and an AND-JOIN, no arcs leading outside this branch should exist. Furthermore, SPLIT constructs of a particular type (AND, OR, XOR) should always have a corresponding JOIN construct of the same type and on the same hierarchical level.

In the following, according to the introduced model transformation development approach, the mapping of Petri net modeling patterns to CES concepts is presented. As Petri nets, in contrast to other business process modeling notations, are composed of only three symbols, the focus lies on the *composite, pattern-based representation of the CES concepts* as well as associated special cases. Due to the same reason, the serialization of the presented patterns in the PNML format is composed only of three corresponding elements (*pnml:place, pnml:transition, pnml:arc*) and can be derived straightforwardly from the presented patterns. Hence, it is not further discussed here. Instead, more challenging aspects of the transformation's technical implementation like an automated diagram layout algorithm or the transformation's underlying traversing algorithm are briefly described. A complete in-depth description can be found in (Setiawan 2009).

5.4.3.1 Mapping CES Concepts on Petri Net Patterns

➤ **Workflow Process:**

The *Workflow Process* concept corresponds to one Petri net diagram.

➤ **Activity and Sequence:**

A named Petri net transition corresponds to the *Activity* concept. Hence, a transition can be linked with other concepts like *Application, Participant* or *Workflow Data*. In order to sustain a block-structured modeling approach as described above, for some model patterns the introduction of a *silent transition*, i.e. a transition labeled “[silent]”, is required. Such a transition has no influence on or meaning to the workflow, it is only required from a structural perspective. Thus, silent transitions are either modeled manually due to the block-oriented modeling guidelines or created in the context of a transformation from the DSM to the Petri Net DIM. Regarding transformations from the Petri Net DIM to the DSM, they are ignored.

The CES concept *Sequence* is represented by a linear, non-conditional sequence of a transition, a place and a transition. Thereby, the first transition may have only one outgoing arc, the place only one incoming and one

outgoing arc and the second transition only one incoming arc. Figure 5-24 illustrates the described CES concepts in their Petri net representation.



Figure 5-24: The CES Concepts Activity (a) / Silent Activity (b) and Sequence (c)

➤ **Start Node and End Node:**

The CES concept *Start Node* is represented as a place without incoming arcs. Correspondingly, the concept *End Node* is expressed by a place without outgoing arcs. Figure 5-25 shows the Petri net representation of these CES concepts.



Figure 5-25: The CES Concepts Start Node (a) and End Node (b)

➤ **AND-Split and AND-Join:**

The CES concept *AND-Split* is represented by a transition with two or more outgoing non-conditional arcs. Accordingly, the *AND-Join* concept corresponds to a transition with two or more incoming non-conditional arcs. Such transitions represent not necessarily *Activities*. In some cases, when the semantic concept prior to the *AND-Split* or after the *AND-Join* concept respectively is not an *Activity*, a *silent transition* is required. Examples could be an immediate sequence of *AND-Joins*, an *AND-Join* directly after the *Start Node* or a new *AND-Split* after the *AND-Join*. Figure 5-26 shows the Petri net representation of these CES concepts.

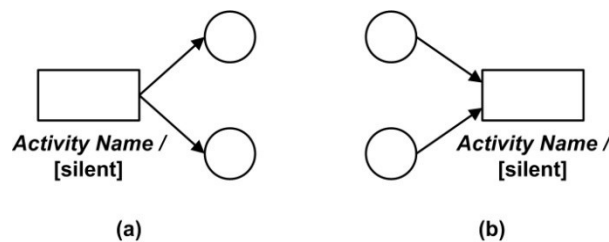


Figure 5-26: The CES Concepts AND-Split (a) and AND-Join (b)

➤ **XOR-Split and XOR-Join:**

The CES concept *XOR-Split* is represented by a place with two or more outgoing conditional arcs. Thereby, one arc marks the default branch and is thus labeled *Otherwise*. For all other arcs, exclusive *conditions* which evaluate to at most one particular branch have to be defined. Accordingly, the *XOR-Join* concept corresponds to a place with two or more incoming non-conditional arcs. The transitions succeeding after the *XOR-Split* place or prior to the *XOR-Join* place may either be *Activities* or *silent transitions*. The latter

are used for cases where the semantic concepts after the *XOR-Split* or before the *XOR-Join* construct respectively do not correspond to the CES concept *Activity*. Example scenarios requiring *silent transitions* after the *XOR-Join* could be nested, immediately succeeding *XOR-Splits* or an immediately succeeding *AND-Split*. Similarly, examples requiring *silent transitions* before the *XOR-Join* could be other directly preceding *XOR-Joins* or *AND-Joins*. Figure 5-27 shows the Petri net representation of the *XOR-Split* and *XOR-Join* CES concepts.

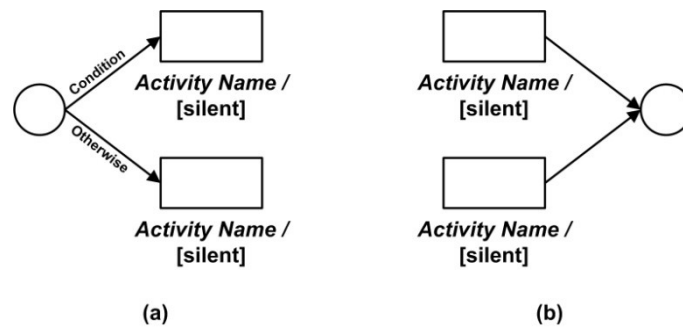


Figure 5-27: The CES Concepts *XOR-Split* (a) and *XOR-Join* (b)

➤ ***OR-Split* and *OR-Join*:**

The CES concept *OR-Split* is represented by a *composition* of an *AND-Split* and two or more *XOR-Split* constructs. Each *XOR-Split* construct includes only one conditional branch and the default branch includes solely a *silent transitions*. In this way, the *OR-Split* semantics, i.e. the concurrent activation of one or more branches according to branch-specific conditions, can be realized. Accordingly, the *OR-Join* concept is composed of an *AND-Join* and several preceding *XOR-Join* constructs. Figure 5-28 depicts the resulting composite Petri net representation of the *OR-Split* and *OR-Join* CES concepts.

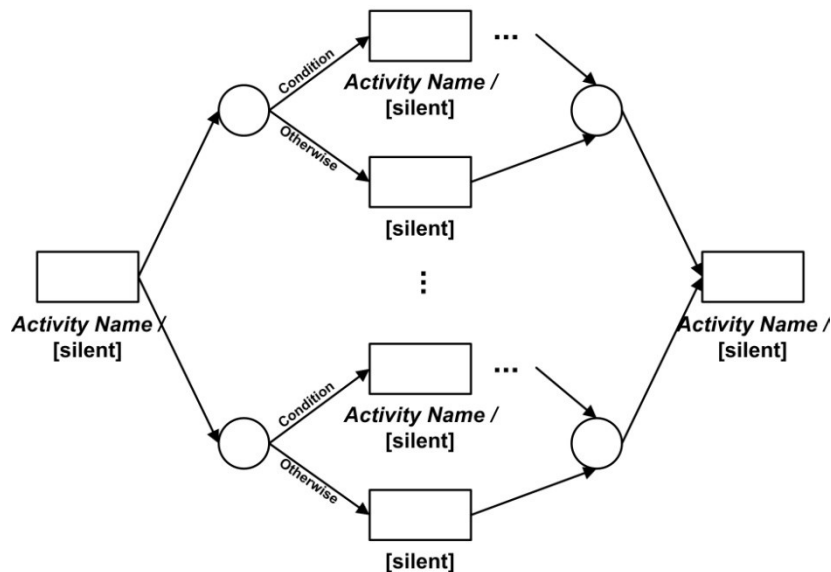


Figure 5-28: The CES Concepts *OR-Split* and *OR-Join* as Composition of an *AND-Split/-Join* as well as multiple *XOR-Split/-Join* structures nested therein

➤ **Structured Loop (While-Do-Loop and Do-While-Loop)**

The CES concept *Structured Loop* can occur in two variants depending on whether the loop condition shall be tested at the beginning (*While-Do-Loop*) or at the end of the loop (*Do-While-Loop*). The former is represented by a *While-Do-Loop-typed place* with a *conditional arc* leading to the loop body, a *default arc* to be followed after the termination of the loop and marked with *Otherwise*, as well as an *incoming arc* returning from the loop body. Similarly, the *Do-While-Loop* variant is represented by a *Do-While-Loop-typed place* with an analog set of arcs. A *silent transition* is inserted between the loop place and the loop body in order to achieve a Petri net-conforming representation. Figure 5-29 illustrates the Petri net representation of the CES concept *Structured Loop* in form of a *While-Do-Loop* and a *Do-While-Loop*.

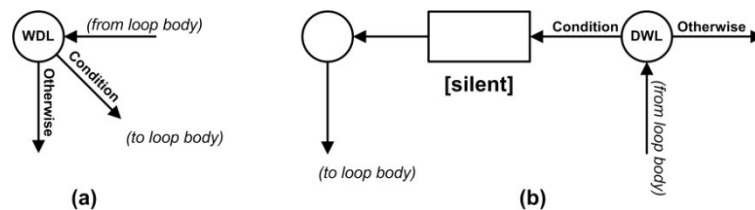


Figure 5-29: The CES Concept Structured Loop as While-Do-Loop (a) and Do-While-Loop (b)

➤ **Workflow Data, Participant / Role-based Distribution and Application:**

For these CES concepts, no visual representation is defined. They are rather specified in form of *properties* attached to a Petri net *transition*. The incorporation of adequate means for configuration in form of a *transition property editor* was exemplarily shown for the Petri net modeling tool INCOME2010 in Section 5.3.4. There, also the introduced *extensions to PNML* and the mapping of the configured property values onto their extended PNML serialization were presented.

5.4.3.2 Technical Implementation

The model-to-model transformations developed in the context of this thesis were mostly specified in form of Extensible Stylesheet Language Transformations (XSLT) (Clark 1999). By applying such a transformation, a source XML document is transformed into a new target XML document. All transformations are founded on a traversal algorithm which was designed for efficiently traversing the source document's corresponding model graph. The algorithm ensures that all nodes between the start and end node are traversed exactly once. At the time of a node's traversal, it is transformed on a pattern basis and inserted into the target document.

Figure 5-30 illustrates the traversal and transformation algorithm in a pseudo code representation. The transformation starts with identifying the start node and creating a corresponding node in the target document. Then, the subsequent node is searched. Depending on the type of the found node, adequate transformation strategies are applied. For the cases of a branching node or a loop node, a stack-based technique ensuring the correct and unique traversal of model elements was

designed. For each branching node type as well as for loops, a corresponding stack variable is declared. For example, if a branching node is found, it is pushed on the stack and all branches are traversed from the split node to the corresponding join node. After the last branch has been traversed (which is also determined based on the stack), the join node is transformed and inserted into the target document and the split node is popped from the stack. The algorithm ends when the model graph's end node is reached.

```

1  PROGRAM WorkflowDSL_ProcessModel_Transformation_Algorithm
2      stack and_stack, or_stack, xor_stack, loop_stack;
3
4      findStartElement()
5      IF startElement was found
6      THEN
7          and_stack = new stack;
8          or_stack = new stack;
9          xor_stack = new stack;
10         loop_stack = new stack;
11         traverse(startElement);
12     END_IF
13
14     FUNCTION traverse(element)
15     BEGIN
16         CASE element OF
17             START-Element:
18                 transform(element);
19                 /* transform() creates a corresponding element in the target
20                  document */
21                 traverse(findNextElement(element));
22                 /* findNextElement() returns the successor of the
23                  given element */
24
25             END-Element:
26                 /* Finish traversal and create end element in target document */
27                 transform(element);
28
29             Activity-Element:
30                 transform(element);
31                 traverse(findNextElement(element));
32
33             AND-Split, OR-Split, XOR-Split:
34                 transform(element);
35                 PUSH(element) in appropriate stack;
36                 FOR all branches
37                     PUSH branch position (intermediate or last) on stack
38                     traverse(findNextElement(element));
39                 END_FOR
40
41             AND-Join, OR-Join, XOR-Join:
42                 POP branch position from stack
43                 IF current branch is last branch
44                 THEN

```

```

45     transform(element);
46     POP corresponding split element from appropriate stack;
47     traverse(findNextElement(element));
48     END_IF
49
50     LOOP-Element:
51     IF TOP(loop_stack) != element
52     THEN /* IF avoids duplicate traversal of loop node */
53         transform(element);
54         PUSH(element) in loop_stack;
55         traverse(findConditionalArc(element));
56         /* findConditionalArc() finds the loop element's associated
57            conditional arc leading to the loop body and returns its
58            first element */
59
60         POP(element) from loop_stack;
61
62         traverse(findOtherwiseArc(element));
63         /* findOtherwiseArc() finds the loop element's associated
64            default exit arc typedOtherwise and returns the first
65            succeeding element */
66     ELSE
67         /* TOP(loop_stack) = element → Do nothing */
68     END_IF
69     END_CASE
70     END_FUNCTION
71 END_PROGRAM

```

Figure 5-30: Pseudo Code of the Model Traversal and Transformation Algorithm

The XSLT-based implementation of the described stack technique is rather non-trivial. The Extensible Stylesheet Language provides the element `xsl:variable` for declaring variables and setting their value. However, once a value has been set, it cannot be modified anymore. Furthermore, XSL includes no stack-like variable types. These problems were solved by passing the variable's value plus a string-based extension to recursive function calls. For extending and evaluating the string-based stack, the XSL string operations `xsl:concat(...)` and `xsl:substring-after(...)` are used. Figure 5-31 shows an example XSLT excerpt illustrating the XSLT-based realization of the stack technique.

```

1 <xsl:template name="traverseNode"> <!-- Main function -->
2 <!-- Parameter definition for LIFO stack for and-constructs -->
3 <xsl:param name="and_stack"/>
4 [...]
5 <xsl:call-template name="traverseNode"><!-- Recursive function call -->
6 <xsl:with-param name="varDecisionNr" <!-- Push 'notlast' on stack -->
7     select="concat('notlast', $and_stack)"/>
8     [...]
9 </xsl:call-template>
10 [...]
11 </xsl:template>

```

Figure 5-31: XSLT-based Implementation of the Stack Technique

When transforming a Web-based workflow from its DSM-based representation to its Petri Net DIM representation, the introduced concepts and methodologies assure semantic correctness and integrity. However, the obtained transformation result still lacks graphical layout information. To this end, a pragmatic *layout algorithm* was designed which reuses the layout facilities contained in the API of Microsoft Visio 2007 (Microsoft Corp. 2006a). To this end, an adequate Visio stencil containing the target DIM notation's shapes has to be provided. This is used by the algorithm to invisibly rebuild the model, perform an automated layout by using the Visio API and extract the coordinates from the layouted model.

The algorithm can analogically be applied for the other DSM-to-DIM transformations as well. Figure 5-32 briefly describes the layout algorithm for the Petri Net DIM. It was implemented with C# and the .NET Framework in form of a plugin for the employed transformation engine which allows the sequential execution of XSLT- and .NET-based transformation plugins on a source document (cf. Section 5.5).

```

1  PROGRAM Auto_Layout_for_PetriNet_DIM
2      Hashtable XmlElement2VisioShape
3
4      VisioAPI.invisiblyCreateEmptyVisioDiagram();
5
6      FOR all notation symbol types(place, transition, directed arc)
7          FOR all symbol instances in Xml Document to be layouted
8              IF (currentSymbolType != directedArc) THEN
9                  addCorrespondingShapeToVisioDrawing();
10             ELSE
11                 connectSourceAndTargetShapeByDirectedArc();
12             END_IF
13             XmlElement2VisioShape.Add(instance, shapeId); /* Remember mapping*/
14         END_FOR
15     END_FOR
16
17     VisioAPI.moveModelToPageCenter();
18     VisioAPI.performAutomatedLayout(shapeLayoutMode,
19                                     connectorLayoutMode,
20                                     spacing);
21     /* Visio supports various layout modes e.g. Flowchart/Tree, Leftto-
22        Right for shapes andCenter-to-Center for connectors.*/
23
24     VisioAPI.adjustPageToDrawingContents();
25
26     FOR all shapes in Visio diagram
27         positionInfo = VisioAPI.getShapePosition();
28         xmlElement = symbolInstances.findXmlElement(currentShape.Id);
29         targetDocument.selectElement(xmlElement).addLayoutInfo(positionInfo);
30     END_FOR
31 END_PROGRAM

```

Figure 5-32: Pseudo Code of the Layout Algorithm for the Petri Net DIM

5.4.4 Vertical Model Transformations – The XOML Workflow Language

Vertical model transformations bridge the gap between the DSM and the workflow execution languages supported by workflow engines, e.g. the Business Process Execution Language (BPEL) (Jordan and Evdemon 2007) or the Extensible Application Markup Language (XAML). The latter is a declarative XML-based application specification language which was introduced by Microsoft and covers a comprehensive set of concerns including presentation and workflow aspects. In order to ease the differentiation, the acronym XOML is used when the workflow-specific parts of the language are meant (Microsoft Corp. 2007). In the context of this thesis, the Microsoft Windows Workflow Foundation (WF) (Microsoft Corp. 2006c) was adopted as workflow execution framework. Hence, in this section, the transformation between the Workflow DSL's DSM and the XOML language used by the Workflow Foundation is exemplarily presented. This is of particular interest as no transformation between XPDL and XOML has been presented yet, neither by science nor industry. Thus, the presented approach presents a valuable contribution, especially against the background of the WF's increasing relevance and free availability as part of the .NET 3.0 Framework. The presented approach can similarly be applied for other workflow execution languages like BPEL as well. The detailed integration of the presented transformation and the WF in the Workflow DSL's technical framework is explained in Section 5.5.

5.4.4.1 Mapping CES Concepts to XOML Language Elements

The XOML language supports state-machine-based and sequential workflows, whereby the latter forms the adequate type for the Workflow DSL. It is represented by the root element *xoml:SequentialWorkflowActivity* within the XML namespace <http://schemas.microsoft.com/winfx/2006/xaml/workflow>. XOML is a *block-structured* language, i.e. transitions between elements are implicitly described by their subsequent appearance within a control-structure-typed block. For example, activities occurring after the *SequentialWorkflowActivity* root node implicitly form the CES concept *Sequence*. The same applies for activities within all other control structure-typed blocks. Thus, the explicit XOML concept *SequenceActivity* presents only an optional, explicit representation of the CES concept *Sequence*. Similarly, no explicit counterparts for the CES concepts *Start Node* and *End Node* exist. They are rather implicitly represented by the encapsulating *SequentialWorkflowActivity* root node within a XOML-based workflow specification. Furthermore, XOML provides no direct complements for the CES concepts *OR-Split* and *OR-Join*. Therefore, as presented in the previous section for the Petri Net DIM (cf. Figure 5-28), they have to be mapped onto a combination of an *AND-Split* and *-Join* and several encapsulated *XOR-Split* and *-Joins*.

With regard to representing the CES concept *Activity*, the WF offers various predefined activity types. However, like in other workflow execution languages, these activities are completely system-oriented and do not address human interaction. Thus, a dedicated human interaction-oriented activity type had to be introduced. Therefore, an extension mechanism of the WF allowing the development

of specialized *Custom Activities* and their referential usage within XOML-based workflow specifications was adopted. The custom activity is named *WslsActivity* (following the name of the WSLS platform, cf. Sections 4.1.2 and 5.5) and represents external tasks which are performed outside the workflow engine. The activity defines a property for specifying eligible roles (CES concept *Participant*) as well as realizes the sending of input and receiving of output parameters to external clients and their mapping on workflow variables (CES concept *Workflow Data*). The *Concern Configuration* within a Workflow DSL program specifying the Web-based realization of human tasks is not intended to be transformed to a workflow execution language. It is rather evaluated by Web-based clients in order to realize the specified behavior and provide a corresponding user interface. Thus, the *WslsActivity* serves only as *mediator* between the workflow and such client applications. A detailed description of the *WslsActivity*'s implementation can be found in (Buck 2007).

As explained earlier, in the case of vertical model transformations, only unilateral mappings from the DSM to the workflow execution language have to be defined. Thus, for each CES concept, a XOML language concept which is either semantically *equivalent* or which *subsumes* the CES concept has to be identified. To this end, the following notations are used:

$$C_{CES} \equiv C'_{XOML}$$

The CES concept C_{CES} is semantically equivalent to the XOML concept C'_{XOML}

While semantic equivalency between a CES concept and its counterpart in the target format is a necessary requirement for bilateral transformations, the subsumption of a CES concept by a target concept is viable if only unilateral transformations are desired:

$$C_{CES} \subseteq C'_{XOML}$$

The CES concept C_{CES} is semantically subsumed by the XOML concept C'_{XOML}

Table 5-2 shows the identified mappings of CES concepts to XOML language elements as well as their semantic relations based on the introduced notations. In order to achieve a more fine-grained comparison, the CES concepts in their DSM representation present the basis for determining the semantic relationship.

Table 5-2: Mapping of CES Concepts to XOML Elements

CES Concept	XOML Element	Semantic Relation
Workflow Process	SequenceActivity	$dsm_xpd:WorkflowProcess \equiv xoml:SequenceActivity$
Activity	WslsActivity (CustomActivity)	$dsm_xpd:TaskActivity \equiv xoml:WslsActivity$
Sequence	SequenceActivity	$Sequence \equiv xoml:SequenceActivity$
AND-Split / AND-Join	ParallelActivity	$dsm_xpd:AND-Split RouteActivity \subseteq xoml:ParallelActivity,$ $dsm_xpd:AND-Join RouteActivity \subseteq xoml:ParallelActivity,$ $[(dsm_xpd:Activity + dsm_xpd:Route + dsm_xpd:Split + dsm_xpd:Type=AND) + \dots + (dsm_xpd:Activity + dsm_xpd:Route + dsm_xpd:Join + dsm_xpd:Type=AND)] \equiv xoml:ParallelActivity$
XOR-Split / XOR-Join	IfElseActivity	$dsm_xpd:XOR-Split RouteActivity \subseteq xoml:IfElseActivity,$ $dsm_xpd:XOR-Join RouteActivity \subseteq xoml:IfElseActivity,$ $[(dsm_xpd:Activity + dsm_xpd:Route + dsm_xpd:Split + dsm_xpd:Type=XOR) + \dots + (dsm_xpd:Activity + dsm_xpd:Route + dsm_xpd:Join + dsm_xpd:Type=XOR)] \equiv xoml:IfElseActivity$
OR-Split / OR-Join	ParallelActivity + IfElseActivity	$dsm_xpd:OR-Split RouteActivity \subseteq (xoml:ParallelActivity + xoml:IfElseActivity),$ $dsm_xpd:OR-Join RouteActivity \subseteq (xoml:ParallelActivity + xoml:IfElseActivity),$ $[(dsm_xpd:Activity + dsm_xpd:Route + dsm_xpd:Split + dsm_xpd:Type=OR) + \dots + (dsm_xpd:Activity + dsm_xpd:Route + dsm_xpd:Join + dsm_xpd:Type=OR)] \equiv (xoml:ParallelActivity+xoml:IfElseActivity)$
Structured Loop	WhileActivity	$dsm_xpd:LoopStandard-Activity \equiv xoml:WhileActivity$
Workflow Data	DependencyProperty	$xpd:DataField + xpd:TypeDeclaration \subseteq xoml:DependencyProperty$
Participant	WslsActivity (CustomActivity)	$xpd:Participant \subseteq xoml:WslsActivity$
Application	WslsActivity (CustomActivity)	$xpd:Application \supseteq xoml:WslsActivity$

5.4.4.2 Technical Implementation: From Graph-Structured XPDL to Block Structured XOML

While business process modeling languages like BPMN, UML 2.0 Activity Diagrams, Petri Nets, amongst others, as well as the XML Process Definition Language (XPDL) are usually graph-oriented languages, workflow execution languages like BPEL or XOML follow a block-structured specification style. Thus, a transformation from the Workflow DSL's predominantly XPDL-based DSM to the workflow execution language XOML has to deal with additional challenges originating from the different structural styles. Figure 5-33 illustrates these differences by depicting a single process model both in a graph-structured and a block-structured representation.

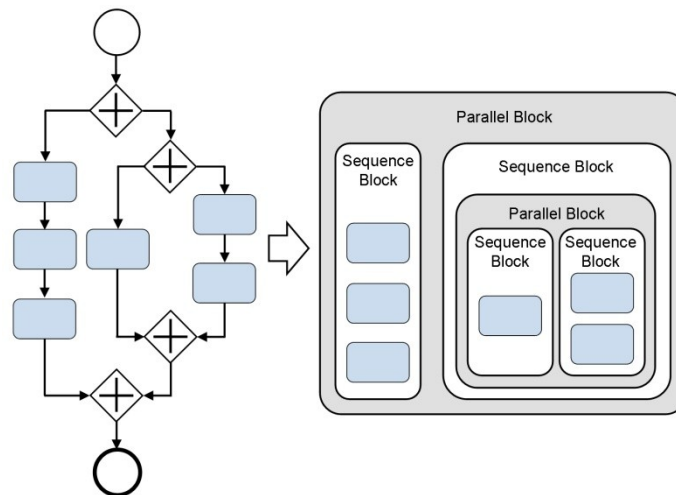


Figure 5-33: Graph-Structured vs. Block-Structured Specification Styles

Hence, the main challenge lies in mapping graph sections onto nested block structures. To this end, a two-staged, fully XSLT-based transformation process was developed. In the first stage, *structural patterns* are identified and extracted from the DSM-based source document, whereby the control flow order is sustained. Thereupon, the second stage conducts the *syntactic transformation* of the extracted block structures into the XOML target format. Due to the two-staged transformation procedure, other workflow execution languages could be flexibly incorporated by adapting the format-specific transformation codes in the second stage. The transformations in the first stage are language-independent and can be retained.

In the following, selected mappings from graph-structured XPDL *patterns* to block-structured XOML statements are schematically illustrated. The complete mapping and its implementation via XSLT is described in detail in (Orozov 2008). Each figure shows on the left the BPMN-based visual representation of the XPDL patterns as well as the associated XPDL markup. On the right, a block structured representation and the corresponding XOML markup is depicted.

Figure 5-34 depicts the *Parallel* pattern in XPDL and XOML. A set of statements between an XPDL *AND-Split-Activity* and an XPDL *AND-Join-Activity* is mapped onto several nested blocks within a XOML *ParallelActivity*.

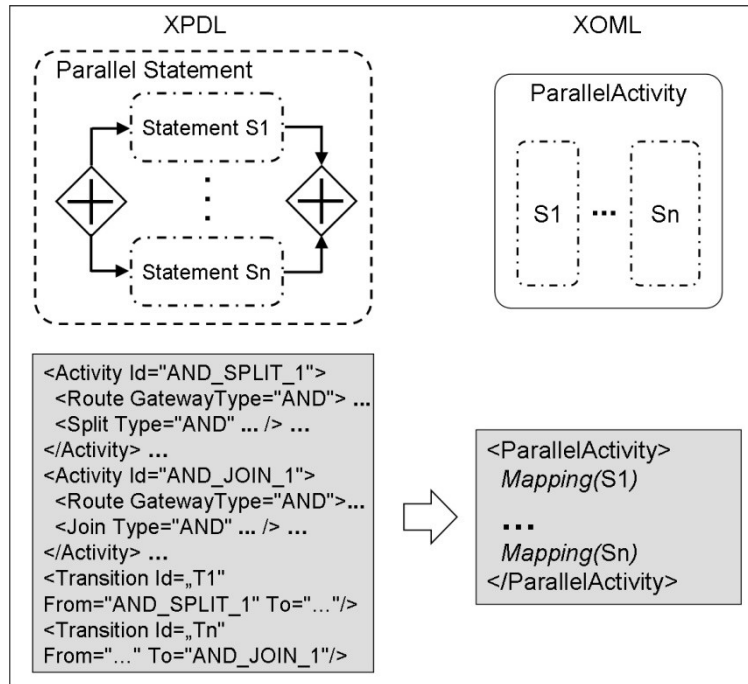


Figure 5-34: The Parallel Pattern in XPD and XOML

Figure 5-35 illustrates the *Decision* pattern. The statements between an XPD *XOR-Split-Activity* and an XPD *XOR-Join-Activity* are mapped onto individual XOML *IfElseBranchActivity* blocks within an XOML *IfElseActivity*. For each XPD *Transition* with a condition, a XOML *RuleExpressionCondition* has to be created and referenced by the corresponding *IfElseBranchActivity*.

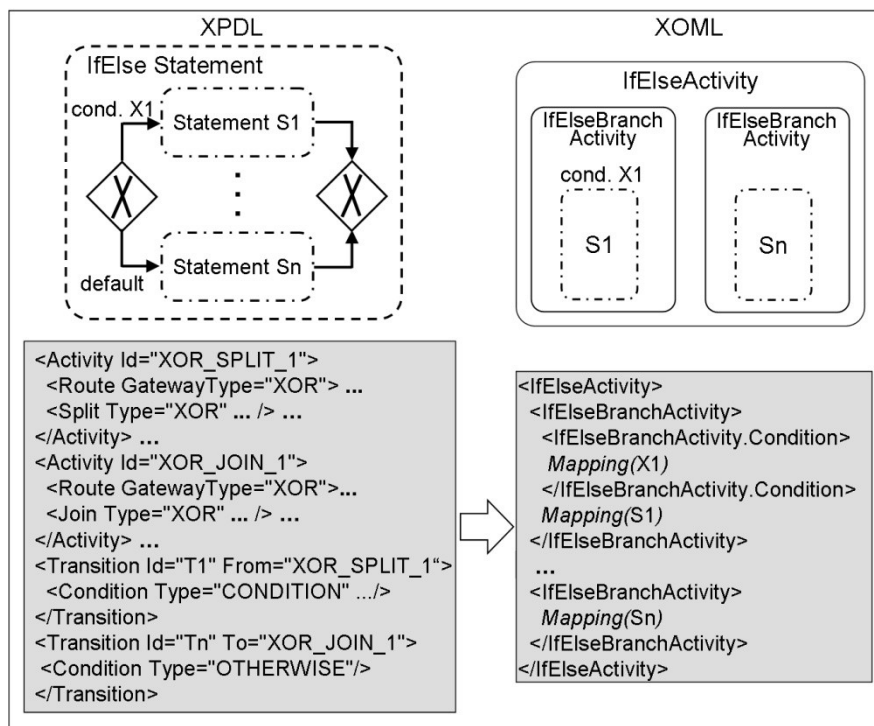


Figure 5-35: The Decision Pattern in XPD and XOML

Such declarative XOML-based rule expressions are stored in a separate rules specification file and can then be referenced from within the workflow specification (cf. Figure 5-36).

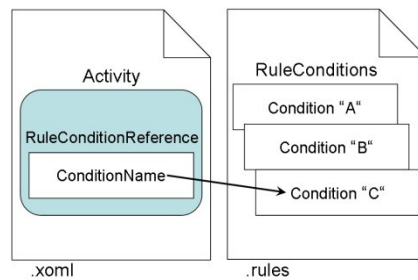


Figure 5-36: Referencing Separately Defined Declarative Rule Conditions

In order to transform a XPDL *Transition Condition* into a XOML-based *RuleExpressionCondition*, the condition has to be parsed and mapped onto a declarative hierarchical XOML representation. For example, a simple less-than-based condition is mapped onto an enclosing *CodeBinaryOperatorExpression* element with the attribute *Operator="LessThan"*. Inside this element, both sides of the expression are separately defined (*CodeBinaryOperatorExpression.Left|Right*). Therefore, either a *CodeMethodInvokeExpression* element allowing the access to workflow variables as well as invoking functions or a *CodePrimitiveExpression* element for providing a comparison value can be used. Within these elements, several other elements are required for specifying types, objects, properties and methods. A detailed example can be found in (Orozov 2008).

Figure 5-37 depicts the *Structured Loop* pattern. The elements within the XPDL loop body are nested as block within the XOML *WhileActivity*. The XPDL *Transition Condition* for the loop is mapped onto a XOML-based *RuleExpressionCondition* and referenced by the XOML *WhileActivity*.

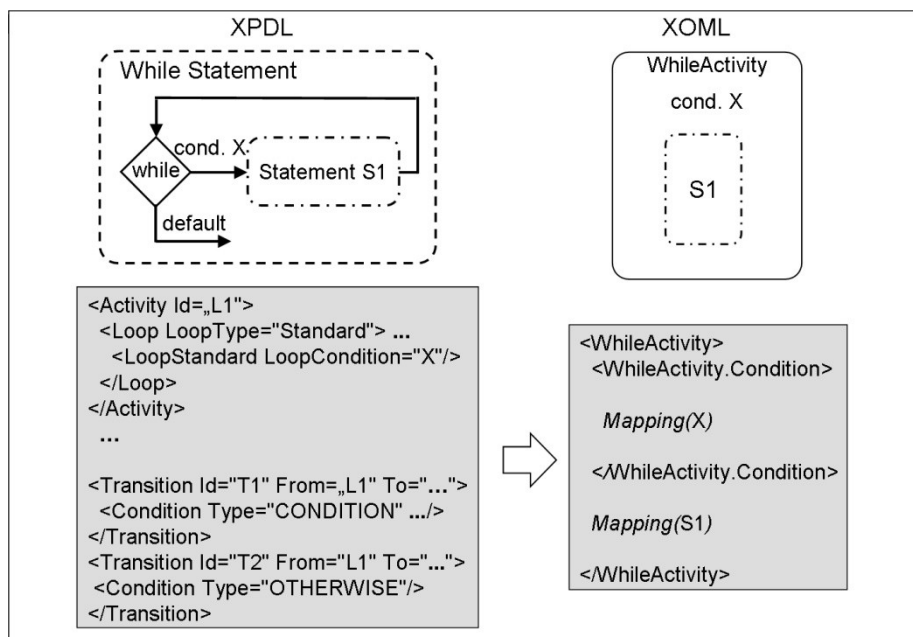


Figure 5-37: The Structured Loop Pattern (variant: While-Do-Loop) in XPDL and XOML

Besides transforming control-flow and transition conditions, also mappings for the CES concept *Workflow Data* have to be performed. In this regard, *XPDL DataFields* and *TypeDeclarations* have to be mapped on XOML *DependencyProperties*. In Figure 5-4, the *xpdl:TypeDeclaration* for the *ExpenseReportType* and an associated *xpdl:DataField* acting as workflow variable were shown. Figure 5-38 shows the corresponding C#-based representation as required by the Windows Workflow Foundation. The XSLT-based transformation is performed on a template-basis whereby the bold marked terms are variable and have to be filled according to the XPDL specification while the non-bold sections remain constant.

```

1  public partial class BusinessTripWorkflow : SequentialWorkflowActivity
2  {
3      public static DependencyProperty ExpenseReportProperty =
4          DependencyProperty.Register(ExpenseReport",
5                                     typeof(System.Xml.XmlElement),
6                                     typeof(KIT.BusinessTripWorkflow));
7
8      [System.ComponentModel.DesignerSerializationVisibilityAttribute(
9          DesignerSerializationVisibility.Visible)]
10     [System.ComponentModel.BrowsableAttribute(true)]
11     [System.ComponentModel.CategoryAttribute("Parameters")]
12     public System.Xml.XmlElement ExpenseReport
13     {
14         get
15         {
16             return ((System.Xml.XmlElement) (base.GetValue(
17                 KIT.BusinessTripWorkflow.ExpenseReportProperty))); }
18         set
19         {
20             base.SetValue(
21                 KIT.BusinessTripWorkflow.ExpenseReportProperty, value); }
22     } [...]
23 }

```

Figure 5-38: XOML *DependencyProperty* for XPDL *TypeDeclaration* and *DataField*

In summary, the transformation results in three separate parts within a single XML-based result document: The XOML-based workflow specification, the XOML-based rules declarations as well as the C#-based data representation. Each of these is stored in a separate file with the extension *.xoml*, *.rules*, and *.cs* respectively and automatically passed to the Workflow Foundation's workflow compiler. The resulting workflow library can then be directly used for the instrumentation of the Workflow Foundation's workflow engine. Thereupon, workflow instances based on this library can be immediately instantiated and executed.

In conclusion, human interaction-enabled workflows can thus be constructed on a pure model basis. The automated construction of complementary Web-based user interfaces based on the *Application* specifications contained in the DSM as well as their interaction with the workflow engine is presented in Section 5.5. There, also the model transformation framework's technical support framework is described.

5.4.5 Complete Catalog of DIM Mappings

This section gives an overview of the mappings between all presented DIMs and the Workflow DSL's DSM which is mainly founded on XPD L 2.0.

Table 5-3: Complete Overview of Mappings between DIMs and the DSM (Part 1/2)

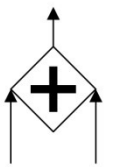
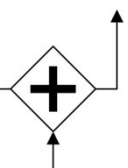
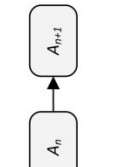
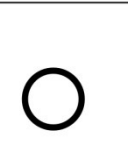
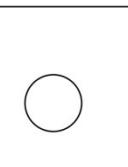
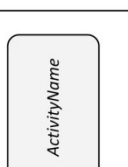

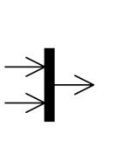
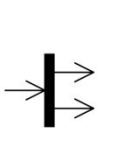
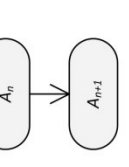

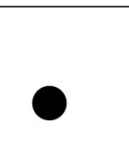
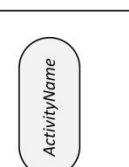

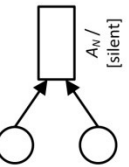
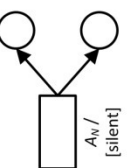
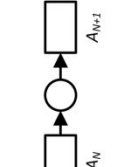
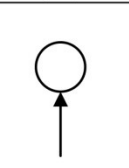
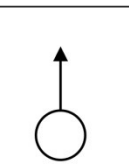
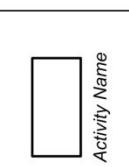






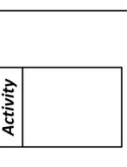
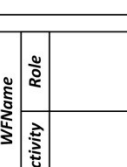














CES Concepts DSM / DIMs	AND-Join <pre><xpdl:Activity> <xpdl:Route GatewayType="AND"> ... <xpdl:Join Type="AND"> <xpdl:TransitionRefs>... </xpdl:TransitionRefs> ... </xpdl:Activity></pre> 	AND-Split <pre><xpdl:Activity> <xpdl:Route GatewayType="AND"> ... <xpdl:Split Type="AND"> <xpdl:TransitionRefs>... </xpdl:TransitionRefs> ... </xpdl:Activity></pre> 	Sequence <pre><xpdl:Activity> <xpdl:Transition></pre> 	End Node <pre><xpdl:Activity></pre> 	Start Node <pre><xpdl:Activity StartActivity="true"></pre> 	Activity <pre><xpdl:Activity></pre> 	Workflow Process <pre><xpdl:WorkflowProcess></pre> 	Business Process Modeling Notation (BPMN)	UML 2.0 Activity Diagram	Petri Net	Simple Sequence Only (SSO)
	AND-Join 	AND-Split 	Sequence 	End Node 	Start Node 	Activity 	Workflow Process 	Business Process Modeling Notation (BPMN)	UML 2.0 Activity Diagram	Petri Net	Simple Sequence Only (SSO)
	AND-Join 	AND-Split 	Sequence 	End Node 	Start Node 	Activity 	Workflow Process 	Business Process Modeling Notation (BPMN)	UML 2.0 Activity Diagram	Petri Net	Simple Sequence Only (SSO)
	AND-Join 	AND-Split 	Sequence 	End Node 	Start Node 	Activity 	Workflow Process 	Business Process Modeling Notation (BPMN)	UML 2.0 Activity Diagram	Petri Net	Simple Sequence Only (SSO)
	AND-Join 	AND-Split 	Sequence 	End Node 	Start Node 	Activity 	Workflow Process 	Business Process Modeling Notation (BPMN)	UML 2.0 Activity Diagram	Petri Net	Simple Sequence Only (SSO)
	AND-Join 	AND-Split 	Sequence 	End Node 	Start Node 	Activity 	Workflow Process 	Business Process Modeling Notation (BPMN)	UML 2.0 Activity Diagram	Petri Net	Simple Sequence Only (SSO)

Table 5-4: Complete Overview of Mappings between DIMs and the DSM (Part 2/2)

CES Concepts DSM / DIMs	XOR-Split	XOR-Join	OR-Split	OR-Join	Structured Loop	Workflow Data	Participant	Application
DSM (Extended XPDL 2.0)	<pre><xpdl:Activity> <xpdl:Route GatewayType="XOR"> ... <xpdl:Split Type="XOR"> <xpdl:TransitionRefs>... </xpdl:TransitionRefs> ... </xpdl:Activity></pre>	<pre><xpdl:Activity> <xpdl:Route GatewayType="XOR"> ... <xpdl:Join Type="XOR"> <xpdl:TransitionRefs>... </xpdl:TransitionRefs> ... </xpdl:Activity></pre>	<pre><xpdl:Activity> <xpdl:Route GatewayType="OR"> ... <xpdl:Split Type="OR"> <xpdl:TransitionRefs>... </xpdl:TransitionRefs> ... </xpdl:Activity></pre>	<pre><xpdl:Activity> <xpdl:Route GatewayType="OR"> ... <xpdl:Join Type="OR"> <xpdl:TransitionRefs>... </xpdl:TransitionRefs> ... </xpdl:Activity></pre>	<pre><xpdl:Activity> <xpdl:Loop LoopType="Standard"> <xpdl:LoopCondition="< LoopCondition="< TestTime="Before" After"/> </xpdl:Loop> </xpdl:Activity></pre>	<pre><xpdl:TypeDeclaration> <xpdl:DataField></pre>	<pre><xpdl:Participant> ... <xpdl:Activity> ParticipantName <xpdl:Performer> </xpdl:Activity></pre>	<pre><xpdl:Application> <wc:Dialog_Extension> <wc:WebService_ Extension> <wc:DataPresentation_ Extension> <wc:Commit_AppType></pre> <p>Activity-specific Properties (e.g. Visio Shape Data)</p>
Business Process Modeling Notation (BPMN)								<p>UML Action Stereotypes: <pre><<wbApplication>> <<wbDialog>> <<wbWebService Communication>> <<wbData Presentation>> <<wbCommit>></pre></p>
UML 2.0 Activity Diagram								<p>Transition-associated Properties</p>
Petri Net								<p>Transition-associated Properties</p>
Simple Sequence Only (SSO)								<p>Transition-associated Properties</p>

5.5 Technical Platform

The Workflow DSL's technical platform comprises various layers covering different aspects and is depicted in Figure 5-39. These include business process and workflow modeling tools, the model transformation framework's technical platform as well as the workflow execution platform. The various modeling tools and their associated standardized serialization formats have already been presented in the previous sections and are thus not further discussed here. It may be noted though that, based on the introduced model transformations, any modeling tool supporting one of these standardized serialization format could be adopted. In the following, the other layer's components as well as the process for the automated construction of Web-based workflows will be presented.

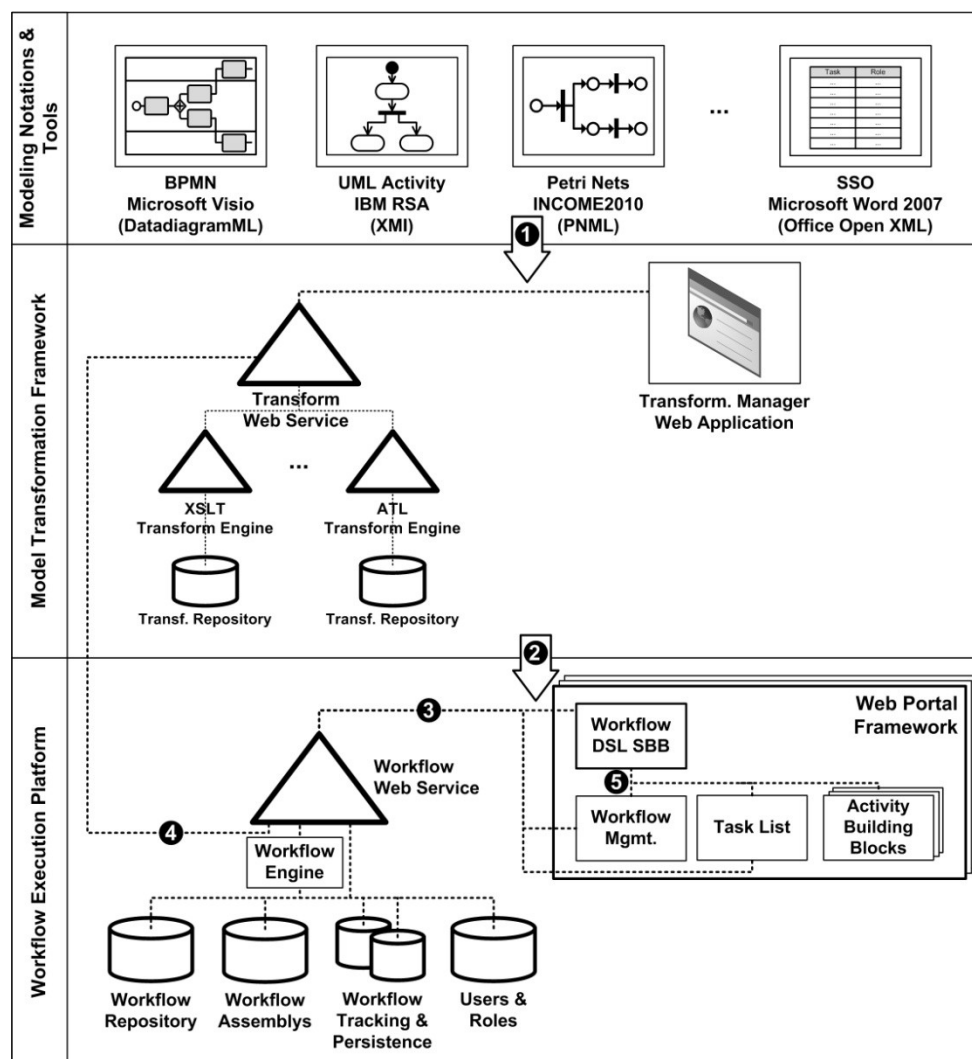


Figure 5-39: The Technical Platform of the Workflow DSL

5.5.1 Technical Platform for the Model Transformation Framework

The technical support platform for the model transformation framework contains two core components: The *Transform Web Service*, an *XSLT-based Transformation Engine*, and the *Transformation Manager Web Application*. The former enables the integration of model transformation management and execution facilities in external applications and systems, e.g. modeling tools. Furthermore, it allows the integration of various transformation engines and acts as mediator between requesting clients and these engines. The Transformation Manager Web Application uses the Transform Service and provides a user interface for administrating and executing model transformations.

5.5.1.1 Transform Web Service

Figure 5-40 shows an excerpt of the Transform Web Service's public interface which provides operations for managing formats and associated transformations as well as for executing transformations. A detailed description of the interface can be found in (Orozov 2008). The first set of operations addresses the creation (*CreateCoreElementsSet*), retrieval (*GetCoreElementsSets*, *GetCoreElementsSetByName*), modification (*UpdateCoreElementsSet*) and deletion (*DeleteCoreElementsSet*) of formats. As indicated by the operation names, the focus lies thereby on a format's mappings to the introduced *Core Elements Set (CES)*. Similarly, the second set of operations covers the creation (*CreateTransformation*), retrieval (*GetTransformations*, *GetTransformationByName*), modification (*UpdateTransformation*) and deletion (*DeleteTransformation*) of model transformations between formats. Thereby, the operation *GetTransformations* return both *direct* and automatically computed *transitive* transformations based on the model transformation graph's *transitive closure* as described in Section 5.4.1.2. Finally, the *Transform* operation allows for executing transformations on a given XML-based input document and returns the result again in form of an XML document.

```

/* Part 1: Operations for Managing Formats and their CES Mappings */
public void CreateCoreElementsSet(XmlElement coreElementsSet)
public XmlCollection GetCoreElementsSets();
public XmlElement GetCoreElementsSetByName(string coreSetName)
public void UpdateCoreElementsSet(XmlElement coreElementsSet)
public void DeleteCoreElementsSet(string coreSetName)

/* Part 2: Operations for Managing Transformations */
public void CreateTransformation(XmlElement transformation)
public XmlCollection GetTransformations();
public XmlElement GetTransformationByName(string transformName)
public void UpdateTransformation(XmlElement transformation)
public void DeleteTransformation(string transformName)

/* Part 3: Operation for Executing Transformations */
public XmlElement Transform(string name, string type, XmlElement inputXML)

```

Figure 5-40: Excerpt of the Transform Web Service's Public Interface

Thus, based on this interface, all kinds of applications and systems can use the Transform Web Service for managing and executing transformations. In order to preserve interface continuity for existing clients in the case of modifications or extensions, the Web service uses the generic type *XmlElement* for parameters which are likely to be subject to evolution (Nussbaumer 2008). Thus, based on the same interface definition, new clients can consume extended functionalities while existing clients continue to use earlier versions. Systematic versioning is achieved based on XML namespaces.

Against the background of the emerging multitude of model transformation languages (Czarnecki and Helsen 2003), the Transform Web Service supports the flexible incorporation of transformation engines based on the *Strategy* design pattern (Gamma, Helm, Johnson et al. 1995). According to the Strategy pattern, a family of algorithms can be specified, encapsulated and flexibly interchanged at runtime. Hence, the Transform Web Service encapsulates the common characteristics of transformation engines in an interface definition which can then be implemented by various engines. Thus, the particular engine used for a transformation can be selected at runtime. Furthermore, new engines can be integrated without requiring code-based modifications to the Transform Service.

Figure 5-41 illustrates the adoption of the Strategy design pattern for the Transform Web Service based on a simplified class diagram excerpt. The class *TransformService* represents the relevant part of the Transform Web Service. The interface definition *TransformEngine* specifies the methods a transform engine has to implement. These address the retrieval of the set of available transformations as well as their execution on a given XML-based source document.

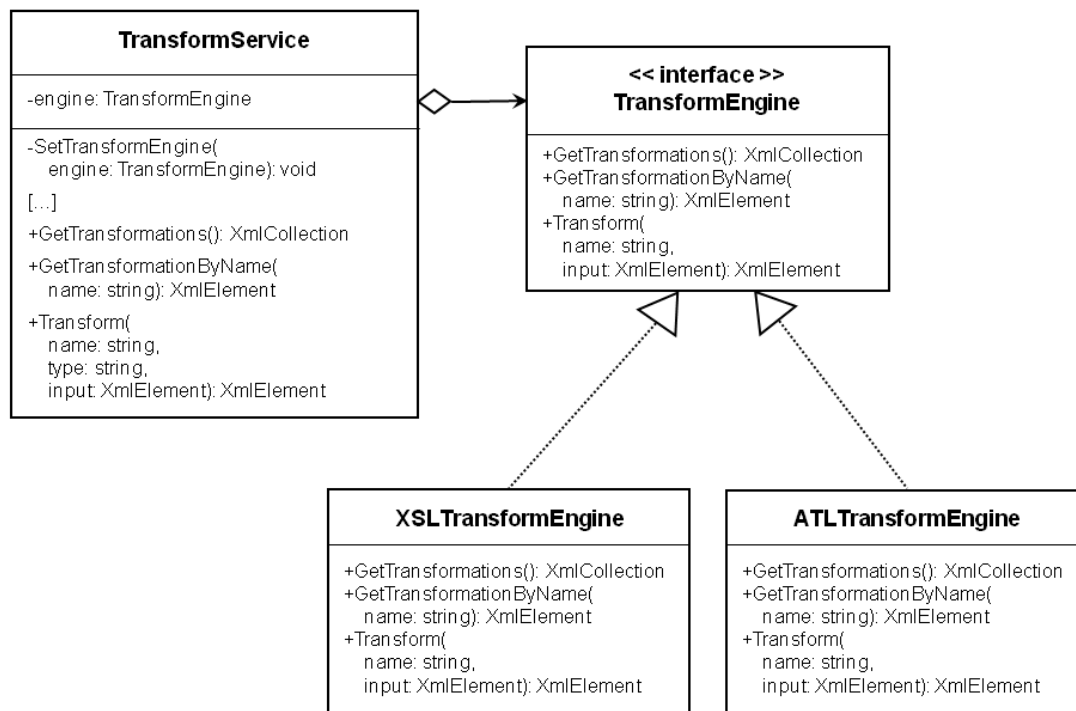


Figure 5-41: Achieving Interchangeable Transformation Engines based on the Strategy Design Pattern

The *TransformService* class contains a reference to a particular engine which can be interchanged at runtime by invoking the method *SetTransformationEngine*. Thus, depending on a requested transformation's type, the Transform Web Service selects the appropriate engine. In this regard, the figure exemplarily depicts two possible implementations of the *TransformEngine* interface. The *XSLTransformEngine* realizes XSLT-based model transformations (cf. Section 5.5.1.2) and the *ATLTransformEngine* processes transformations based on the Atlas Transformation Language (ATL) (Jouault and Piers 2009). Figure 5-42 illustrates the interaction between the *TransformService* and various implementations of the *TransformEngine* interface for processing different types of transformations requested by a client. Thereby, the *TransformService* acts as mediator between clients and various *TransformEngines*, thus enabling transparent management and execution of transformations across multiple engines and transformation languages. New engines can be flexibly incorporated by adding their name, type and URL to the Transform Service's configuration file.



Figure 5-42: Interaction between Clients, the Transform Service and Various Engines

5.5.1.2 The XSLT Transformation Engine

The XSLT-based transformation engine was used for all model transformations presented in this thesis. It was originally developed in the context of a study thesis (Schmid 2006) and allows the specification and execution of composite transformations. Based on a declarative XML-based configuration language, a sequence of multiple XSLT-based transformations can be defined. Beyond that, also code-based transformations in form of plugins can be integrated as pre- or post-processing steps. The transformation engine realizes the sequential execution of plugins and XSL transformations and ensures the correct transfer of input and output documents.

In the context of this thesis, the engine was extended by a Web service endpoint according to the previously described *TransformEngine* interface. Furthermore, a plugin for transformations from Microsoft Visio's DatadiagramML format was developed. As this format does not include graphical composition-related relationships between shapes, the plugin is used as a preprocessing step and computes hierarchical *is-contained-in* relationships based on shape coordinates (Orozov 2008). Furthermore, post-processing plugins computing an adequate layout for business process diagrams were implemented (Setiawan 2009). They are based on the algorithm presented in 5.4.3.2 and are required for transformations from the DSM to a DIM. By adopting this engine for the Workflow DSL's model transformation framework, the criticism that XSL transformations cannot be developed in a modular and thus comprehensible way was effectively met.

5.5.1.3 Transformation Manager Web Application

The Transformation Manager was implemented as a Web application supporting the management and execution of model transformations. It presents an alternative to the direct integration of model transformations into existing tools and systems via the Transform Web Service. Hence, the Transformation Manager provides a Web-based user interface for the operations offered by the Transform Web Service. Thus, users can view the set of available formats and their associated mappings to the Core Elements Set (CES). New formats can be added and their CES mapping specified as well as the available model transformations be viewed, whereby both direct and transitive transformations are listed (cf. Figure 5-43). The application also guides users through the design and implementation process for new model transformations. Finally, the Transform Manager allows users to execute model transformations by selecting the desired transformation, providing the source document and receiving the transformation result. An in-depth presentation of the so-called *Poseidon Transformation Manager* can be found in the diploma thesis of Nikolay Orozov (Orozov 2008).

The screenshot shows the Poseidon Transform Manager interface in a Mozilla Firefox browser. The page title is "Poseidon Transform Manager - Transformations - Mozilla Firefox". The main content area is titled "Stored Transformations" and contains a table with the following data:

Name	Type	Description	Source Set	Target Set	
BPMN2XPDL	XSLT	Abbildung von BPMN nach XPDL 2.0 zur Unterstützung des MWRG-Vorgehensmodells.	BPMN	XPDL	Details
PetriNets2XPDL	XSLT	Abbildung von Petri Nets nach XPDL 2.0 zur Unterstützung des MWRG-Vorgehensmodells.	Petri Nets	XPDL	Details
SSO2XPDL	XSLT	Abbildung von SSO nach XPDL 2.0 zur Unterstützung des MWRG-Vorgehensmodells.	SSO	XPDL	Details
UMLActivity2XPDL	XSLT	Abbildung von UMLActivity nach XPDL 2.0 zur Unterstützung des MWRG-Vorgehensmodells.	UMLActivity	XPDL	Details
XPDL2BPMN	XSLT	Abbildung von XPDL 2.0 nach BPMN zur Unterstützung des MWRG-Vorgehensmodells.	XPDL	BPMN	Details
XPDL2PetriNets	XSLT	Abbildung von XPDL 2.0 nach Petri Nets zur Unterstützung des MWRG-Vorgehensmodells.	XPDL	Petri Nets	Details
XPDL2UMLActivity	XSLT	Abbildung von XPDL 2.0 nach UMLActivity zur Unterstützung des MWRG-Vorgehensmodells.	XPDL	UMLActivity	Details
XPDL2XOML	XSLT	Transformation von XPDL 2.0 nach XOML zur Automatisierung des MWRG-Ansatzes.	XPDL	XOML	Details

Below the "Stored Transformations" table, there is a section for "Transitive Transformations" with the following table:

Name	Source Set	Target Set	Stored Transformations
BPMN2Petri Nets	BPMN	Petri Nets	1. BPMN2XPDL. 2. XPDL2PetriNets

The browser window also shows a sidebar with navigation links: Home, Show Core Element Sets, Create Core Element Set, Show Transformations, Create Transformation, and Start Transformation. The status bar at the bottom indicates "Fertig".

Figure 5-43: List of Available Transformations in the Transformation Manager

5.5.2 Workflow Execution Platform

The Workflow Execution Platform comprises various components for the management and execution of workflow-based Web applications. On the one hand, this includes the *Workflow Web Service* which enables external applications and systems to manage and execute workflow instances and definitions. To this end, it integrates a *Workflow Engine* and is capable of performing its correct instrumentation based on a Workflow DSL program. Therefore, it uses the *Transform Web Service* in order to transform the Workflow DSL program into the workflow execution language required by the Workflow Engine.

On the other hand, one or more *Web portal frameworks* provide adequate *Web-based user interfaces* based on the Workflow DSL program and realize the communication with the Workflow Web Service. These user interfaces support the management of workflows and open tasks for the current user and particularly the Web-based processing of workflow activities.

5.5.2.1 Workflow Web Service

Figure 5-44 shows the public interface of the Workflow Web Service which comprises operations for managing workflow definitions and instances as well as operations supporting the distributed execution of workflow activities by external client applications.

The CRUDS methods *Create*, *Read*, *Update*, *Delete* and *Search* support the *management* of workflow definitions and instances. As suggested by the canonical CRUDS interface concept (Nussbaumer 2008), the XML namespace of the submitted parameter indicates whether the operation shall work on a workflow definition or a workflow instance. Workflow definitions are Workflow DSL programs which are stored in the *Workflow Repository* and can thus be submitted, updated, deleted and retrieved.

When a new or modified workflow definition is submitted, it has to be transformed into a *Workflow Assembly* embodying the executable workflow specification in the format required by the *Workflow Engine*. The current implementation uses the *Windows Workflow Foundation's Workflow Engine* which expects workflow specifications in the *XOML* format (cf. Section 5.4.4). Hence, when the Workflow Web Service's *Create* operation is invoked with a Workflow DSL program as parameter, the service calls the *Transform Web Service* and requests a transformation of the Workflow DSL program into XOML. Then, the Workflow Web Service passes the received XOML document to the Workflow Foundation's *workflow compiler* and stores the resulting *Workflow Assembly* in a dedicated repository.

Based on such a Workflow Assembly, new *workflow instances* can be instantiated via the Workflow Web Service's *Create* method. Therefore, the method's implementation triggers the *Workflow Engine* to create a new instance of the specific workflow type. The *Delete* method enables the termination of a running workflow instance. The Workflow Web Service stores basic metadata for each workflow instance including its name, description and associated workflow assembly type. This metadata is provided via the *Create* method, can be retrieved via *Search* and modified via the *Update* method.

The second set of operations provided by the Workflow Web Service supports the communication with task-specific (Web-based) client applications in the course of *processing workflow activities*. Similar to the operations described above, the Workflow Web Service acts again as *mediator* between requesting clients and the Workflow Engine. Thus, different workflow engines could be transparently integrated or the existing infrastructure seamlessly scaled out. Regarding the communication sequence between clients and the Workflow Web Service, usually the first step lies in requesting a list of active tasks for a given workflow type or instance and a given user (*GetTaskList*). Based thereupon, a task is selected for processing on the client side which results in requesting the corresponding input parameters from the workflow (*GetTaskData*). After the task has been completed on the client side, the output of the task is sent back to the workflow and the task is marked as completed (*CommitTaskData*). A subsequent call of the *GetTaskList* operation returns a new set of active tasks which reflects the completion of the previous activity and potential data-related control-flow branchings. Thus, diverse

and possibly distributed clients can navigate through the workflow based on the described operations sequence. Therefore, the *Workflow Engine* transparently ensures the correct control-flow and role-based task distribution. Furthermore, it ensures that idle workflow instances are persisted to the *Persistence Database* and are resumed when needed. A detailed description of the Workflow Web Service's implementation for the integration of the Windows Workflow Foundation's engine can be found in (Buck 2007).

```

/* Part 1: Operations for Managing Workflow Definitions and Instances*/
public Status Create(XmlElement prototype);
public XmlElement Read(XmlElement readContext);
public Status Update(XmlElement updateContext, XmlElement element);
public Status Delete(XmlElement deleteContext);
public XmlCollection Search(XmlElement searchContext);

/* Part 2: Operations related to the Execution of Workflow Activities */
public XmlCollection GetTaskList(string InstanceID,
                                string WorkflowType,
                                string Identity);
public XmlElement GetTaskData(string InstanceID,
                              string Activity,
                              string Identity);
public XmlElement CommitTaskData(string InstanceID,
                                 string Activity,
                                 string Identity,
                                 XmlElement Data);

```

Figure 5-44: Public Interface of the Workflow Web Service

The operation *GetTaskList* returns the list of active tasks for the given *Identity*, whereby the scope lies either on a particular workflow instance (*InstanceID*) or on all active instances of a particular type (*WorkflowType*). Therefore, based on the *Users & Roles* directory, the service performs a mapping between the given *Identity* and a set of predefined roles. Depending on the determined roles, the Workflow Engine's *Tracking Database* is queried and returns the set of active workflow activities. This rather basic identity mapping concept was designed open towards adopting existing, more comprehensive authentication and authorization concepts. Particularly in the context of cross-organizational workflow scenarios, federative security concepts seem promising (Meinecke, Nussbaumer and Gaedke 2005).

The operation *GetTaskData* provides requesting clients with information required for the processing of a particular *Activity* within a specific workflow instance (*InstanceID*). The *Identity* parameter serves as basis for adequate authorization strategies preventing unauthorized users from retrieving workflow data. The returned data corresponds to the workflow activity's input parameters as specified in the workflow model.

The completion of a workflow activity and the submission of corresponding output data are realized via the *CommitTaskData* operation which receives, besides the

XML-based *Data*, the *InstanceID* of the workflow instance as well as the identifier of the respective *Activity* and *Identity* as parameters.

5.5.2.2 Core Portal Component: Workflow DSL Solution Building Block (SBB)

According to the *Web Engineering DSL Framework*, each DSL comprises a dedicated *Solution Building Block (SBB)* being capable of executing the DSL's programs. Hence, the Workflow DSL's SBB forms the central technical component which can be configured with a Workflow DSL program at runtime. Thereupon, it submits the Workflow DSL program to the *Workflow Web Service* in order to store the new workflow definition in the repository and having an associated Workflow Assembly generated. Furthermore, it constructs corresponding *Web-based user interfaces* as specified by the *Application* declarations in the Workflow DSL program. Finally, the SBB realizes the communication with the Workflow Web Service for *managing* and *executing workflows* and particularly for *processing workflow activities*. The Workflow DSL SBB was developed in form of a software component for the WebComposition Service Linking System (WSLS), i.e. the Web Engineering DSL Framework's technical platform (cf. Section 4.1.2). However, implementations for other Web portal frameworks, e.g. Microsoft Office SharePoint Server 2007 or IBM WebSphere Portal Server, are conceivable in an analog way.

For the SBB's sufficient *configuration*, only two properties have to be configured: The URL of a Workflow Web Service instance as well as a Workflow DSL program conforming to the Workflow DSL's DSM. Concerning the construction of Web-based user interfaces, the SBB parses the *Application* specifications contained in the Workflow DSL program and instantiates a corresponding *Activity Building Block (ABB)* for each application. Each ABB instance is provided with an initial configuration set which is also derived from the *Application* specification in the DSL program. In addition, an instance of a *Workflow List* component and an instance of an *Activity List* component are created. All of these dynamically created instances are considered subordinate to the Workflow SBB, both from a visibility and lifecycle perspective. Figure 5-45 illustrates this assembly process.

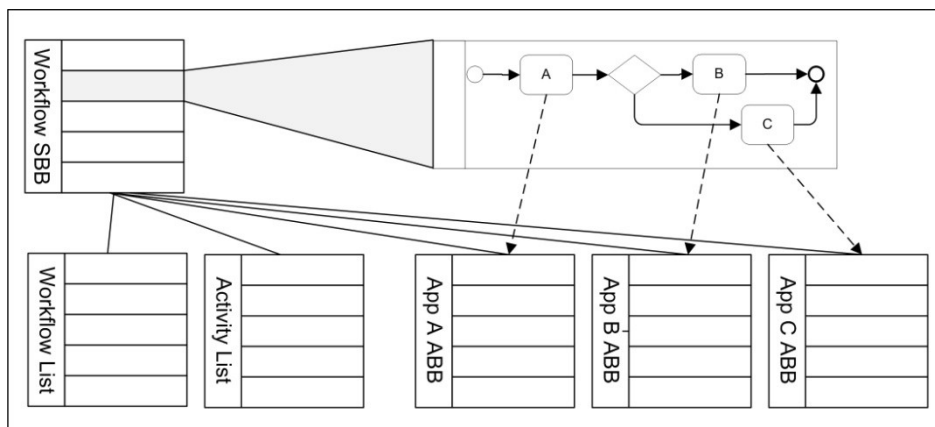


Figure 5-45: Assembly of Application Building Blocks According to the Application Specification in the Workflow DSL Program

In the course of a workflow's execution, the SBB sets the ABB instance corresponding to the current workflow activity visible. By default, the *Workflow List* is visible. This component shows all available workflow instances to a user. Therefore, it performs a *Search* operation on the Workflow Web Service. Furthermore, the *Workflow List* enables the creation, deletion and modification of workflow instances or their metadata respectively – provided that the user has sufficient privileges. This functionality is likewise realized via the Workflow Web Service's corresponding operations *Create*, *Delete* and *Update*. When a user selects a workflow instance in the *Workflow List*, an instance-specific task list, realized by the *Activity List* component, is presented. Therefore, the Workflow Web Service's *GetTaskList* operation is used. Similarly, the *Activity List* component allows users to view their active tasks across all workflow instances. These two components, *Workflow List* and *Activity List*, are accessible via corresponding hyperlinks at any time as shown in Figure 5-46. The figure depicts a screenshot of a Workflow SBB instance for the 'business trip' example scenario. The marked *Presentation Places* (indicated by dashed or dotted rectangles) illustrate the above-mentioned concept of dynamically presenting subordinated components in an *Inner Presentation Place* encapsulated by the Workflow SBB. In the screenshot, the *Workflow List* component displaying active workflow instances is currently set to visible.

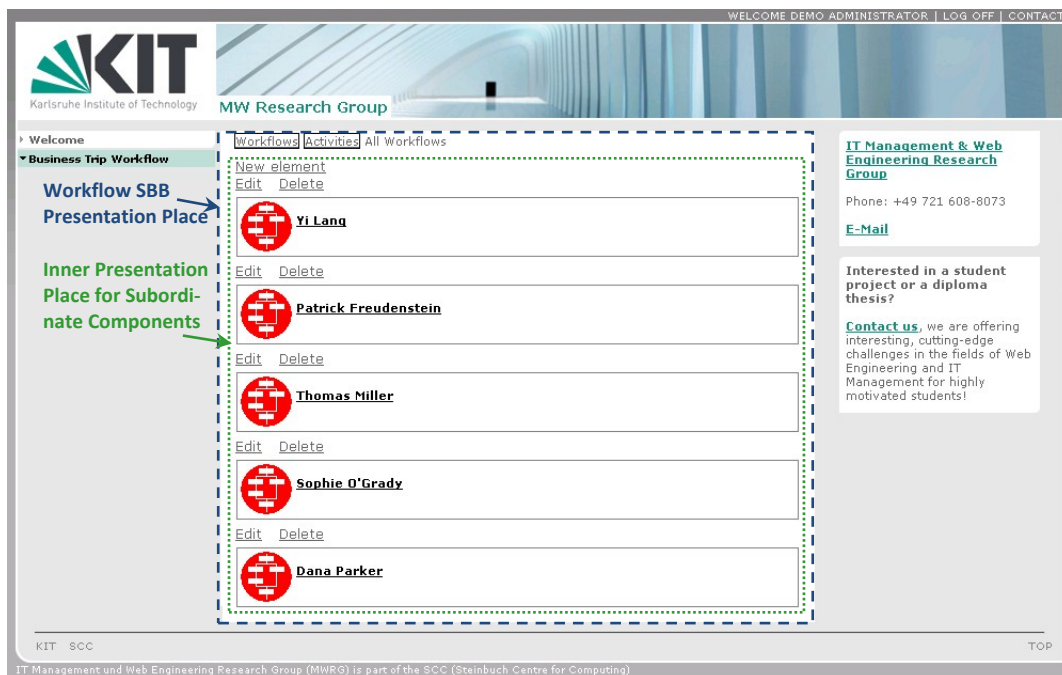


Figure 5-46: Screenshot of the Workflow List in the Business Trip Example Scenario

The communication between the Workflow SBB and the subordinated components which have been created and configured in the initial setup procedure is founded on a *generic token-based event mechanism*. Tokens contain a *type identifier* as well as *data payload*. Based on a generic token-specific event handler, the Workflow SBB in its role as *Token Container* receives tokens which were raised by a subordinated component. Depending on the token type and the contained data payload, the Token Container performs corresponding actions. In the most cases, this includes

replacing the currently visible component by another subordinated component. Before the Token Container sets a component visible, it informs the component by raising a corresponding event named *PreToken* and passing data payload used for the component's initialization. Accordingly, after the component was set visible, the Token Container raises the *PostToken* event. The set of possible states and the token-type-based transitions between them could be formalized in form of a finite state machine.

In the following, the application of this generic token-based event mechanism to the communication between the Workflow SBB and its subordinated components is exemplarily described:

- When a user selects a workflow instance in the *Workflow List* (cf. Figure 5-46), it raises a token with the type identifier *WorkflowSelected* and the workflow instance ID stored in the token's payload.
- The token is received by the *Token Container*, i.e. the *Workflow SBB*, which evaluates the type identifier and accordingly requests the current task list for this instance via calling the *GetTaskList* operation of the *Workflow Web Service*. If the returned task list contains more than one active task, the *Workflow SBB* removes the *Workflow List* from its *Inner Presentation Place* and instead inserts the *Activity List*. Before setting it visible, the *Workflow SBB* raises the *PreToken* event on the *Activity List* and passes the set of active tasks received from the *Workflow Web Service* as payload. If the returned task list contains only a single activity, the *Activity List* is skipped and the procedure continues with the step after the next.
- The *Activity List* renders this list and thereby allows the user to select a particular activity for processing. Upon selection, the *Activity List* raises a token with the type identifier *ActivitySelected* and the identifiers of the selected activity and the associated workflow instance as payload.
- The *Token Container* either extracts the identifiers of the activity and its associated workflow instance from the payload of the token raised in the previous step or, in the case of a single active task where the *Activity List* was skipped, uses the values of this single activity. Thereupon, according to the token's type identifier, it invokes the *Workflow Web Service's* operation *GetTaskData* and receives the input parameters for the respective activity. Then, based on the *Workflow DSL* program, the *Workflow SBB* determines the corresponding subordinated component which is assigned as application for the considered activity and inserts it into the *Inner Presentation Place*. It passes the received input parameters via the *PreToken* event to the component and subsequently sets it visible.
- Based on this activity-specific application component, e.g. an instance of the *Dialog-based User Interaction ABB*, the user can now process the workflow activity, e.g. fill out a dialog. Upon completion, the component raises a token with the type identifier *Update* and the task output data, e.g. a filled data model, as payload.

- The *Token Container* receives the token, evaluates its type identifier, and accordingly invokes the Workflow Web Service's operation *CommitTaskData*, whereby it passes the received output data. Subsequently, it requests the new task list for the current workflow instance via the *Workflow Web Service's GetTaskList* operation. Then, the procedure continues as described above in step two.

Figure 5-47 illustrates the described transitions between the subordinated components within the *Workflow SBB's Inner Presentation Place* in form of a finite state machine graph. The state machine's active state indicates the currently visible subordinated component. The *ABB Instance* state represents all subordinated components which are instances of an ABB and serve as applications for the Web-based realization of one or more activities. Hence, the transition from the *ABB Instance* state to itself should be (in the most cases) interpreted as a transition between two different ABB instances.

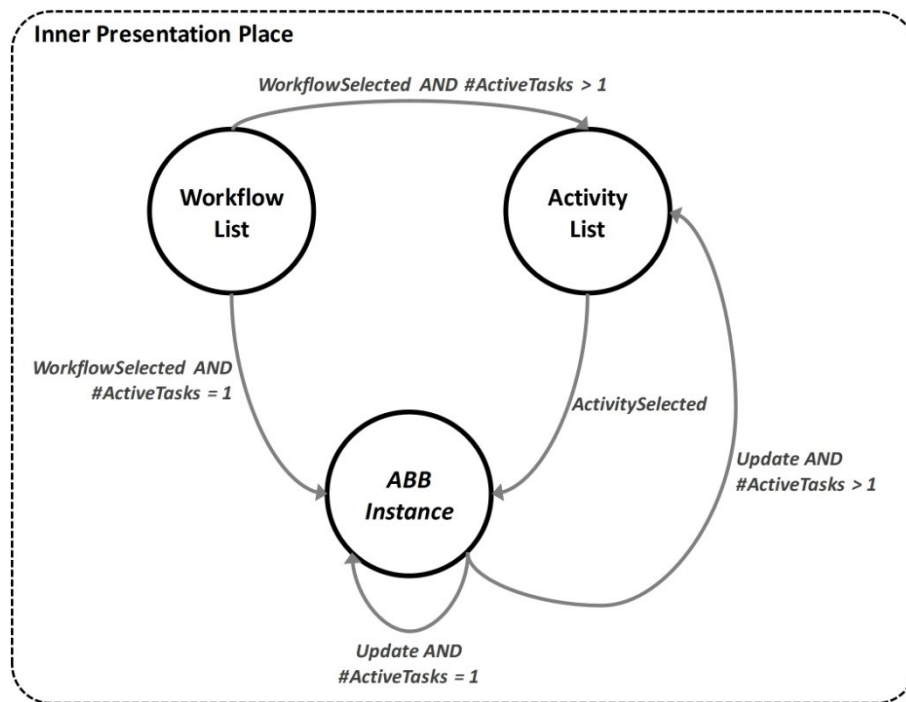


Figure 5-47: Transitions between the Currently Visible Subordinated Components within the *Workflow SBB's Inner Presentation Place*

5.5.3 Automated Application Construction: From Modeling to Execution

Based on the preceding presentation of the various technical components, this section describes their interplay in the course of a workflow-based Web application's automated construction ranging from modeling to workflow execution. The described steps are also marked by corresponding numbers in Figure 5-39. The presentation uses the running example of this thesis, the 'business trip' scenario.

- **Phase 1 – DIM-to-DSM Transformation:** Having completed an iteration of the business process and workflow modeling activities, the resulting model is transformed into its DSM-based representation. This can be achieved either directly from within the modeling tool via an integration of the *Transform Web Service* or manually using the *Transform Manager Web Application*. The result of this phase is a DSM-based Workflow DSL program. Figure 5-48 shows a screenshot of the *Transform Manager Web Application* with the obtained transformation result, i.e. the Workflow DSL program. This can now be copied to the clipboard or saved into a file.

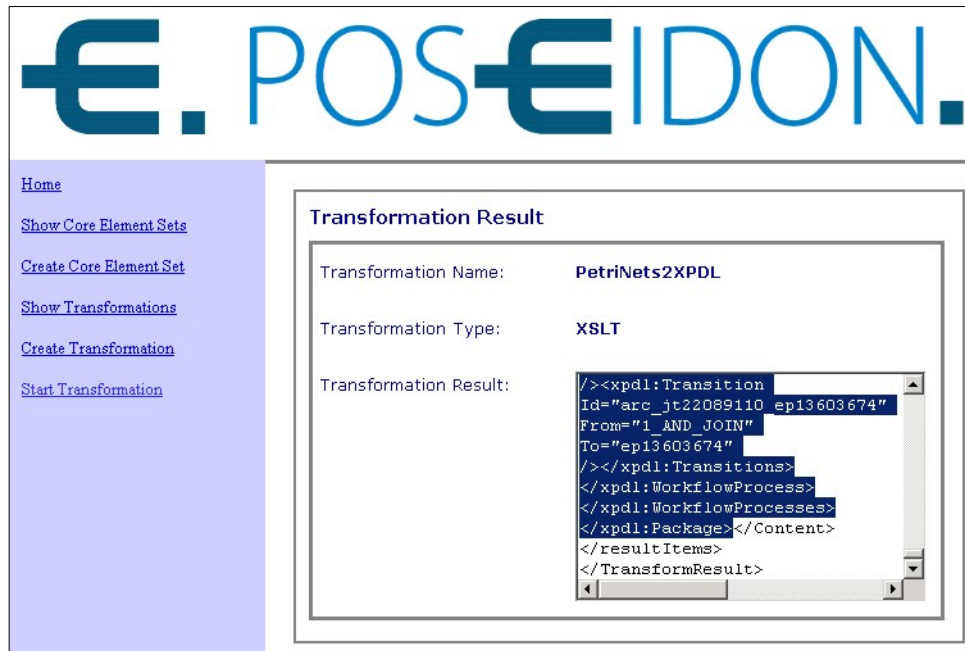


Figure 5-48: DIM-to-DSM Transformation via the Transform Manager

- **Phase 2 – Workflow SBB Instantiation and Configuration:** In this phase, a new instance of the *Workflow SBB* is added to a Web page and configured with the *Workflow Web Service's* URL and the Workflow DSL program obtained from the previous phase. Figure 5-49 shows the configuration properties for a *Workflow SBB* instance within the *WSLS Web Portal Framework*. The property named *Configure* is used to explicitly initiate a new automated application construction process which is described in the subsequent phases.

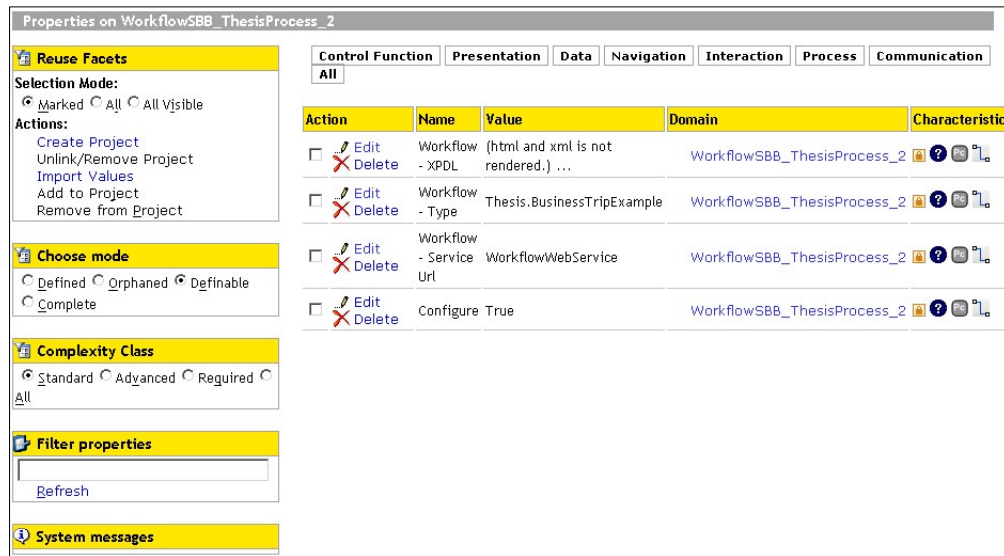


Figure 5-49: Configuring a Workflow SBB Instance in the WSL Framework

- **Phase 3 – Registration at the Workflow Web Service:** The *Workflow SBB* invokes the *Workflow Web Service's* *Create* operation and passes the configured Workflow DSL program. Thereupon it is stored in the *Workflow Repository*, thus being retrievable for other clients.
- **Phase 4 – Instrumentation of the Workflow Engine:** Subsequently, the *Workflow Web Service* invokes the *Transform* operation of the *Transform Web Service*, thereby requesting a transformation of the Workflow DSL program into the workflow execution language required by the *Workflow Engine*. The received transformation result is automatically compiled into a *Workflow Assembly* and stored in the corresponding repository. Finally, a manual mapping of existing *users and roles* to the roles specified in the workflow may be necessary in some cases.
- **Phase 5 – Application Assembly:** On the *Web Portal Framework* side, the *Workflow SBB* parses the application specifications contained in the Workflow DSL program and assembles corresponding *Activity Building Blocks (ABB)* as subordinated components. Each ABB is provided with an initial configuration set according to the respective application specification. If a matching ABB instance already exists, only the changed properties are adopted. In addition, the *Workflow SBB* instantiates the *Workflow List* and *Activity List* components serving for the management of workflow instances and displaying a personalized list of active tasks to the current user respectively. Therewith, the automated application construction process is completed and users can start to create new workflow instances and process them. Figure 5-50 exemplarily shows a part of the Web-based dialog which was automatically generated by the *Dialog-based User Interaction ABB* and which supports the *Create Expense Report* activity of the 'business trip' example process.

Figure 5-50: Workflow Execution – The Create Expense Report Activity

- **Detailed Design of ABB Instances:** Although the automatically constructed Web-based workflow is fully functional, a detailed design of the *ABB Instances* supporting the Web-based processing of workflow activities is desirable. For example, the dialog shown in the figure above enables entering the required information and submitting it back to the workflow. However, it could be improved regarding usability and aesthetical aspects. To this end, the Workflow DSL approach allows the detailed design of *ABB Instances* at runtime using the ABB-specific DIMs and associated editors. Such changes apply directly to both new and already running workflow instances. An example showing the detailed design of the above *Expense Report* dialog based on the *Dialog-based User Interaction ABB*, i.e. the *Dialog DSL*, is presented in Chapter 6.

5.5.4 Support for Federative Scenarios

In advanced scenarios, workflows do not stay within the boundaries of an organization but may rather span multiple organizations including their respective systems and employees. Against this background, Web Service-orientation was identified as a crucial requirement for the workflow execution platform in order to establish a foundation for such federative scenarios (cf. Section 2.2.1). Hence, in the following, the presented workflow execution platform's federation enablement shall be briefly outlined.

Figure 5-51 illustrates possible cross-organizational usage patterns of the Workflow DSL approach's workflow execution platform. In this context, federation can be considered from two perspectives: On the one hand, integrating *systems* from diverse external organizations and, on the other hand, enabling *participants* from various organizations to collaborate in the workflow using various client applications.

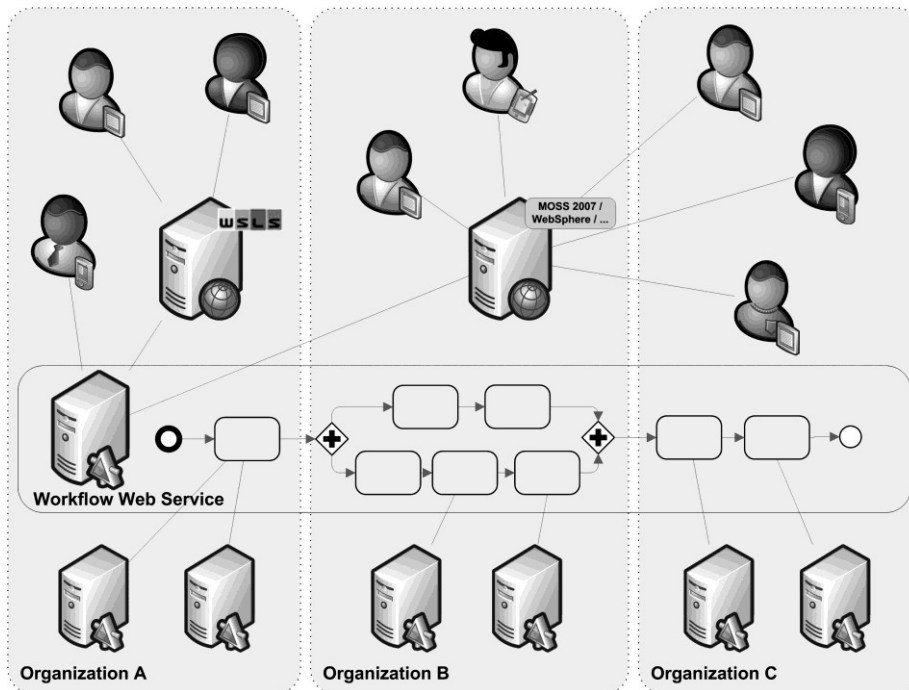


Figure 5-51: Support for Cross-Organizational Web-Based Workflow Scenarios

With respect to the integration of external Web Service-enabled systems, the *Web Service Communication ABB* can be employed. It can be mapped to workflow activities, either in combination with other ABBs or solely, and allows the secured communication with Web Service endpoints according to a given WS-SecurityPolicy specification. Based on that, also specialized federative security concepts, e.g. according to the WS-Federation standard, could be integrated into the Web Service Communication ABB (Meinecke, Nussbaumer and Gaedke 2005; Lockhart, Andersen, Bohren et al. 2006).

Concerning the cross-organizational collaboration of distributed participants, the *Workflow Web Service* forms the central solution element. Besides being used by a *Workflow SBB* running on the *WLS framework* as presented in the previous sections, it can analogically serve arbitrary client applications or portal frameworks respectively. Thus, particular activities can also be processed using specialized task-specific applications, e.g. a spreadsheet application. Via the *Workflow Web Service*, such heterogeneous clients can retrieve workflow specifications, i.e. Workflow DSL programs, and accordingly construct adequate user interfaces. In this regard, they benefit from the fact that the Workflow DSL's formalized schema, i.e. the DSM, was founded on a widely-adopted standard. Furthermore, clients can consume all of the *Workflow Web Service's* operations for managing and executing workflows. Thus, from a technical perspective, workflow client applications as well as the

corresponding users can reside in different organizational realms. Similar to the integration of systems described above, the incorporation of specialized federated security concepts could be necessary also in this context and is inherently supported by the platform's service-oriented architectural design.

5.6 Summary

This chapter presented the Workflow DSL, a novel approach for the fully model-based construction of workflow-based Web applications. It was designed in accordance with the previously introduced Web Engineering DSL Framework and places strong emphasis on the effective and continuous involvement of stakeholders. In the following, the approach's unique solution elements as well as the fulfillment of the requirements elaborated in Chapter 2 are briefly summarized.

The Workflow DSL approach bridges the gap between existing commercial workflow execution platforms on the one hand and the need for Web-based user interfaces for the efficient and effective processing of human tasks on the other hand. Therefore, the Workflow DSL's *Domain-Specific Model (DSM)* was founded on the *XML Process Definition Language (XPDL) standard* and well-defined extensions towards specifying Web-based user interface aspects were introduced. Regarding the latter, a catalog of generic *Activity Building Blocks (ABBs)* was presented. An ABB embodies a certain type of Web-based support for the realization of workflow activities like *Dialog-based User Interaction*, *Data Presentation* or *Web Service Communication*. For each ABB, the *minimal configuration set* required for executing the desired type of behavior was described. The ABBs and their minimal configuration sets constituted the basis for the Web-specific extensions mentioned above. Moreover, they represent a valuable contribution also beyond the scope of the Workflow DSL approach. As they form the basis for the implementation of *highly reusable and generic Web-based software components* embodying the respective behavior type, their applicability and utility extends to the development of Web-based solutions in general. The ABBs were designed in accordance with the *Web Engineering DSL Framework*. Thus, their detailed design can be conducted at runtime, i.e. after a basic but already fully functional Web-based workflow has been set up. In summary, the resulting DSM forms a novel, *standard-based foundation* for a holistic and continuous specification of *workflow execution and Web-based user interface* aspects.

The Workflow DSL allows the incorporation of various *Domain Interaction Models (DIMs)* and associated editors for the model-based specification of Workflow DSL programs. Thus, learning efforts can be reduced and stakeholders can employ the modeling notation and tools they already know. As all DIMs work on a *single shared Workflow DSL program*, a *novel degree of model continuity and integrity* throughout the complete development lifecycle is achieved. Thus, the presented approach allows the *incremental, completely model-based* specification of Workflow DSL programs from initial requirements engineering to business process and workflow modeling to workflow execution. In this chapter, four DIMs and corresponding tools

were exemplarily presented. First, a custom table-based notation named *Simple Sequence Only (SSO)* supported by Microsoft Word and tailored at initial requirement engineering activities. Furthermore, three additional *standard* business process modeling notations and supplemental tools supporting the phases *Business Process Modeling* and *Workflow Modeling* were presented: *The Business Process Modeling Notation (BPMN)* and Microsoft Visio, *UML 2.0 Activity Diagrams* and IBM Rational Software Architect, as well as *Petri Nets* and *INCOME2010*. The adoption of these notations and tools as DIMs for the Workflow DSL was based on extensions to *standardized (notation-specific) model interchange formats*. Thus, other tools adhering to these standards could be used as well.

The modeling of Web-based workflows using these DIMs was designed in a *stakeholder-oriented way*, thus focusing on the business process structure and *hiding unwanted technical complexity*. The minimal technical specification sets required by the ABBs combined with the ability to perform a detailed design at runtime present a *lightweight alternative* to existing heavy-weight, developer-oriented modeling approaches.

The emerging multitude of DIMs and associated serialization formats allowing the cross-notational specification of a single DSM-based Workflow DSL program leads to the necessity of adequate *bilateral model transformations*. To this end, a novel *model transformation framework* was introduced. Facing the so far unsolved challenge of integrating heterogeneous business process modeling languages, the approach strikes a new path by introducing the *Core Elements Set (CES)* concept. The CES defines a *set of common business process and workflow concepts* which *abstracts* from an individual notation or language. Although the CES is not intended to provide full coverage for all theoretically possible modeling constructs, it establishes sufficient support for the great majority of scenarios occurring in practice. This was confirmed by an own empirical evaluation of business process models presented in Section 8.1.1 as well as by similar recently published empirical studies on usage distributions of business process modeling concepts. Based on the CES concept, *semantic congruence* between heterogeneous business process modeling languages can be achieved and thus *lossless, bilateral model transformations* be realized. In this chapter, two challenging examples for a *horizontal* and a *vertical* model transformation were described. While the former concerns bilateral transformations between the *Petri Net DIM* and the DSM, the latter addresses the transformation from the DSM into the *workflow execution language XOML*. In both cases, the notation-specific mappings to the CES as well as the transformation's technical realization were discussed. Based on these examples and the presented solutions, further transformations for new DIM notations and tools can be *systematically* and *non-invasively* realized. Finally, a *complete mapping catalog* between the Workflow DSL's various DIMs and the DSM was presented.

The Workflow DSL's technical platform comprises the technical support platform for the *model transformation framework* as well as the *workflow execution platform*. The former supports the management and execution of model transformations both by *external systems* and *humans* respectively. It supports the *flexible incorporation* of transformation language-specific engines and transparently provides both direct and *transitive* transformations.

The *workflow execution platform* realizes the automated construction and Web-based execution of fully operational long-running workflows spanning diverse roles and systems. It was designed in a strongly *service-oriented* way, thus providing proficient support both for *multimodal participation* and *federative scenarios*. It encapsulates a state-of-the-art *Workflow Engine* and exposes management- and execution-related operations via a well-defined Web Service-based endpoint. The Workflow DSL's *Solution Building Block (SBB)* serves as central solution component within Web portal frameworks and manages the *fully automated application construction* process according to a Workflow DSL program. During the execution of workflow instances, it mediates between the Web service endpoint encapsulating the *Workflow Engine* on the one hand and activity-specific *Web-based user interfaces* on the other hand. The latter are set up by assembling and configuring *Activity Building Block (ABB)* instances during the automated construction process. Due to their inherent characteristics as *DSLs*, their detailed design, e.g. regarding usability and aesthetical aspects, can be performed at runtime.

The presented solutions provide a sound basis for an *agile* and *evolution-oriented development process* as presented at the beginning of this chapter. On the one hand, the completely model-based and fully automated construction approach enables *rapid development cycles*. On the other hand, lossless model transformations as well as the unique degree of model continuity preserve *consistency* throughout all phases. Thus, *changes* can be *efficiently adopted* on a model basis and propagated to the existing application or directly performed at runtime using the ABB's respective DSLs.

All of the presented solution elements of the Workflow DSL approach were implemented and used for the practical realization of various workflow-based Web applications. Some of the Workflow DSL's key concepts were also adopted in the KIM Project (Juling 2005) for the efficient model-driven construction of page-flow-based portal features. More information about the Workflow DSL's empirical and practical evaluation is presented in Chapter 8.

6 Constructing Advanced Web-based Dialogs⁴

The efficient and effective construction of advanced dialog-based user interfaces plays an important role in the development of workflow-based Web applications. Consequently, the *Dialog DSL* presented in this chapter forms a central pillar of the previously introduced *Activity Building Blocks (ABBs)* supporting the Web-based processing of workflow activities. The Dialog DSL was designed according to the *Web Engineering DSL Framework* and comprises models, tools and an evolutionary methodology for the model-based construction of complex and highly interactive dialog components. It explicitly addresses the requirements identified in Chapter 2 and places particular emphasis on *usability* aspects and *effective stakeholder involvement*. The Dialog DSL enables considerable efficiency gains and is excellently applicable by both developers and stakeholders. This was successfully confirmed by a *formal empirical evaluation* presented in Chapter 8.

6.1 The Dialog DSL at a Glance

The Dialog DSL is an executable specification language tailored to the domain of dialog-based user interaction in the Web. It comprises a two-tiered Petri net-based *Domain Interaction Model (DIM)* and an associated Web-based model editor, both strongly focusing on simplicity and hiding technical complexity. Its *Solution Building Block (SBB)* is capable of automatically generating fully operational dialogs based on given data schemas or Web service specifications. Furthermore, it supports runtime model adaptations according to characteristics of requesting client devices as well as renders models into executable dialog-specific markup languages such as the W3C XForms standard (Boyer, Dubinko, Leigh L. Klotz et al. 2007). An agile and evolution-oriented development methodology complements the Dialog DSL approach.

⁴ Parts of this chapter have been published in (Freudenstein and Nussbaumer 2008a; Freudenstein and Nussbaumer 2008b; Freudenstein, Nussbaumer, Allerding et al. 2008)

6.1.1 Elements of the Dialog DSL

According to the *Web Engineering DSL Framework*, the Dialog DSL consists of three core elements:

- **Domain-Specific Model (DSM):** According to the *Web Engineering DSL Framework*, the DSM embodies the formalized schema for all dialogs that can be specified with the Dialog DSL. Accordingly, the Dialog DSL's DSM comprises two groups of core concepts: On the one hand, concepts for describing *Interaction Elements*, which were specified based on the *W3C XForms standard*. On the other hand, concepts for specifying dynamic behavior of a dialog, so-called *Interaction Structures*. *Dialog Partitions* serve as container elements for semantically cohesive *Interaction Elements* and provide the basis for modeling *Interaction Structures* between them. The DSM provides well-defined *extension points* for systematically incorporating additional *Interaction Elements* or *Interaction Structures*.
- **Domain Interaction Model (DIM):** So far, the Dialog DSL provides a two-tiered, Petri net-based DIM notation in accordance with the DSM. On the first tier, the elements from the data model or associated *Interaction Elements* respectively are distributed on various *Dialog Partitions* and dynamic behavior between them using *Interaction Structures* is modeled. *Dialog Partitions* are represented by Petri net *places* containing *Interaction Elements* which are bound to the data model. Petri net *transitions* correspond to the performed user interaction, i.e. changing a value in the dialog's data model or triggering an action. *Interaction Structures* are represented by predefined graphical Petri net *templates*. On the second tier, the concrete *appearance* of each partition based on *Interaction Elements* is specified. With respect to device-dependent model adaptations at runtime, dedicated symbols allow for marking partitions and groups of *Interaction Elements* as non-dividable. This two-tiered modeling approach fosters reuse and allows for separation of concerns - thus improving its usability and simplicity. A supporting *Web-based DIM editor* allows for the comfortable creation and adaptation of dialog models.
- **Solution Building Block (SBB):** The Dialog DSL's SBB forms the core of the technical platform. It communicates with a *Dialog Web Service* for initiating the generation of raw dialog models based on a given data schema or for reusing dialogs. Moreover, it links to *the Web-based model editor* for creating and adapting dialogs. Finally, the SBB identifies requesting user agents at runtime and performs corresponding dialog adaptations as well as ultimately transforms dialog models into executable markup, e.g. according to the *W3C XForms standard*.

6.1.2 Evolutionary Process Model

Figure 6-1 illustrates the Dialog DSL's associated process model for the construction of advanced Web-based dialogs. It consists of three phases in the course of a continuous evolution.

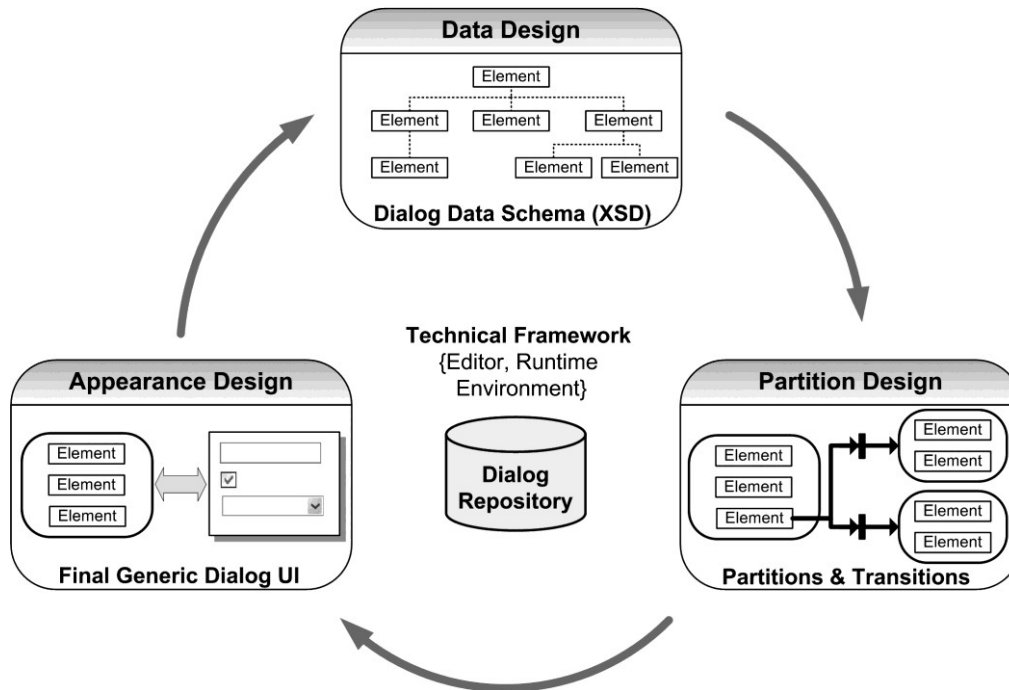


Figure 6-1: Overview of the Evolutionary Dialog Engineering Methodology

- **Data Design:** In this phase, the data model for the dialog being constructed is developed. This can be achieved in several ways: Firstly, a suitable data model can be retrieved via the Web Engineering Reuse Sphere (cf. Chapter 7). Therefore, sophisticated search mechanisms considering criteria from the development context, e.g. the type of application being developed or the workflow context the dialog is part of, could be employed. Secondly, if the data entered in the dialog shall be submitted to a Web service, the target data schema can be extracted from the Web service's WSDL document. Thirdly, the data schema can be elaborated from scratch in strong collaboration with the involved stakeholders, ideally supported by an elicitation tool. The output of this phase is an XML Schema document specifying the dialog's data schema. Based on this schema, the Dialog DSL's technical framework is already able to construct a fully operational dialog that can be directly used in production or further refined using the Web-based editor at runtime.
- **Partition Design:** This phase addresses the modeling of *Dialog Partitions* and dynamic behavior based on *Interaction Structures*. Therefore, in the first step, the elements from the dialog's data schema are distributed on several *Dialog*

Partitions, each of them representing a semantically cohesive dialog unit, e.g. personal data, travel itinerary or expenses. Then, employing predefined *Interaction Structures* like *Sequence* or *Choice*, dynamic transitions between these partitions are defined. Due to this template-based modeling approach, this phase is ideally supported by a visual drag & drop editor, thereby again emphasizing simplicity and enabling the strong participation of stakeholders.

- **Appearance Design:** In this phase, the concrete appearance of each *Dialog Partition* is designed, again supported by the Web-based editor. Therefore, a concrete *Interaction Element* is assigned to each element from the data model. Based on the type of a data element, a possible *Interaction Element* was already assigned at dialog generation time (e.g. input for string, select1 for enumerations etc.) and can be modified. This can be done by either selecting the appearance, i.e. how shall the interaction element be rendered (e.g. select1 either as radio buttons or dropdown list) or switching to a different *Interaction Element* type. Considering the final rendering for diverse clients with possibly smaller screen sizes, a *Dialog Partition* may be split up into several smaller partitions. To this end, partitions as well as groups of elements therein can be marked as non-dividable. In order to provide additional guidance to users, input validations or dynamic features like hints or calculations can be defined. Due to the visual editor, this phase can also be performed in strong collaboration with stakeholders.
- **Evolution:** In the case of extensions or modifications in the *Data Design* phase, new or modified data elements can be designed in detail with respect to partition membership, dynamic behavior and appearance in the succeeding phases. For changes not affecting the data model, the Data Design phase can be skipped.

6.2 The Domain-Specific Model (DSM)

The Dialog DSL's DSM specifies the formal schema for all dialogs that can be designed with the DSL. Thus, it is tailored to the problem domain, not the solution domain, i.e. it abstracts from the final implementation. Although DIM notations serve for simplifying and tailoring a DSL to a specific stakeholder group, choosing well-known concepts and abstractions from the problem domain already in the DSM is advisable. Exploring the domain of dialog-based user interaction in the Web, two necessary groups of concepts to be integrated in an appropriate DSM were identified: Concepts for describing *Interaction Elements* and concepts for specifying dynamic behavior of a dialog, so-called *Interaction Structures*. Figure 6-2 depicts a simplified excerpt from the Dialog DSL's DSM.

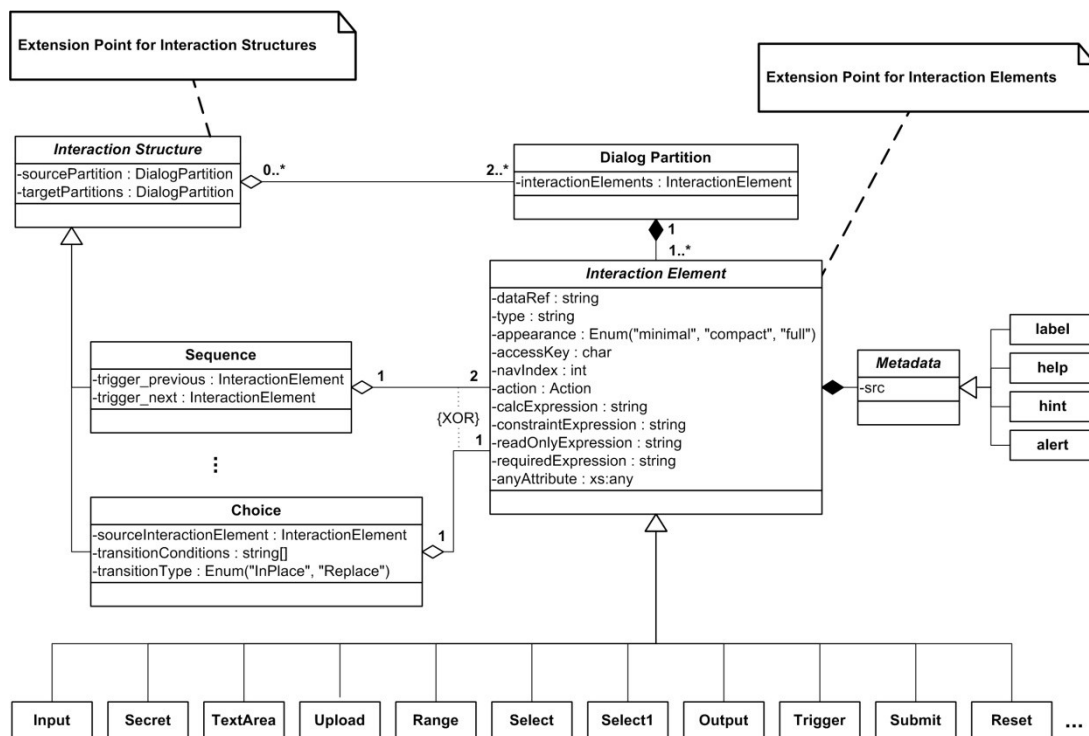


Figure 6-2: Simplified Excerpt from the Dialog DSL's Domain-Specific Model

A *Dialog Partition* presents a semantically cohesive part of a dialog and encapsulates one or more *Interaction Elements*. The design of the concept *Interaction Element* is founded on the specification of interaction elements in the *W3C XForms standard* (Boyer, Dubinko, Leigh L. Klotz et al. 2007). These present a good basis for expressing interaction elements within a DSL as they are based on high-level user interaction primitives instead of presentation- or platform-dependent user controls (Raman 1997). Thus, the DSM achieves a separation of a user control's underlying *intent* from its presentational and implementation aspects. Hence, the DSM comprises *Interaction Elements* for entering values (*Input*), secret information (*Secret*), and larger texts (*TextArea*), for uploading files (*Upload*), for selecting one (*Select1*) or multiple (*Select*) values from a given set or from a sequential range of values (*Range*), for displaying data (*Output*), for triggering actions (*Trigger*) as well as for submitting (*Submit*) and resetting (*Reset*) a form. The abstract *Interaction Element* concept already defines a common set of properties which can be extended by its specialized child concepts. These properties serve for specifying aspects related to the *Interaction Element's appearance*, accessibility (*accessKey*) and navigation order (*navIndex*), event-handling (*action*), data computation (*calcExpression*) and validation (*constraintExpression*, *requiredExpression*) as well as the possible interaction mode (*readOnly*). The DSM can be extended by additional *Interaction Elements* as indicated by the corresponding extension point.

An *Interaction Structure* represents dynamic behavior between a source *Dialog Partition* and one or multiple target *Dialog Partitions*. So far, an extensible core set of *Interaction Structures* representing common dynamic behaviors in dialogs was incorporated. The *Sequence Interaction Structure* represents a wizard-like sequence of dialog partitions, each of them being presented to the user one at a time and

connected via previous / next navigation facilities. Thus, a dialog's complexity can be reduced by semantically grouping *Interaction Elements* into *Dialog Partitions* and interconnecting them via the *Sequence Interaction Structure*. The *Choice Interaction Structure* represents the dynamic display of a *Dialog Partition* in response to a selection made by the user (Nussbaumer 2001). Therefore, a source *Interaction Element* as well as a set of *transition conditions* associated with the set of target *Dialog Partitions* has to be specified. Furthermore, it can be differentiated between an *InPlace*- and a *Replace*-typed transition, i.e. whether the target partition is displayed in addition to the source partition or replaces it. As indicated in the figure by the corresponding extension point, this initial set of *Interaction Structures* can be systematically extended as well.

6.3 The Domain Interaction Model (DIM)

The Dialog DSL's *Domain Interaction Model (DIM)*, i.e. the modeling notation, defines graphical notations corresponding to the concepts defined in the DSM. Hence, it has to cover two major groups of concepts: *Interaction Elements* and *Interaction Structures*.

With regard to *Interaction Elements*, employing well-known dialog user controls turned out to be a good choice. For example, an *Input Interaction Element* is represented by an input field, a *Select1 Interaction Element* by a dropdown list control, and a *Trigger Interaction Element* by a button. This way, a graphical symbol was defined for each *Interaction Element* in the DSM. As a result, almost all symbols in the DIM notation are already known to stakeholders, thus making it rather intuitive and fostering the modeling approach's learnability and simplicity.

Concerning the modeling of dynamic behavior based on *Interaction Structures*, the DIM introduces predefined Petri net constructs. Petri nets provide a sound foundation for modeling dynamic behavior, parallelism and the state of a system. These characteristics can all be found in advanced dialogs as well, thus making Petri nets a good choice. In order to reduce complexity which could arise in complex Petri nets, a transition template for each *Interaction Structure* was predefined, thereby simplifying the modeling process.

With the aim of achieving a sound *separation of concerns*, the modeling notation is divided into two tiers: The first tier addresses the modeling of *Partitions and Transitions* by means of the Petri net transition templates mentioned above. The second tier focuses on the *Appearance Design* of a partition. This marks a significant improvement compared to existing model-based dialog construction approaches and commercial tools. While these strongly focus on a paper-like, single-page-centric form development methodology where dynamic aspects are hidden behind property dialogs, the Dialog DSL's two-tiered modeling approach inherently moves the focus to semantic grouping and dynamic behavior. The *Appearance Design*, being at the center of attention in existing approaches, follows only afterwards in a second step.

This considerably contributes to the resulting dialog's *usability* and particularly helps avoiding *cognitive overload*. Thus, the Dialog DSL's modeling notation naturally encourages the adoption of *usability best-practices* introduced in Section 2.3, i.e. "*Reducing Cognitive Overload by Semantic Partitioning*" and "*Selection-Dependent Inputs*" as well as, to some extent, "*Clear Path to Completion*". The idea of focusing dynamic behavior and lifting it on an own modeling layer on top of a dialog's appearance design could be rather straightforwardly transferred to other dialog construction methodologies and tools as well. By making dynamic behavior explicitly visible to designers and stakeholders, its recognition as well as related communication can be significantly improved.

6.3.1 Partitions & Transitions Modeling Tier

On this tier, semantically cohesive elements from the dialog's data model are grouped into *Dialog Partitions* which are represented by Petri net places. At runtime, if a Petri net place is marked, its encapsulated *Interaction Elements* are visible. Subsequently, the *transitions* between these *Dialog Partitions* are defined using predefined Petri net transition templates according to the Dialog DSL's *Interaction Structures*.

Figure 6-3 illustrates the Petri net representation of a *Choice Interaction Structure*. In this context, elements in a Petri net place are again considered as Petri net places, thus resulting in hierarchical Petri nets. Accordingly, the *Choice* transition template is connected to the data element whose value decides on which transition is fired and to the various target places. The transitions are labeled with the various values the data element in the source place can take. To this end, it is advisable to map such an element to an *Interaction Element* with a discrete value range (e.g. *Select1*), which can be done on the Appearance Modeling Tier.

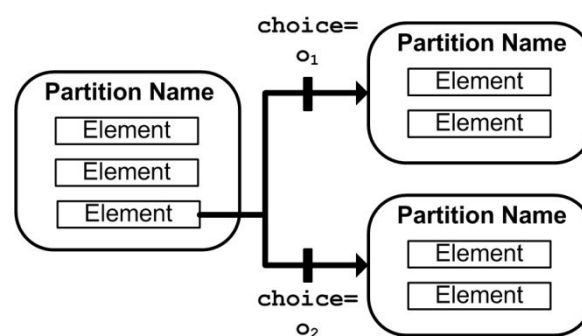


Figure 6-3: The Choice Interaction Structure as Petri Net Transition Template

At runtime, if a place becomes marked, all contained data elements, or their associated *Interaction Elements* respectively, become marked. When the user changes the value of an element connected with a *Choice* transition, the mark of the element flows to the target partition, thus making it and its elements visible. The source partition's mark, however, is still there, meaning that both partitions are

visible. If this is not the desired behavior, i.e. the source partition should become invisible and only the target partition become visible, the transition would have to be connected to the source partition instead of the concrete element. For the sake of simplicity though, a *Choice* transition is always connected to the respective element. If the source partition shall become invisible when a transition fires, the transition can be annotated with a *[Replace]* tag. It should be mentioned that when a partition becomes invisible, its state is preserved by the marking of its encapsulated elements and thus is inherently restored when the partition becomes visible again.

The Petri net representation of a *Sequence Interaction Structure* is depicted in Figure 6-4. Here, the transitions are always connected to the Petri net places as this *Interaction Structure* is independent from the data model. It rather represents a wizard-like navigation through a linear space of *Dialog Partitions*. When the model is rendered into an executable dialog, corresponding *Interaction Elements* (e.g. *Triggers*) allowing the activation of a transition are added to the source and target partition. To this end, the labels annotated at the transitions are taken as labels for the Interaction Elements.

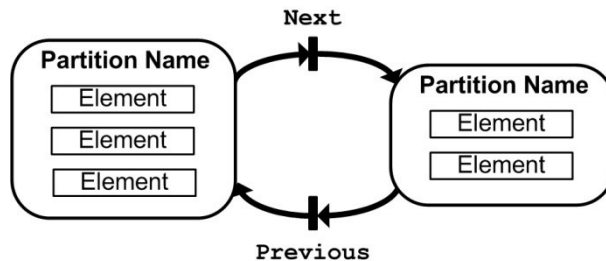


Figure 6-4: The Sequence Interaction Structure as Petri Net Transition Template

6.3.2 Appearance Modeling Tier

Based on the *Dialog Partitions* defined on the superordinate tier, this tier focuses the concrete *Appearance Design* of each of these partitions. Figure 6-5 illustrates a core set of the possible modeling options.

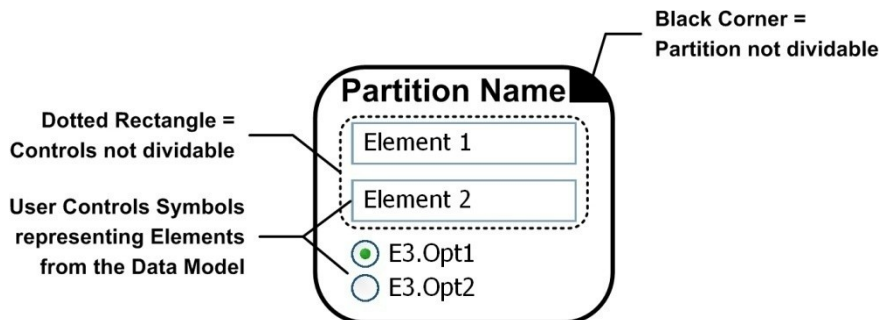


Figure 6-5: Binding Interaction Elements via Corresponding User Control Symbols to Data Elements and Defining Semantic Groups

First of all, an *Interaction Element* represented by a corresponding graphical user control symbol has to be assigned to each data element. Moreover, labels can be defined for each interaction element and additional markup, e.g. for headings, be inserted. With regard to the requirement of *device-independency*, a partition can be semantically tagged as ‘not dividable’, indicated by a black corner. This means that possible runtime model adaptations for clients with small displays should attempt to keep the elements of the partition together. For partitions which are generally considered dividable, a more fine-grained specification of cohesive groups can be achieved based on *Interaction Elements*. Therefore, groups of coherent *Interaction Elements* can be marked by a dotted rectangle. This pen-and-paper-like modeling approach can be augmented by a corresponding editor allowing the detailed configuration of *Interaction Elements* according to the DSM.

6.4 Model Transformations

In the context of the Dialog DSL approach, two kinds of model transformations are required. On the one hand, model transformations are applied for runtime adaptations of the dialog model according to the capabilities of the requesting user agent. On the other hand, the dialog model has to be transformed into executable markup, e.g. XForms code.

6.4.1 User-Agent-related Model Adaptations

The Dialog DSL suggests modeling of dialogs and their decomposition into partitions with regard to a regular desktop terminal. Considering the requirement of device-independency and the variety of device-specific screen characteristics though, dialogs may have to be further decomposed into suitable client-specific partitions, also referred to as *pagination*. This is particularly necessary if either no device-independent markup formats are used as final model serialization formats or client applications are not capable of performing such device-specific adaptations. For example, some XForms-enabled browsers for mobile devices natively perform screen-specific dialog adaptations whereas others only realize a direct rendering of the received markup.

Figure 6-6 exemplifies the model adaptation strategy for decomposing *Partition A* into several smaller partitions, i.e. *Partition A.1-A.3*. The pagination algorithm receives the characteristics of the requesting client as input (cf. Section 6.5) and fills a partition with controls until their combined estimated size on the user agent exceeds the given maximum screen size. In that case, an additional partition is created and filled. As far as possible, *semantic groupings* like the grouping of *Control 3* and *Control 4* are preserved. Similarly, *Dialog Partitions* which have been marked as *non-dividable* on the *Appearance Design* tier remain unchanged as well. The

interconnection of the resulting micro-partitions is realized via the *Sequence Interaction Structure*.

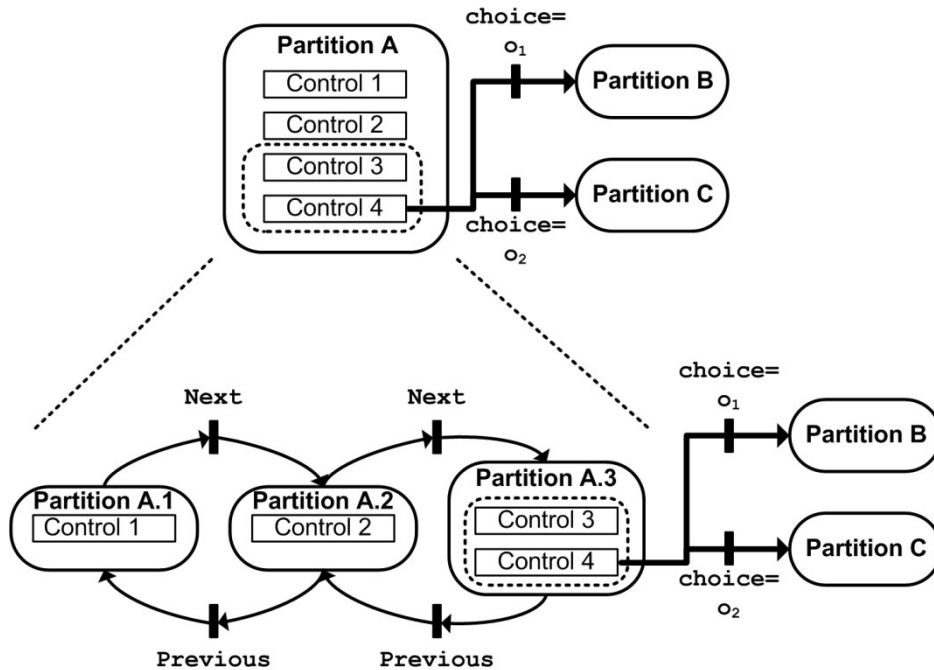


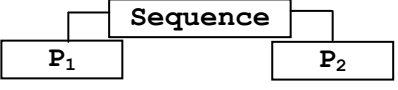
Figure 6-6: Pagination of a Dialog Partition via the Sequence Interaction Structure

6.4.2 Model-to-Code Transformations

Transformations between the DSM and one or more executable markup formats are required for two reasons: On the one hand, after potential client-specific model adaptations have been conducted, the dialog model has to be translated into executable markup to be rendered by the client. On the other hand, backward transformations enabling the import of existing markup code from third parties and its subsequent editing using the Web-based model editor have to be provided. In the context of this thesis, bilateral transformations to and from the W3C *XForms* standard were developed and integrated in the Dialog DSL's technical platform. Thereby, the transformation strategy's underlying idea was adopted from (Nussbaumer 2001).

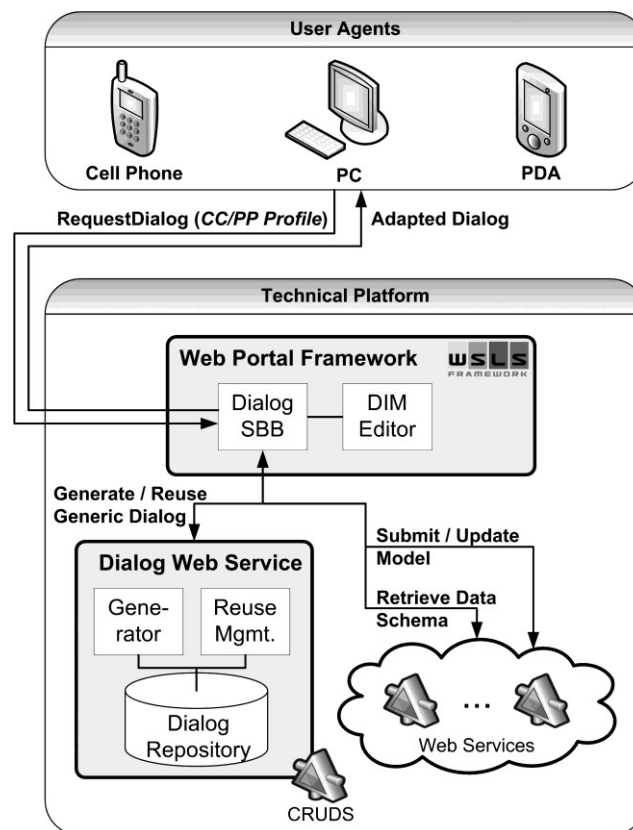
Table 6-1 illustrates the multi-step transformation process. In the first step, a DSM-based model element (1) is mapped to a context-free grammar-based expression (2). Then, this expression is extended by a term-algebraic operation (3) in order to enable its processing by a term rewriting-based compiler. In the last step, term rewriting rules are applied to translate the expressions into the final markup code (4). The shown mapping corresponds to the XForms language. However, by providing different term rewriting rules to other markup languages in this fourth step, additional markup languages, e.g. XAML, could be flexibly and rather straightforwardly incorporated.

Table 6-1: Multi-Step Transformation of Dialog Models into Executable Markup

(1) DSM-based Pattern	
(2) Context-Free Grammar Rule	Sequence := \mathbb{R} P ₂
(3) Extended Rule	Sequence := seq(P ₁ , P ₂)
(4) Term Rewriting Rule	seq(t ₁ , t ₂) → <pre><switch> <case id="t1">eval(t1)</case> <case id="t2">eval(t2)</case> </switch></pre>

6.5 Technical Platform

The Dialog DSL's technical platform comprises the *Solution Building Block (SBB)* as central solution component running on a Web portal framework, e.g. the WSLs Framework, as well as the *Dialog Web Service* and the *Web-based DIM Editor*. Figure 6-7 gives an overview of these components and their interplay for generating and evolving dialogs, serving requesting clients and integrating external Web Services to send or receive data.

**Figure 6-7: Overview of the Dialog DSL's Technical Platform**

6.5.1 The Web-based DIM Editor

In order to support the model-driven construction and evolution of dialogs using the Dialog DSL's *Domain Interaction Model (DIM)*, i.e. the modeling notation, a supplemental Web-based editor was developed. Hence, it can be made immediately available from running dialog instances, thus facilitating rapid roundtrip engineering and making it easily accessible for all audiences. The main design principles driving the development of the editor were *simplicity* and encouraging a *usability-oriented* dialog design.

Figure 6-8 shows a screenshot of the editor's Web-based user interfaces supporting the modeling phase *Partition & Transition Design*. In the *Toolbar* at the top, graphical buttons for adding new *Dialog Partitions* and defining *Sequence* or *Choice Interaction Structures* are available. Once a new partition has been added to the drawing pane, it can be moved via drag and drop, renamed and deleted. Defining a *Sequence* transition is performed via clicking the associated button and subsequently selecting the source and target partitions as well as providing labels for the transition-enabling *Interaction Elements*. Similarly, a *Choice* transition is defined by clicking on the respective button, selecting the source element and one or more target partitions, entering transition conditions and selecting the type of the transition (*InPlace* or *Replace*). In both cases, the editor draws the transition in the model panel and allows its later modification via clicking on it. The *Interaction Structure* definition process is simplified by immediate textual and visual feedback, thus further enhancing the dialog editor's simplicity and ease of use for all kinds of stakeholders.

The figure depicts the *Partitions & Transitions* view of the *Expense Report Dialog* from the 'business trip' example scenario. Therein, several *Sequence Interaction Structures*, e.g. between the partitions *Start* and *Itinerary* as well as between *Itinerary* and *Car Expenses*, were defined. Furthermore, *Choice Interaction Structures* between the element *DepartureFrom* and the partition *Home Address*, between the element *TravelIncludedCar* and the partition *Type of Car* as well as between the element *RentalOrPrivateCar* and the partitions *Rental Car Expenses* and *Private Car Expenses* are visible. While the former are *InPlace*-typed *Choice* transitions, i.e. the target transition is shown in addition to the source partition, the latter is marked as *Replace*-typed *Choice* transition, i.e. the target partition replaces the source partition. The type of *Choice* transition is indicated by an empty (*InPlace*) or filled (*Replace*) transition symbol.

Beyond that, a button serves for encapsulating all unassigned data elements in a new *Dialog Partition* and the last button saves the model and redirects the user to the corresponding running dialog instance. In the left panel, the editor displays a list of elements from the *data model* that have not yet been assigned to a *Dialog Partition*. The assignment of such an element can be performed via dragging it and dropping it onto a *Dialog Partition*. In the same way, an element can be moved to another partition, whereby integrity, e.g. when the element is connected to a *Choice Interaction Structure*, is observed by the editor.

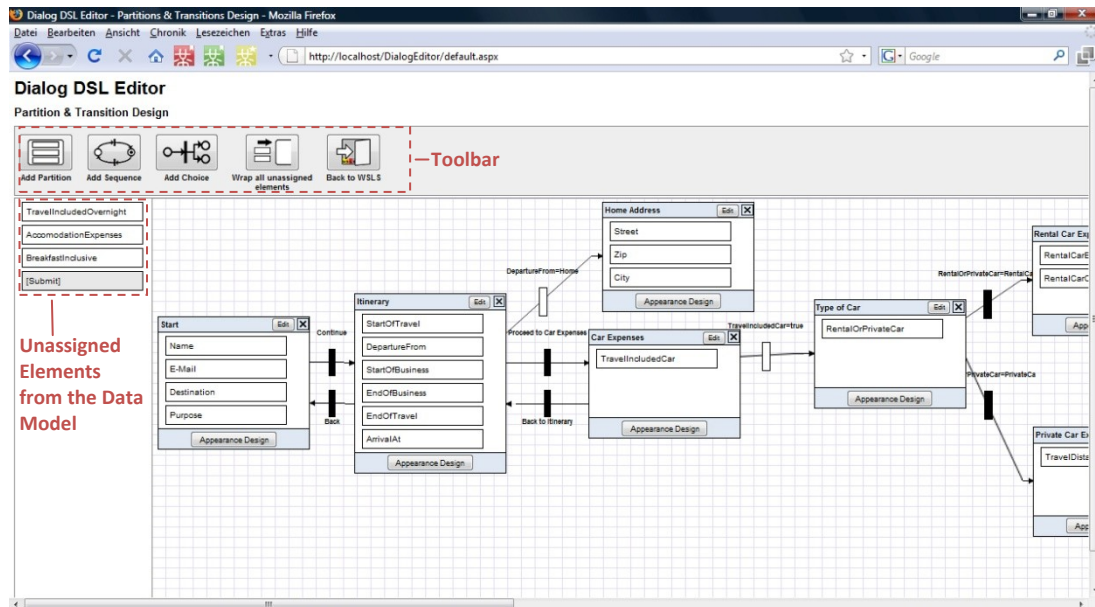


Figure 6-8: Partitions & Transitions Design in the Web-Based DIM Editor

As depicted in the figure, each partition contains a button labeled *Appearance Design*, which leads the user to the *Appearance Design* view of the respective partition. Figure 6-9 shows this view for the *Dialog Partition Itinerary*.

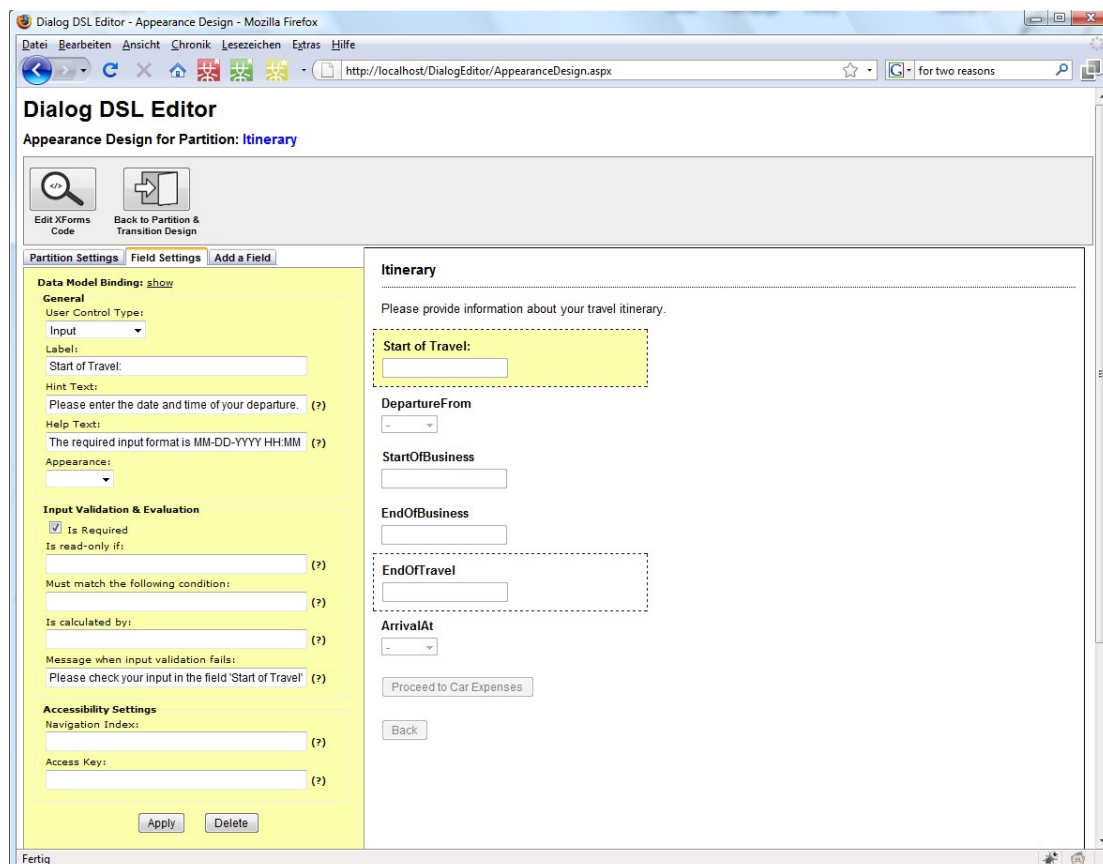


Figure 6-9: Usability-Oriented Appearance Design in the Web-Based DIM Editor

The *Appearance Design* view of the Web-based editor supports the appearance modeling of a *Dialog Partition* and its contained *Interaction Elements*. The user interface comprises three areas: At the top, two buttons for editing the dialog model in its serialized markup representation (*Edit XForms Code*) and for returning to the *Partitions & Transitions* view (*Back to Partitions & Transitions Design*) are available. At the left, three different panes are provided for configuring *Partition Settings*, *Field Settings* and *Adding a Field*. The *Partition Settings* tab allows providing a title and an introductory text for the partition being currently edited. The *Field Settings* tab supports the configuration of the currently selected *Interaction Element* and contains type-specific configuration properties. Thereby, the editor explicitly fosters the incorporation of *usability best practices* by focusing on corresponding configuration facilities.

In the figure, the *Field Settings* tab for the currently selected *Interaction Element* ‘*Start of Travel*’ is shown. The contained configuration facilities allow for changing the *Interaction Element’s type*, its *label*, providing *hint* and *help* texts and specifying its abstract *appearance* size. Furthermore, input validations can be defined, e.g. marking a field as *required*, providing an XSL-based expression which determines if a field is *read-only*, or specifying *value constraints* and specifying meaningful *error messages*. Furthermore, the value of a field can be automatically *calculated* at runtime based on a XSL-based expression. Finally, the *Interaction Element’s navigation index* as well as an *access key* can be defined. For other *Interaction Element* types, additional type-specific properties are available. The question mark symbols next to each configuration property lead to help texts and examples, which further improve the editor’s simplicity and usability for all kinds of stakeholders.

While existing dialog editors predominantly focus on technical details, this set of configuration facilities for *Interaction Elements* explicitly draws the designer’s attention to *usability best practices* as introduced in Section 2.3. Thus, the adoption of best practices like “*In-Context Help and Hints*”, “*Immediate Feedback and Meaningful Error Indication*” as well as “*Clear Path to Completion*” is considerably eased and their recognition improved.

In the preview pane on the right, the relative *layout* of the dialog’s *Interaction Elements* can be modified via drag and drop. Thus, the editor natively ensures a consistent and uniform dialog layout, thereby inherently adopting the *usability best practices* “*Consistent Form Layout*” and “*Visual Continuity*”.

The Web-based dialog editor was implemented based on the Microsoft .NET Framework using ASP.NET 2.0 and the ASP.NET AJAX Extensions 1.0 (Microsoft Corp. 2009). Thus, it offers a rich, highly interactive behavior known from desktop applications which significantly eases its usage. Page reloads are avoided by asynchronous background communication with the server.

Due to the presented bilateral model transformations from and to the W3C *XForms* standard, the editor’s applicability is not restricted to the Dialog DSL context. It rather enables a global audience to create and modify arbitrary *XForms* documents based on the Dialog DSL’s modeling notation. Therefore, it is publicly available on the research homepage of the IT Management and Web Engineering Research Group (MWRG 2009b).

6.5.2 The Solution Building Block (SBB)

The Solution Building Block forms the central technical solution component of the Dialog DSL and runs on a Web portal framework. In the context of this thesis, an implementation for the WSLs Framework was performed. The SBB is in charge of various functions during the development and execution process. In the beginning, a SBB instance can be either configured with a data schema and a submission target or with an URL to a WSDL file and the name of the operation the dialog shall be submitted to (cf. Figure 6-10).

Figure 6-10: Initial Configuration of a Dialog SBB Instance

Thereupon, the SBB instance passes the respective data schema to the *Dialog Web Service* which *automatically generates* a corresponding basic dialog model and returns it to the SBB instance. At generation time, an appropriate *Interaction Element* is assigned to each element of the data model depending on its data type. If requested, a decomposition of the dialog into *Dialog Partitions* derived from the data model's structure is performed. A detailed description of the generation process and methodology can be found in (Allerding 2007). Alternatively, an existing dialog model from the *Dialog Repository* can be searched and reused based on advanced semantic context-dependent search mechanisms (cf. the *Web Engineering Reuse Sphere* presented in Chapter 7). From that moment on, a fully operational Web-based dialog is already available, without having performed any manual modeling. The dialog can now either be modeled in detail using the *Web-based DIM*

editor presented in the previous section or directly used in production. In either case, no (re)compilation or (re)deployment is required. Changes made in the dialog editor are directly visible in the running dialog, thus enabling an agile and evolutionary development approach.

A figure of the automatically generated *Expense Report Dialog* was already included in the previous chapter (cf. Figure 5-50). Figure 6-11 shows the *Car Expenses* section of the *Travel Expense Report Dialog* which embodies the dynamic behavior specified on the *Partitions & Transitions* modeling tier as depicted in Figure 6-8. The depicted dialog section consists of three *Dialog Partitions*: *Car Expenses*, *Car Type* (2) and *Rental Car Expenses* (4). By selecting the value 'Yes' at the *Interaction Element* marked with (1) the *Choice Interaction Structure* to the *Dialog Partition Car Type* is activated and results in showing the respective partition within the *Car Expenses* partition. Similarly, selecting the value 'Rental Car' at the *Interaction Element* marked with (3) results in activating an associated *Choice Interaction Structure* and in dynamically making the partition *Rental Car Expenses* visible within the partition *Car Type*. The buttons marked with (5) serve for triggering *Sequence Interaction Structures* between the *Dialog Partitions Itinerary* and *Car Expenses* as well as *Car Expenses* and *Accommodation Expenses*.

Figure 6-11: Rendered Web-based Travel Expense Report Dialog Incorporating Choice (1, 3) and Sequence (5) Interaction Structures

Beyond that, if a client requests a Web page containing a Dialog SBB instance, the SBB identifies the client's screen characteristics based on the user agent string contained in the HTTP request or, in case of a mobile device, by evaluating the *User Agent Profile (UAProf)* (Wireless Application Forum 2001) which is supplied based on the *W3C Composite Capability Preferences Profile (CC/PP)* standard (Kiss 2007). Thereupon, by initiating the presented *model transformations*, the SBB adapts the

dialog model accordingly, translates it into executable markup, e.g. XForms, and returns it to the client. In order to achieve an adequate performance, the final markup is cached until the dialog model gets changed and can thus be reused for identical requests. With regard to rendering XForms-based markup in browsers providing an insufficient coverage of the standard, the FormFaces Framework (Progeny Systems 2007) was adopted and slightly extended. FormFaces is a completely JavaScript-based XForms rendering engine supporting a broad range of today's browsers and is integrated via simply referencing the FormFaces JavaScript library from a Web page. Thus, the actual rendering is performed at client side whereby the XForms markup is dynamically transcoded into XHTML.

Finally, the Dialog DSL's SBB acts as *mediator* between a dialog and submission endpoints. Thus, submissions of the dialog's data model instance in whole or part are received by the SBB and processed, e.g. in the context of a workflow, or forwarded to a Web service the dialog asynchronously communicates with. In the latter case, the SBB receives the response from the Web service and forwards it to the corresponding client.

6.6 Summary

This chapter presented the Dialog DSL, a novel engineering approach for the fully model-based construction and evolution of advanced Web-based dialogs. The approach is based on the previously introduced *Web Engineering DSL Framework* and places strong emphasis on *simplicity*, thus enabling stakeholders to intensely participate in the development process by validating, modifying and creating dialogs or their models respectively. In the following, the Dialog DSL's unique solution elements as well as the fulfillment of the requirements elaborated in Chapter 2 are briefly summarized.

The Dialog DSL's two-tiered *Domain Interaction Model (DIM)*, i.e. the modeling notation, strongly focuses on *usability* and *dynamic behavior* in particular. On the Petri net-based *Partitions & Transitions* modeling tier, semantic groupings in terms of *Dialog Partitions* and dynamic behavior based on *Interaction Structures* are modeled. On the *Appearance Design* tier, the appearance of each *Dialog Partition* and its contained *Interaction Elements* is specified. This approach marks a significant improvement to the current state of the art as the Dialog DSL inherently moves the focus away from technical implementation details and towards usability aspects and best practices. The *Appearance Design*, being at the center of attention in existing approaches, is considered only after that in a second step. The Dialog DSL's unique modeling approach could be transferred to other dialog construction methodologies and tools as well. The supplemental *Web-based DIM editor* further encourages and facilitates a usability-oriented design, also for non-programmers.

The requirement for supporting *device-independent* access and usage of Web-based dialogs is considered both in the DIM and the technical framework. In this regard, the main focus lies on the pagination of a dialog according to the screen

characteristics of a requesting client. To this end, the DIM allows for marking *Dialog Partitions* as non-dividable as well as defining semantically cohesive groups of *Interaction Elements* within a partition. At runtime, the Dialog DSL's *Solution Building Block (SBB)* identifies the characteristics of requesting clients and applies *model transformations* for adapting the dialog model accordingly.

The Dialog DSL's technical platform provides strong *Web service support*. The *Dialog Web Service* is capable of *automatically generating* a basic but fully operational dialog for a Web service-based submission endpoint. At runtime, the SBB mediates the asynchronous communication between external Web services and dialog instances. Beyond that, a systematically extensible model transformation strategy realizes the transformation of dialog models into *standardized dialog markup languages*, e.g. the W3C *XForms* standard.

Due to the automated dialog generation facilities as well as the rapid, fully model-based roundtrip engineering, the Dialog DSL enables an *agile* and *evolutionary* development methodology. Having generated a basic dialog model, the *Web-based DIM editor* supports its easy yet detailed refinement and modification, also by *stakeholders* without software development skills. Performed modifications are immediately applied and visible in the running dialog, which further eases stakeholder involvement and allows for short evolution cycles.

Due to the simple modeling notation and the intuitive Web-based DIM editor, the Dialog DSL is excellently applicable by both developers and *stakeholders*. Furthermore, the purely model-based construction approach results in significant *efficiency* gains. These factors were evaluated and successfully confirmed in a *formal empirical evaluations* presented in Section 8.3. In combination with the Dialog DSL's unique focus on *usability* and *dynamic behavior*, these factors present the key improvements compared to the current state of the art. Thus, the Dialog DSL gained widespread attention during its presentation at international conferences, e.g. the 17th World Wide Web Conference (WWW'08).

7 The Web Engineering Reuse Sphere⁵

Reuse has been identified very early as an important software engineering principle being able to significantly improve development efficiency and quality (McIlroy 1968). In fact, reuse can lead to greater schedule and effort savings than any other rapid-development practice – if implemented as a systematic and dedicated long-term strategy and supported by an effective framework (McConnell 1996). This holds equally true for the Web Engineering discipline in general and particularly for the *Web Engineering DSL Framework*. Against this background, this chapter introduces the *Web Engineering Reuse Sphere*, a novel reuse framework for the Web Engineering domain. It explicitly addresses the requirements identified in Chapter 2 and thus establishes a sound foundation for effective, *cross-methodological* reuse and strong *stakeholder involvement*. Based on the insight that the *understanding* of an artifact is strongly correlated to its utility and thus directly influences a reuse approach's *effectiveness*, the Web Engineering Reuse Sphere treats the inherent consideration of *stakeholder skills* as a key factor.

To this end, the Web Engineering Reuse Sphere introduces an *ontology* which conceptualizes the Web Engineering reuse domain based on Semantic Web standards and technologies (cf. Section 7.2). Based on this semantic, homogenizing foundation, the approach provides advanced *knowledge-based, cross-methodological search* facilities (cf. Section 7.3) as well as efficient *implicit* and *explicit registration* mechanisms (cf. Section 7.4). The technical integration of existing heterogeneous artifact stores is guided by a *reference architecture framework* (cf. Section 7.5). In conclusion, the *Web Engineering Reuse Sphere* provides a novel degree of strongly stakeholder-oriented support for the following cross-methodological search scenarios (cf. Section 7.6): On the one hand, finding adequate *Resolution Strategies*, i.e. methodologies, and related artifacts for a particular *Task Type* and a particular *Stakeholder* audience or *Skill Set* respectively. On the other hand, finding existing *Artifacts* based on various contextual parameters, again including the respective *Stakeholder* audience or *Skill set* respectively.

⁵ Parts of this chapter have been published in (Freudenstein, Boettger and Nussbaumer 2008)

7.1 The Sphere Concept

The Web Engineering Reuse Sphere is based on the idea of several *spheres* of distributed, *ad-hoc*- and *infrastructure*-based *repositories* and a semantic *registry* in their core as depicted in Figure 7-1. The spheres are divided into various areas representing the different types of artifacts occurring in the Web Engineering domain, e.g. *documents*, *models*, *components* etc. Each area contains type-specific repositories for its reusable artifacts.

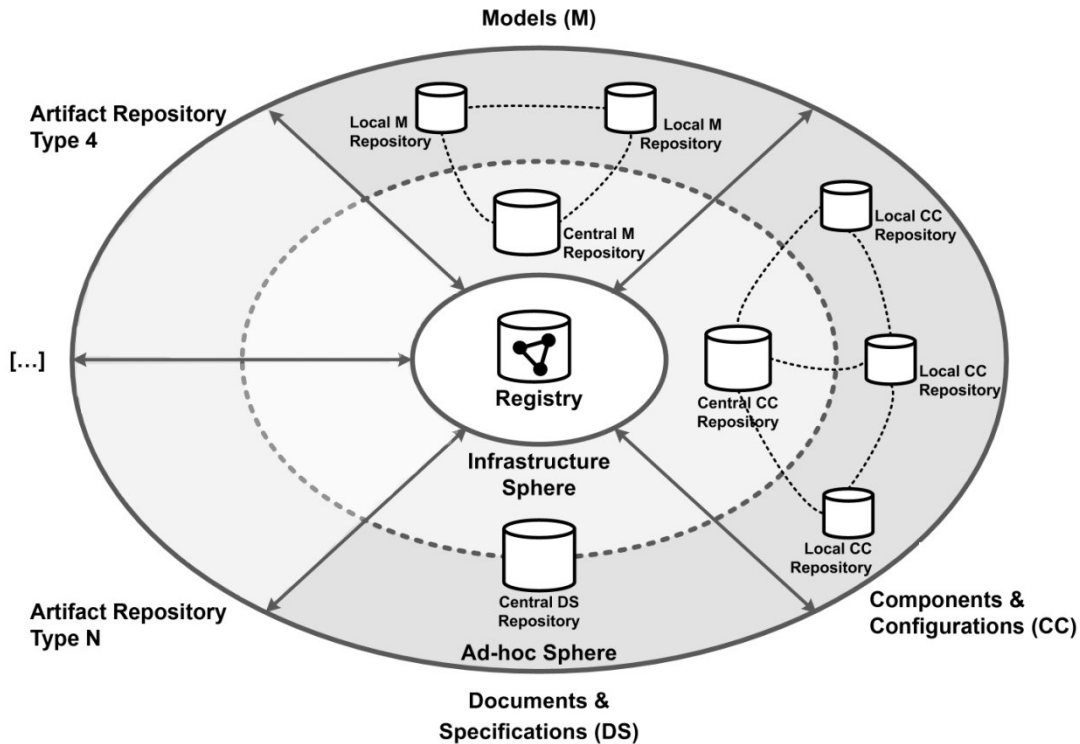


Figure 7-1: The Web Engineering Reuse Sphere

The *Web Engineering Reuse Sphere* approach defines two *sphere levels*. The *infrastructure* level contains one dedicated reuse repository per area serving for planned reuse. Therein, sufficiently mature and stable artifacts are explicitly published for being reused. As indicated by the term 'infrastructure', such repositories are specifically set up for systematic long-term storage of artifacts including versioning.

The *ad-hoc* level is optional for an area and contains repositories for *spontaneous* reuse. Such ad-hoc repositories are usually already in use and are rather application-specific data stores than actual reuse repositories. In the *models* area, for example, a local database containing current models could be such an ad-hoc repository. Another example would be the data store of a Web application development environment running on a developer's computer and representing the current state of development. Consequently, artifacts are available in the ad-hoc level from the moment on when they are saved for the first time until they are deleted.

A *central ontology-based registry* forms the core of the sphere. It registers all artifacts in all repositories – both on the *infrastructure* and the *ad-hoc level* – along with their semantic metadata and provides holistic registration and search functionalities. When searching for artifacts, results can encompass both artifacts that were explicitly published in a repository on the *infrastructure level* and artifacts from a repository on the *ad-hoc-level* being still under development. Reuse can thus be performed in a *peer-to-peer* style on the *ad-hoc-level* and in a *planned* way on the *infrastructure level*, whereby both mechanisms contribute to the approach's efficiency and effectiveness. The former allows for discovering and exchanging work in progress between local application-specific stores. This in turn results in a *coordinated* and *efficient collaboration* by reducing redundant developments and avoiding consolidation efforts.

An interesting symptom that can be observed on the *ad-hoc-level* is the correlation of an artifact's popularity and its persistency. Artifacts being very popular, e.g. due to their quality, applicability, generality etc., will be more persistent than others. This is due to the fact that repository contents on the *ad-hoc-level* are usually only available while at least one person uses them for their current project. When a person removes an artifact from their local repository and the artifact is not contained in any other repository on the *ad-hoc level*, i.e. nobody else (re-)uses this artifact, it is no longer available. Analyzing factors like an artifact's degree of persistency or its (re-)usage in various settings can thus help to derive statements about its characteristics like e.g. its quality, applicability, usefulness etc.

After an artifact was completed and has gained sufficient maturity, e.g. by passing quality inspections, it can be transferred to a repository on the *infrastructure level*, thus being persistently and reliably available for planned reuse.

7.2 The Semantic Core: The Web Engineering Reuse Ontology

The semantic ontology-based *registry* forming the core of the *Web Engineering Reuse Sphere* is in charge of registering all artifacts throughout the repository space based on semantic metadata. Therefore, a generic *Web Engineering Reuse Ontology* which provides the basis for classifying artifacts as well as powerful inference-based search mechanisms was developed. The ontology was elaborated according to established ontology engineering methodologies (Uschold and King 1995; Prieto-Diaz 2003) and formalized based on the *Web Ontology Language (OWL)* in its *OWL-DL* variant (Bechhofer, Harmelen, Hendler et al. 2004). Strong emphasis was placed on *generality*, i.e. keeping the ontology open for any Web Engineering method and incorporating well-defined extension points. Furthermore, existing ontologies were integrated where possible. For example, the *FOAF* ontology (Brickley and Miller 2007) being related to the concept *Stakeholder*, the *Dublin Core* ontology (Dublin Core Metadata Initiative 2008) defining standardized metadata properties for core concepts like *Artifact* or *Project* or the *OntoWeb* ontology (Fensel 2003) covering the concepts *Product* and *Business Domain* were incorporated. The resulting *Web*

Engineering Reuse Ontology is publicly available on the research homepage of the IT Management and Web Engineering Research Group (MWRG 2009b).

The following section gives an overview of the ontology and briefly exemplifies its application for evaluating skill- and knowledge-based search queries based on inference. Afterwards, a detailed presentation of selected parts of the ontology follows in the subsequent sections. Thereby, the systematic integration of knowledge about existing Web Engineering methodologies by instantiating abstract concepts of the ontology is exemplified.

7.2.1 Overview of the Web Engineering Reuse Ontology

Figure 7-2 depicts a simplified overview of the ontology's core concepts and relations. The ontology defines concepts for *Artifacts* and their context in terms of the associated *Web Application*, *Project*, *Process model*, *Product*, the employed *Modeling Technique* etc. Furthermore, the ontology describes the interrelation of particular *Task Types* occurring in the development of a *Web Application*, corresponding *Resolution Strategies*, associated *Modeling Techniques* and *Software* as defined by *Web Engineering Methodologies* as well as the *Skills* and *Knowledge* required therefore. In addition, the ontology allows for describing representative *Stakeholder* groups and their *Skills*.

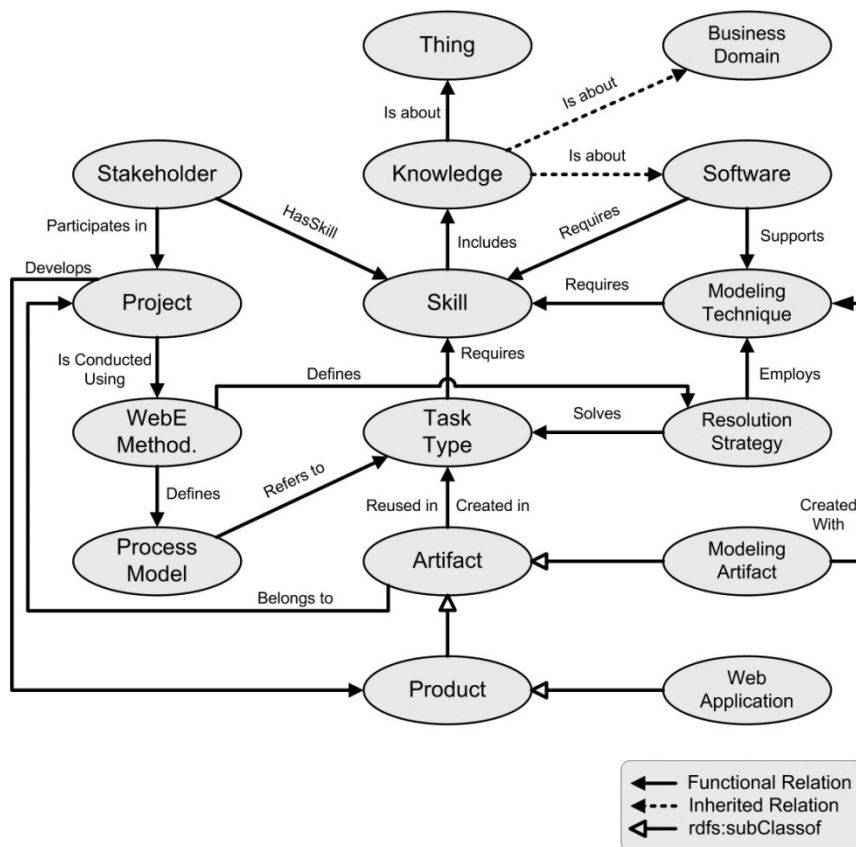


Figure 7-2: Simplified Overview of the Ontology

Based on the ontology, powerful knowledge-based search queries can be processed. Besides simple queries like finding existing *Artifacts* being related to a particular *Business Domain, Concern* (i.e. content, navigation, presentation, interaction, process, communication, amongst others), *Web Engineering Methodology* or *Task Type*, more advanced queries, especially supporting effective stakeholder involvement can be resolved. This shall be illustrated by the following example.

Figure 7-3 depicts a simplified excerpt from the ontology with concepts and relations from the core ontology (white ellipses), Web Engineering methodology-independent instances (grey ellipses in the middle) as well as exemplary instances for the Web Engineering methodologies *UWE* and *WebML* (left and right).

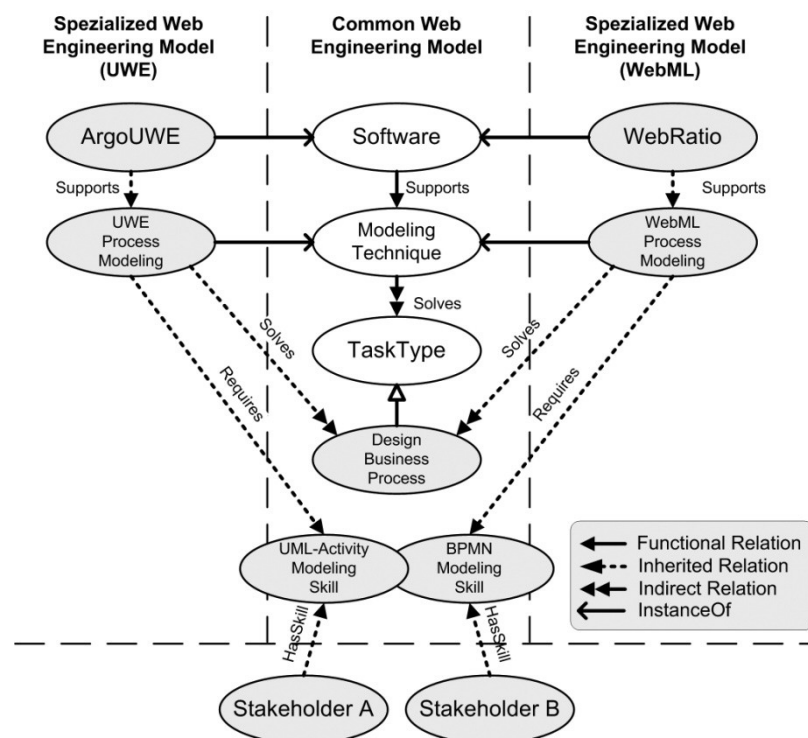


Figure 7-3: Ontology Excerpt with Instances for WebML and UWE

Thus, for the given *Task Type* instance *Design Business Process* and the *Skills* of *Stakeholder B* (i.e. *BPMN Modeling Skills*), appropriate *Modeling Techniques* can be determined by inference. In this example, the query result would be the *Modeling Technique WebML Process Modeling* which is based on BPMN and supported by the *Software WebRatio*. For *Stakeholder A* having *UML Activity Modeling Skills*, the result would be the *Modeling Technique UWE Process Modeling* which is based on UML and supported by the *Software ArgoUWE*.

In addition to determining adequate *Modeling Techniques*, search results could directly include existing *Artifacts* – in this case *Modeling Artifacts* - created with the same *Modeling Technique* in similar *Project* or *Web Application Type* contexts. Furthermore, also *Artifacts* marked as templates for the determined *Modeling Technique* could be supplied.

Such *cross-methodological* scenarios are gaining increasing importance in the context of current consolidation activities in the Web Engineering research community like *MDWEnet* (Vallecillo, Koch, Cachero et al. 2007). To this end, the *Web Engineering Reuse Sphere* approach and its associated architectural framework can serve as a valuable accelerator unfolding the potential of cross-methodological interchange and collaboration.

7.2.2 The Concepts *Knowledge* and *Stakeholders*

Figure 7-4 illustrates a simplified excerpt of the ontology's concepts and relations covering the domains *Knowledge* and *Stakeholder*. These concepts form a central part of the ontology as they are used to specify the semantic foundation used for evaluating inference-based queries concerning the adequacy of *Artifacts*, *Resolution Strategies*, *Modeling techniques* and *Software* for given stakeholders. The white ellipses represent connecting concepts which are out of the current figure's scope.

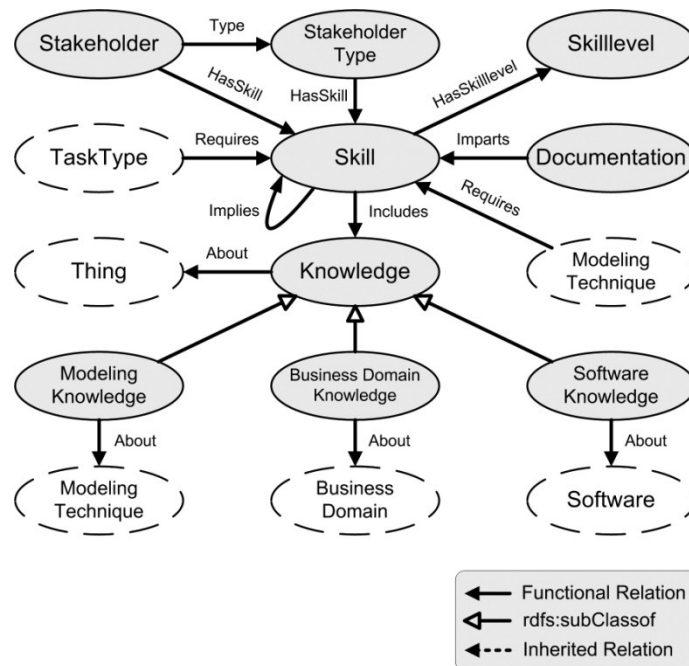


Figure 7-4: *Ontology Concepts related to Knowledge and Stakeholders (Simplified Excerpt)*

Therefore, the ontology includes the central concept *Knowledge* which is differentiated into several types of knowledge like *Business Domain Knowledge*, *Modeling Knowledge* or *Software Knowledge*. The *About* relations between these knowledge types and the subjects of knowledge realize the connection to other concepts in the ontology, i.e. *Modeling Techniques*, *Business Domain* and *Software*. The concept *Skill* realizes the connection between *Knowledge* and *Stakeholders* or *Stakeholder Types* in the sense of *having* knowledge as well as with *Task Types* and *Modeling Techniques* in the sense of *requiring* knowledge. In each case, the relation

is attributed with a *Skill Level* for classifying the degree of the required or possessed *Knowledge*. Furthermore, concepts and relations for expressing that *Documentation* can *impart* missing *Skills* and that particular *Skills* *imply* other *Skills* are available.

7.2.3 The Concepts *Artifact*, *Methodology*, *Process* and *Product*

A simplified excerpt of the concepts and relations around *Artifact*, *Web Engineering Methodology*, *Process Model* and *Product* is depicted in Figure 7-5. This part of the ontology primarily provides the foundation for *integrating Web Engineering Methodologies* along with their development *Process Models*, *Resolution Strategies* and *Artifact Types*.

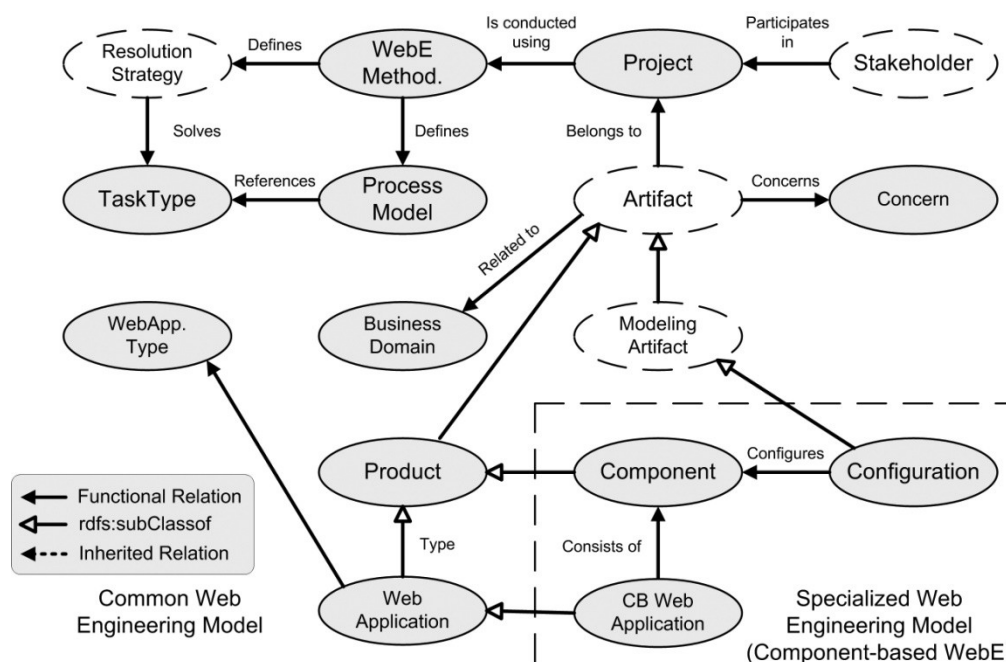


Figure 7-5: Ontology Concepts Related to Artifact, Methodology, Process and Product (Simplified Excerpt)

By including instances of the concept *Web Engineering Methodology*, well-known methodologies like WebML, UWE, OOHDM or OO-H can be included in the ontology. Each methodology defines or refers to its software development *Process Model* which in turn refers to (ideally cross-methodologically shared) *Task Types*. Furthermore, each methodology defines one or more *Resolution Strategies* for every *Task Type*; this relation is considered in more detail in the next subsection. Naturally, the majority of tasks occurring in the development of a Web application, e.g. ‘design workflows’ or ‘design navigation’ can be found across all Web Engineering methodologies, even though their names differ amongst them (Selmi, Kraiem and Ghezala 2005). Thus, in order to support cross-methodological queries, referring to corresponding *existing Task Types* should always be preferred to defining new (redundant) *Task Types*.

Beyond that, refined concepts for diverse *Artifact* types, e.g. *Modeling Artifact* or *Product* are available. As indicated by the separate area in the figure, specific *Artifact* types and *Products* for each Web Engineering methodology can be integrated here. In the figure, extensions for a component-based Web Engineering methodology are exemplarily shown, where *Components* are subclasses of *Product* which in turn is an *Artifact*. Furthermore, they are configured with *Configurations*, which in turn are a special type of *Modeling Artifacts*. Likewise, for integrating the WebML methodology, subclasses or instances of the concept *Modeling Artifact* for its various model types, e.g. 'WebML Hypertext Model', 'WebML Business Process Model' etc. could be defined.

Artifact presents the central concept in the ontology representing all kinds of reusable artifacts. It incorporates general metadata properties from the Dublin Core ontology and can be further classified with respect to related *Project(s)*, Web-specific *Concern(s)*, i.e. content, navigation, presentation, interaction, process, communication, amongst others), or business domain, e.g. Travel Management, Procurement etc.

The concept *Project* is used to indicate in which project(s) an *Artifact* was created or reused. Additionally, it can be expressed which *Web Engineering Methodologies* were used in a *Project*.

7.2.4 The Concepts *Resolution Strategy*, *Modeling Technique* & *Software*

The integration of methodology-specific knowledge in the ontology is a crucial factor for cross-methodological reuse scenarios, e.g. determining *Resolution Strategies*, *Modeling Techniques* and *Software* along with corresponding *Artifacts* in accordance with a given *Stakeholder's Skills* across various *Web Engineering Methodologies*. Thereby, the strengths of each methodology can be used and, in combination with initiatives like the *MDWEnet* activity, the hitherto existing methodological frontiers be overcome.

Figure 7-6 illustrates a simplified excerpt of the ontology covering the concepts *Resolution Strategy*, their *Modeling Technique(s)* and supporting *Software* as well as the resulting *Modeling Artifact(s)*. As before, relations to connecting concepts are represented by white ellipses on the left.

On the right side, dedicated instances describing the *Web Engineering Methodology* WebML and thus integrating it in the ontology are depicted. The shown example shows instances related to the *Task Type* 'Design Navigation'. Therefore, the WebML methodology proposes the *Resolution Strategy Hypertext Design* that employs the *Modeling Technique WebML Hypertext Modeling* which is supported by the *Software WebRatio* and results in the *Modeling Artifact type Hypertext Model*.

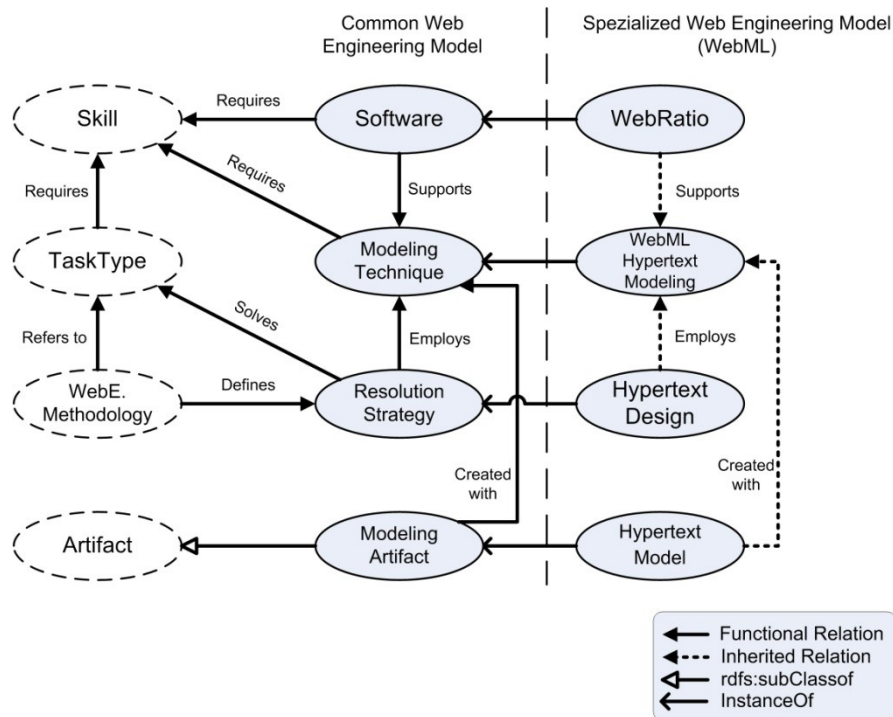


Figure 7-6: Ontology Concepts Resolution Strategy, Modeling Technique & Software and their Instantiation for the WebML Methodology (Simplified Excerpt)

Another example describing the *Web Engineering DSL Framework* is shown in Figure 7-7. In this case, the extension is performed by adding new concepts for *DSL*, *Graphical Notation (DIM)*, *DIM Editor*, *Domain-Specific Model*, *DSL Program* and *Configuration* as subclasses of the core ontology's concepts. Based on these concepts, instances for particular *DSLs* can be defined. Regarding the *Workflow DSL* for example, the *Graphical DIM Notations* BPMN, UML, Petri Nets, SSO and associated *editors*, e.g. Microsoft Visio, IBM Rational Software Architect, INCOME2010 and Microsoft Word, would be integrated. The *Workflow DSL* would be associated with the *Task Type Design Business Process* and the *Web Engineering Methodology DSL-based Web Engineering*.

Based on integrating knowledge in form of such methodology-specific ontology extensions, suitable *Artifacts* and *Resolution Strategies* for a given *Task Type* and given *Stakeholder Skills* can be cross-methodologically determined. Moreover, assumed that cross-methodological model interchange is possible as aspired by the *MDWEnet* initiative, artifacts could be cross-methodologically reused.

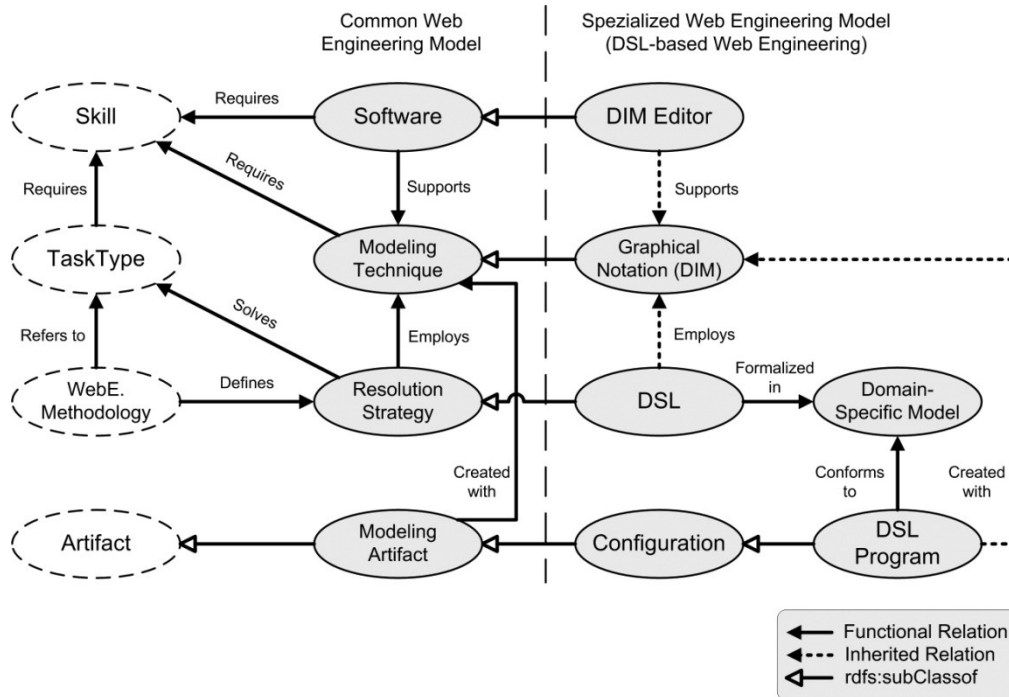


Figure 7-7: *Ontology Concepts Resolution Strategy, Modeling Technique & Software and their Instantiation for the Web Engineering DSL Framework (Simplified Excerpt)*

7.3 Effective Search and Integration

In order to ease the process of finding artifacts, search mechanisms should be both easy to use and effective in terms of finding adequate results very quickly. Common search engines, e.g. Google, usually offer a simple mode, i.e. one input parameter for all kinds of search terms, and an advanced mode, i.e. lots of query parameters. When inexperienced people use such search facilities, it can be observed that for them the simple mode is easy to use, but leads to unsatisfying search results (Nielsen 2008). A lot of knowledge about adequate search terms and query syntax is required to achieve good results. The advanced mode offers more guidance regarding search constraints, but still requires significant knowledge about adequate search terms.

Facing these problems and considering the goal of effective stakeholder involvement, the *Web Engineering Reuse Sphere* pursues an *extensible, user- and scenario-based* approach for providing search facilities. First, in strong collaboration with stakeholders – both development team members and domain experts – reuse scenarios are identified and relevant search parameters elicited. Then, based on the ontology, a corresponding *query template* based on the SPARQL Protocol and RDF Query Language (SPARQL) (Prud'hommeaux and Seaborne 2008) is developed. In doing so, possibly missing relations or sometimes even concepts could be determined. In such a case, the ontology is extended following a systematic ontology evolution process (Haase 2007). Finally, a suitable *search dialog* for the reuse scenario is developed using the *Dialog DSL* (cf. Chapter 6) - again in strong

collaboration with stakeholders. In this context, usability aspects in terms of providing guidance to the user and including dynamic behavior, e.g. in form of multi-step search dialogs, are key factors. At runtime, the user input from the search dialog is inserted in the corresponding SPARQL query template which is then executed on the registry's triplet store and results in a set of relevant artifacts.

In addition, the *Web Engineering Reuse Sphere* supports using the search results as a starting point for *browsing* through the registry space and performing *context switches* following the relations defined in the ontology. For example, for a given *Artifact*, all artifacts from the same *Project*, *Web Application Type*, *Business Domain* etc. or created with the same *Modeling Technique* or *Resolution Strategy* could be retrieved. Beyond that, also more powerful *inference-based context switches* are possible. For example, all *Artifacts* that required similar *Stakeholder Skills* for their creation and that were created in the same *Task Type* and for the same *Business Domain* can be identified. Examples for such scenario-based search dialogs and the described browsing facilities can be found in Section 7.6.

Having found a potentially suitable artifact, it should be easily and safely integrable in the current development context and artifact-specific tools. Therefore, it is desirable to perform searches and retrieve suitable artifacts directly from within artifact-specific tools and editors. To this end, the *Web Engineering Reuse Sphere* includes a *reference architecture framework* (cf. Section 7.5) based on concepts from the field of Enterprise Application Integration (EAI). By establishing a generic Web service layer on top of the repositories and the registry and – if required – tool-specific Web service adapters on top of the registry, proprietary tools can retrieve search results including URLs from the registry and artifacts from the repositories.

For example, Microsoft applications like Word, Excel, PowerPoint or Visio can natively interact with external Web services adhering to the *Research Interface* (Fransen 2003). Thus, e.g. reusable artifacts in form of documents or models could be directly searched and retrieved from within Word or Visio. By providing additional Web service adapters, other tools and applications can be easily integrated. When performing searches from within a tool, some search parameters could be automatically derived from the current context, e.g. the artifact type or the software with which the artifact should be editable. However, such existing facilities for external data source integration usually allow for single-parameter searches only. In order to offer comprehensive search dialogs exploiting the full potential of advanced knowledge-based searches, plugin-based extensions in form of specific search dialogs can be integrated in most of today's applications. Alternatively, the proposed architectural framework contains a generic Web-based search portal for finding and retrieving artifacts.

7.4 Storing Artifacts with Rich Metadata

While registering an artifact in a repository on the *infrastructure level* should require as little manually entered metadata as possible, registering artifacts on the *ad-hoc-*

level should be performed automatically in the background. Therefore, the metadata provided within the associated application should be used and no additional manual input should be required. Thus, in order to minimize the amount of manually provided metadata, approaches for extracting and mapping proprietary metadata statements to the concepts and properties defined in the ontology are required. Figure 7-8 summarizes the relations between the concept *Artifact* and other concepts and thus illustrates the required *metadata* for an effective registration. Beyond that, the *Artifact* concept comprises metadata according to the Dublin Core ontology in form of attributes, e.g. *Identifier*, *Title*, *Publisher*, *LastModified* etc.

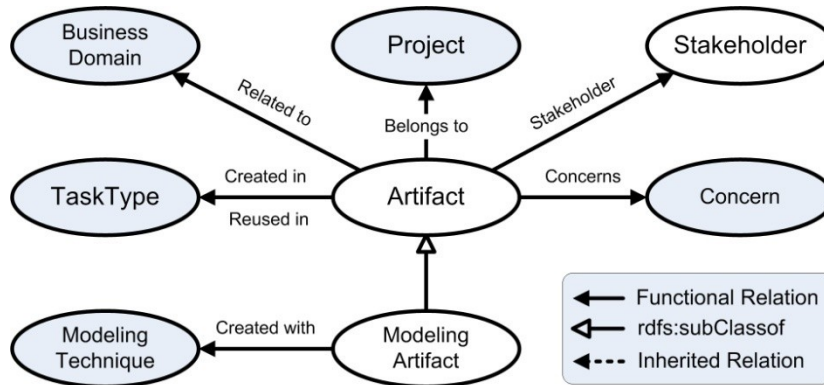


Figure 7-8: Overview of an Artifact's Relations

To this end, on the *ad-hoc-repository level*, the Web Engineering Reuse Sphere's architectural framework introduces *observer agents* which identify new artifacts, extract metadata statements and submit them automatically to the registry (cf. Section 7.5). Thereby, new artifacts become registered automatically only a few moments after their creation or modification - without requiring modifications or extensions to the existing tools or repositories. In the future, when tool and application vendors will have adopted established ontologies which were also incorporated in the *Web Engineering Reuse Ontology* (e.g. the Dublin Core ontology), metadata mapping efforts will significantly be reduced. Beyond that, it would be desirable that the presented ontology is taken on in the Web Engineering research community for including methodology-specific extensions and incorporating it in their associated development frameworks and tools.

On the *infrastructure level*, artifacts are either again stored and registered from within artifact-type-specific tools and applications or submitted via the generic reuse Web portal. In order to allow for submitting artifacts to infrastructure repositories from within the tools they were created or modified with dedicated extensions for communicating with registry or repository Web services as well as dialogs for entering metadata are required. If such extensions are not feasible, the reuse Web portal can be used to store and register artifacts.

In each case, as much metadata as possible is extracted automatically in the same way as described above for the *ad-hoc-level*. However, as registering artifacts on the infrastructure level is – in contrast to the *ad-hoc-level* – an explicit task and metadata quality requirements are much higher, it is reasonable to have the user complement the automatically derived metadata.

In order to gain even more valuable metadata automatically, deriving semantic information from the artifact's context or a user's behavior while working with an artifact seems to be a promising approach. For example, if a particular stakeholder registers an artifact that was created using a particular modeling technique, the stakeholder's current skill set can be automatically augmented by the skills that were required for the employed modeling technique, the related business domain and the used software. Furthermore, similar approaches could be adopted for Web development frameworks. For example, by measuring how long users have worked on a component regarding a particular concern (e.g. presentation, interaction etc.), statements about the major relation of the component to a particular concern could be derived. Likewise, analyzing a component's relative location on a page and thereupon (combined with other aspects) deriving statements about its type, e.g. content component, satellite, menu, landmark or login seems to be promising.

7.5 Reference Architecture Framework

This section presents the *Web Engineering Reuse Sphere's generic reference architecture* serving as a technical platform for the realization of the presented concepts. Furthermore, it forms an architectural framework guiding the integrating of clients and repositories from heterogeneous Web Engineering methodologies. Figure 7-9 gives an overview of the reference architecture framework which was designed based on concepts from the fields of Service-oriented Architecture (SOA) and Enterprise Application Integration (EAI) (Arsanjani 2004; Erl 2005). Corresponding to the sphere concept presented in Section 7.1, the architecture defines a *Registry Layer* for the semantic registry in the sphere's core, a *Repository Layer* for the *Ad-Hoc* and *Infrastructure* repositories and a *Client Layer*.

The *Registry Layer* comprises a *Semantic Web API* being able to deal with the Semantic Web standards Resource Description Framework (RDF) (Klyne and Carroll 2004), OWL and SPARQL, a *Triplet Store* for storing RDF instances as well as the *Web Engineering Reuse Ontology*. The implementation performed in the context of this thesis uses the *Jena Semantic Web Framework* (Hewlett-Packard Development Company 2003). In order to allow for platform-independent storage and retrieval of RDF data as well as for executing SPARQL queries, a CRUDS-based *Registry Web Service* on top of the Jena API was developed. This service forms the central component of the *Registry Layer* and enables distributed clients to perform searches on the *triplet store* or create, read, update and delete metadata in form of RDF statements in a platform-independent way. Furthermore, it supplies up-to-date information about the concepts, relations and attributes defined in the ontology, thus enabling applications to dynamically extend their metadata registration dialogs accordingly.

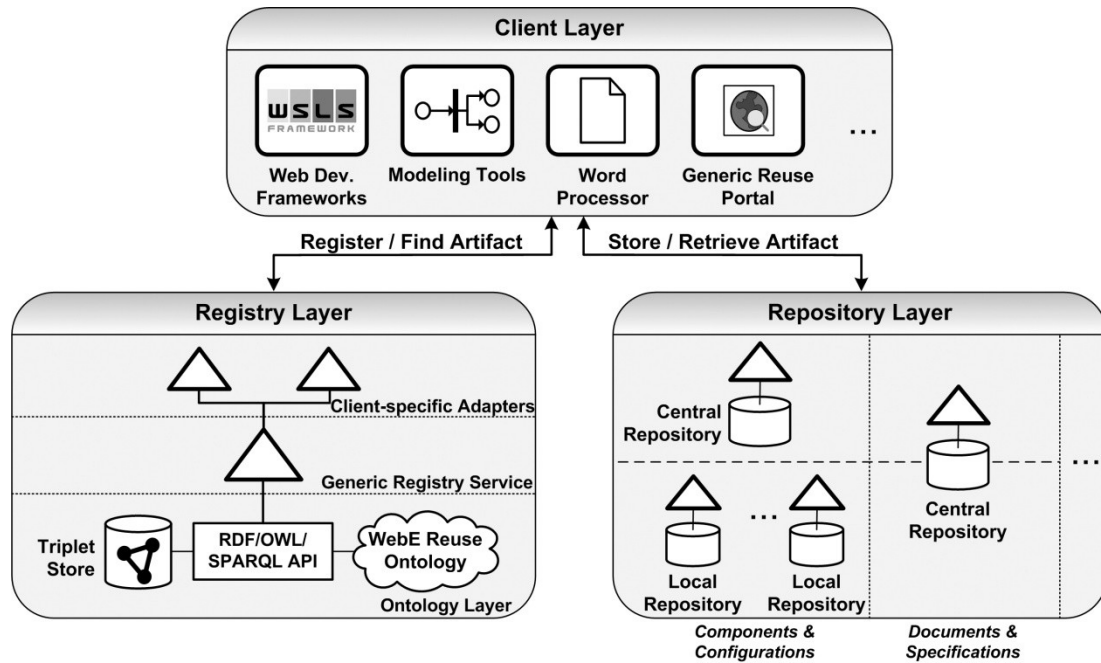


Figure 7-9: The Reference Architecture Framework

As the *Registry Web Service* encapsulates the actual implementation of the *Semantic Web API*, any equivalent, possibly already existing framework could be integrated. Based on the *Adapter* design pattern (Gamma, Helm, Johnson et al. 1995), *Client-specific Web Service Adapters* realizing specific interfaces required by particular client applications can be provided on top of the *Registry Web Service*. Due to such adapters, *Clients* entailing mechanisms for external data source integration can communicate with the *Registry* without requiring modifications to the client application itself.

The *Repository Layer* comprises all repositories on the *ad-hoc* (i.e. *local repositories*) and on the *infrastructure level* (i.e. *central repositories*), covering all types of artifacts, e.g. documents, components, models etc. In order to integrate these heterogeneous repositories into the *Web Engineering Reuse Sphere*, each of them is equipped with a dedicated *Web Service Wrapper*, thus leading to a homogeneous access layer for the distributed repositories. These wrappers share a uniform CRUDS-based interface, allowing for storing, retrieving, updating, deleting and searching versioned artifacts. Beyond that, repositories on the *ad-hoc level* are equipped with *Observer Agents*, being responsible for identifying new or modified artifacts, extracting metadata and registering them via the *Registry Web Service*. This way, the Microsoft Office SharePoint Server 2007 as repository for all kinds of documents and the component and configuration store of the WLS Framework (cf. Section 4.1.2) were exemplarily integrated (Böttger 2008). Thus, the reuse of single configuration properties, fully-configured components or even complete applications via the *Web Engineering Reuse Sphere* is enabled. Besides such database-oriented integration components, a generic *file system-oriented Wrapper* and *Observer* could be used for integrating file-based development frameworks and modeling tools from other Web Engineering methodologies on the *ad-hoc level*.

The *Client Layer* comprises all kinds of client applications participating in the *Web Engineering Reuse Sphere* by storing, registering, finding and retrieving artifacts. In order to integrate such applications, the *plugin* facilities provided by most of today's applications can be used. Alternatively, clients can be integrated based on *Adapter Web Services* as described above. As a first step, the Microsoft Office suite including Microsoft Visio and the *WSLS Framework* were integrated. While the former was achieved based on an *Adapter Service* adhering to the *Microsoft Research Interface* (Fransen 2003), the latter integration is based on a *plugin*.

In this way, a dedicated *Reuse View* was integrated which supports users in performing reuse-related operations via the *Web Engineering Reuse Sphere* from within the *WSLS Framework* (cf. Figure 7-10-1). The *Reuse View* enables creating new *Reuse Projects* (*Create*) or searching and integrating existing *Reuse Projects* (*Checkout*) (cf. Figure 7-10-2). Such a *Reuse Project* can encapsulate single properties, components and their configuration, or even complete trees of configured components. To this end, multi-step dialogs for explicitly registering and storing *Reuse Projects* to a repository on the *infrastructure level* as well as for finding and retrieving them based on the previously described knowledge-based search strategies were integrated. Therefore, these dialogs communicate with the *Registry* and *Repository Web Services*. Furthermore, the *Reuse View* allows for *editing* a *Reuse Project's* metadata and structure, *committing* a new version, *getting* a specific version or *removing* it (cf. Figure 7-10-3). Finally, Figure 7-10-4 shows the case of a domain being part of a superordinate *Reuse Project*.

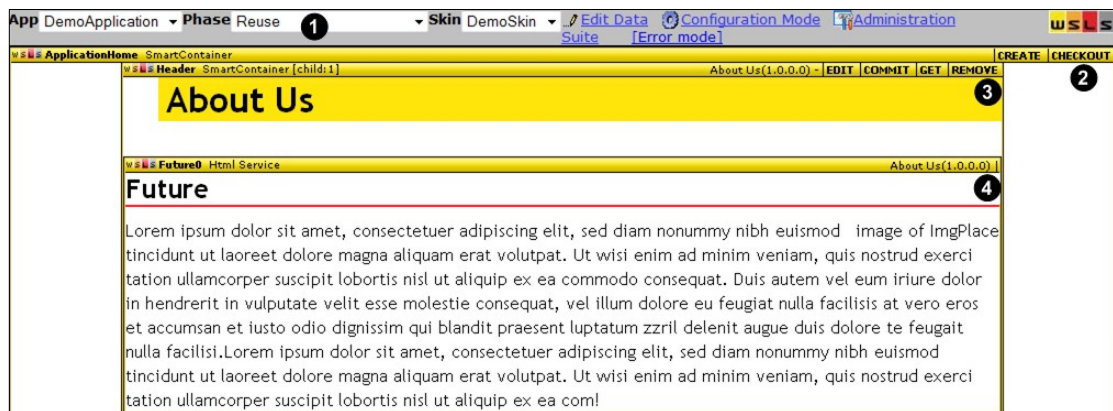


Figure 7-10: The Reuse View in the WSLs Framework

A detailed description of the complete technical implementation can be found in (Böttger 2008). The next step would be the implementation of a *generic Web Portal* serving as central access point for interacting with the *Web Engineering Reuse Sphere*. This could be used in cases where no client-specific plugins are available as well as to support management operations by a *Reuse Librarian*.

7.6 Cross-Methodological Reuse with Stakeholders in Practice

In the following, the Web Engineering Reuse Sphere's application for cross-methodological reuse with stakeholders is illustrated based on this thesis' running example, the 'business trip' scenario. The *WSLS Framework* which also serves as one possible technical platform for the *Web Engineering DSL Framework* (cf. Section 4.1.2) is used as a client application.

As shown in Section 7.2, the *Web Engineering DSL Framework* is – as well as other Web Engineering methodologies – only a specific extension to the presented ontology. Likewise, it was shown how other Web Engineering methodologies can be incorporated. Thus, from the perspective of the *Web Engineering Reuse Sphere* approach, the following examples can be directly transferred to other Web Engineering methodologies. Hence, the achieved improvements are not restricted to the *Web Engineering DSL Framework* approach.

7.6.1 Finding Stakeholder-Tailored Resolution Strategies and Artifacts

The first scenario deals with the realization of the Web-based 'business trip' workflow. For its specification, a variety of diverse stakeholder types, e.g. travelers, secretaries, representatives of the travel department, institute directors etc., with different skills have to be effectively involved. Assuming that a search for existing 'business trip' workflows had no satisfying results, the example scenario starts with an initial conceptual design with a stakeholder from the travel department. Therefore, a suitable *Resolution Strategy* for this *Task Type* has to be determined first.

Thus, the registry search dialog within WSLS is opened and the search strategy *Search for Resolution Strategy and related Artifacts* as depicted in Figure 7-12-1 is selected. In the next dialog, the current process phase *Conceptual Design* and thereupon the desired task type *Design Business Process* are selected (cf. Figure 7-12-2). The available values in the *process phase* and *task* dropdown lists stem directly from the *Web Engineering Reuse Ontology* and were retrieved via the *Registry Web Service*.

Based on the selection, the *Registry Web Service* is called which configures a predefined SPARQL query template with the given task type and executes it, resulting in a set of possible *Resolution Strategies*, *Modeling Techniques*, *Software* and the respectively required *Skills*. The obtained results are not restricted to a particular Web Engineering methodology, but rather encompass all methodologies included in the *Web Engineering Reuse Ontology*. Figure 7-11 shows such an example SPARQL query for the selected task *Design_Business_Process_ModelingTask*.

```

PREFIX atlas<http://mwrq.tm.uni-karlsruhe.de/atlas#>
SELECT ?knowledge ?ktype ?skill ?skilllevel ?resstrategy ?mtech ?software
WHERE
{
  {
    ?skill rdf:type atlas:Skill.
    ?resstrategy atlas:Solves atlas:Design_Business_Process_ModelingTask.
    ?resstrategy atlas:Employs ?mtech.
    ?software atlas:RequiresSkill ?skill.
    ?software atlas:SupportsTask atlas:Design_Business_Process_ModelingTask.
    ?software atlas:SupportsModelingTechnique ?mtech.
    ?skill atlas:Includes ?knowledge.
    ?knowledge rdf:type ?ktype.
  } UNION {
    ?skill rdf:type atlas:Skill.
    ?resstrategy atlas:Solves atlas:Design_Business_Process_ModelingTask.
    ?resstrategy atlas:Employs ?mtech.
    ?mtech atlas:RequiresSkill ?skill.
    ?skill atlas:Includes ?knowledge.
    ?knowledge rdf:type ?ktype.
  } OPTIONAL {
    ?skill atlas:HasSkillLevel ?skilllevel
  }
}

```

Figure 7-11: SPARQL Query for Determining all Modeling and Software Skills related to the Task ‘Design Business Process’ across all Web Engineering Methodologies

Based on the obtained results, the third dialog is constructed (Figure 7-12-3). Therein, either a predefined skill set corresponding to the given stakeholder type or an individual skill level for each knowledge type can be selected. Thereby, stakeholders can restrict the cross-methodological set of available *Resolution Strategies* for the given tasks in accordance with their individual *Knowledge* and *Skills*. In the example, the stakeholder states *expert* skills in *BPMN*, *intermediate* skills in *UML Activity Diagrams* and *novice* skills in *Petri nets*. Regarding software skills, *intermediate* skills in *Microsoft Word* and *novice* skills in *Microsoft Visio* are specified. Thereupon, the selected skills are submitted to the *Registry Web Service* which thereupon again configures a corresponding predefined SPARQL query template and executes it.

The query results in a list of matching *Resolution Strategies*, *Modeling Techniques* and *Software* as well as related *Artifacts*. The query also evaluates ontology relations expressing that particular *Skills* imply other *Skills* or that particular *Documentation Artifacts* can impart missing *Skills*. Finally, the results are ranked according to the matching degree between the specified and inferred skill levels and the required skill levels.

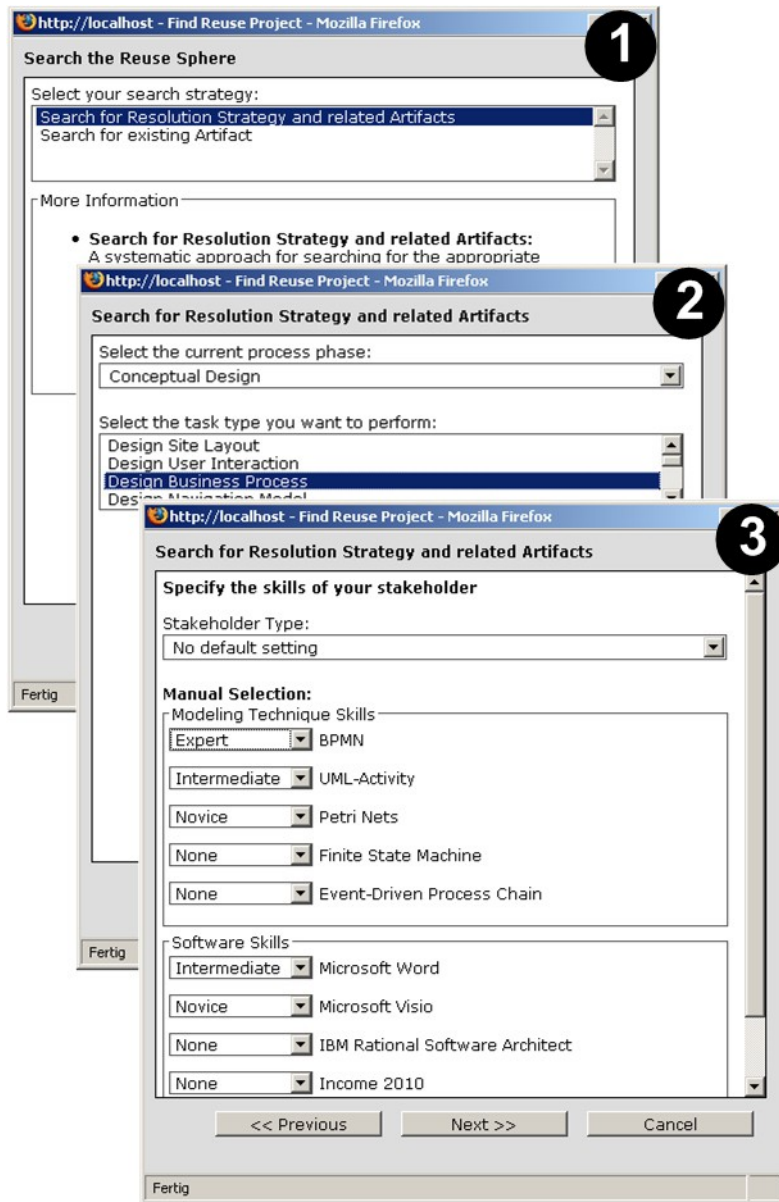


Figure 7-12: The ‘Search for Resolution Strategy’ Wizard

Figure 7-13 shows the search result dialog. The *Resolution Strategy* ‘Workflow DSL’ with the *Modeling Technique* ‘BPMN’ and supported by the *Software* ‘Microsoft Visio’ was identified as a perfect match for the given stakeholder. In the dialog’s *details* panel, the individual elements are listed along with their required skills. As the stakeholder stated only *novice* skills in Microsoft Visio, a link to a *Documentation Artifact* is provided. Moreover, download links for related *Artifacts* (e.g. a Microsoft Visio template for starting the modeling of the workflow) for the selected result are listed. By clicking on the button labeled ‘Reuse this component’, the *Solution Building Block (SBB)* associated with the selected result is inserted in the current WLS development project. It has to be configured with a DSL program which could either be modeled using the downloadable template or searched for by following the link ‘Reuse selected component and find corresponding artifacts’.

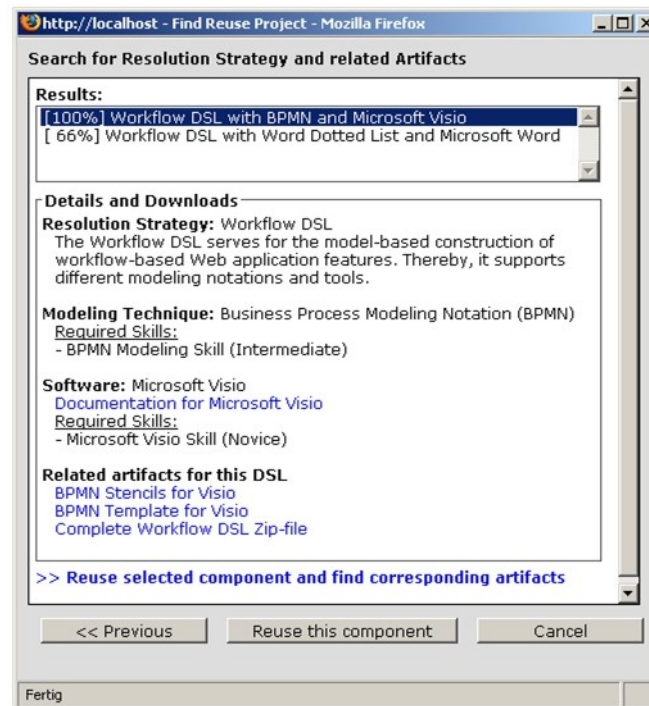


Figure 7-13: The ‘Search for Resolution Strategy’ Wizard

The obtained results as depicted in the figure include only *Resolution Strategies* based on the *Web Engineering DSL Framework*. As mentioned before, this is due to the fact that the WSLs Framework was used as client application which natively filters the results to WSLs-compatible *Resolution Strategies*. However, the same search performed in a more generic client, e.g. a Web-based Reuse Portal as mentioned in the previous section, would comprise results across all Web Engineering methodologies included in the *Web Engineering Reuse Ontology*. For example, if WebML was integrated in the ontology as indicated in Figure 7-3ff, the *Resolution Strategy* ‘WebML Business Process Design’ with the *Modeling Technique* ‘WebML Process Modeling with BPMN’ and supported by the *Software* ‘WebRatio’ along with appropriate templates or documentation would be included. Similarly, the UWE methodology, whose integration was also illustrated in Figure 7-3, would be suitable for stakeholders preferring a UML-based notation and associated tools.

7.6.2 Stakeholder-Oriented Facetted Search and Browsing Facilities

The second scenario concerns the *Expense Report Dialog* within the ‘business trip’ example process. As an adequate or similar dialog could already exist and be used as starting point for adapting it to the given requirements, a search in the *Web Engineering Reuse Sphere’s* registry shall be performed. Therefore, the registry search dialog in WSLs is opened and the search strategy *Search for Existing Artifact* selected (cf. Figure 7-14-1).

In the succeeding dialog, various search facets for specifying query parameters are available (cf. Figure 7-14-2). According to the given scenario, the *Artifact Type* ‘DSL Program’ related to the *Business Domain* ‘Travel’ and the *Concern* ‘Interaction’ that can be used for the *Task Type* ‘Design Dialog’. By selecting a particular *Stakeholder Type*, *Resolution Strategy* or *Modeling Technique*, the query could already be constrained according to the knowledge required for the modification of a found artifact. Beyond that, a *Project* the artifact was created or reused in could be selected as well as keywords for a full-text search specified.

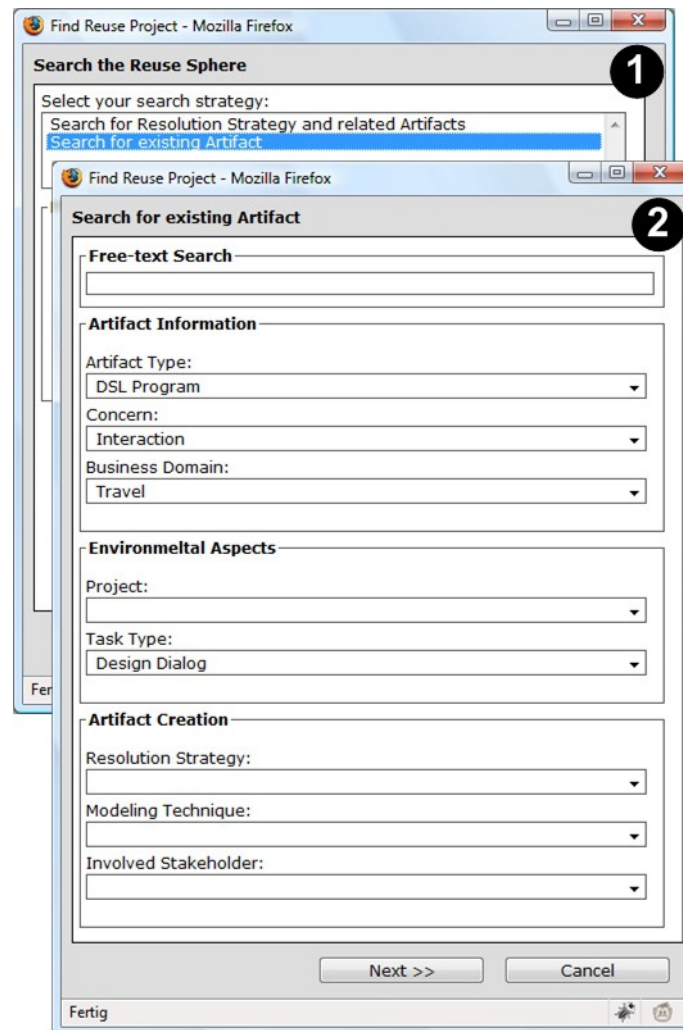


Figure 7-14: The ‘Search for Existing Artifact’ Wizard

The supplied query parameters are submitted to the *Registry Web Service* which inserts them in a predefined SPARQL template and executes it. The search results dialog is depicted in Figure 7-15. This result set covers both the *Web Engineering Reuse Sphere’s ad-hoc and infrastructure levels* and can be filtered according to a stakeholder’s knowledge regarding *Modeling Techniques* and *Software*. Thereby, it can be assured that found artifacts are actually understood by stakeholders and can be reused and modified based on their individual *skills*.

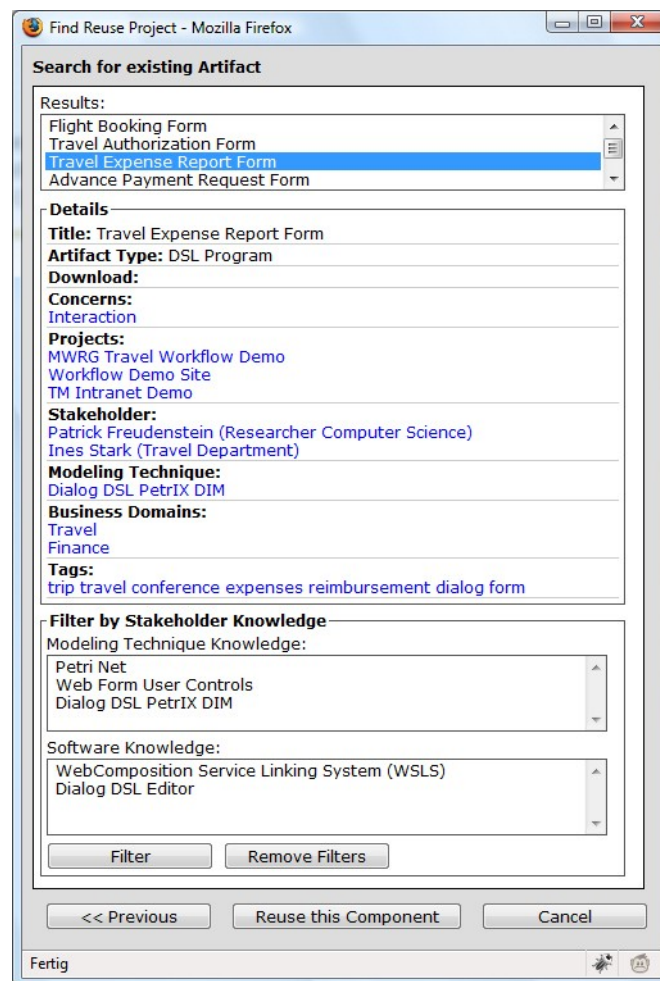


Figure 7-15: Search Results with Browsing and Filtering Facilities

Beyond that, the result set forms a starting point for *browsing* through the registry space. Therefore, most values in the details panel are rendered as hyperlinks allowing for a context switch. For example, other artifacts which were used in the same *Project* or which relate to the same *Business Domain* or *Concern* can be retrieved. Similarly, artifacts created or modified by the same *stakeholder* or *stakeholder type* as well as artifacts sharing the same *Modeling Technique* could be explored. Both browsing and filtering facilities are realized by executing corresponding SPARQL queries via the *Registry Web Service*; an in-depth description of these queries can be found in (Böttger 2008).

Potentially adequate artifacts can be directly and safely integrated in the current WSLS application at runtime. Therefore, a *preview mode* including rollback mechanisms was integrated into the WSLS Framework. Thus, the selected artifact can be safely tested which again improves the communication with stakeholders as well as the effective understanding of reusable assets.

As described for the previous scenario, artifacts from other methodologies could also be found here, e.g. *WebML* or *UWE* dialog models. This could be achieved without any modifications to the SPARQL query or the dialogs. Currently, the result set is being filtered by WSLS so that only compatible results are displayed. However,

assumed that dialog models were interoperable as strived for by the *MDWEnet* Initiative, e.g. based on adequate model transformations, this filtering mechanism could be removed and artifacts could be cross-methodologically found and reused.

7.7 Summary

This chapter presented the *Web Engineering Reuse Sphere*, a novel approach enabling *effective cross-methodological* reuse in the Web Engineering domain. The approach inherently considers *stakeholder characteristics and skills* as key factors for *reuse effectiveness*, i.e. the capability to understand, evaluate and subsequently modify and use reusable artifacts. In the following, the Web Engineering Reuse Sphere's unique solution elements as well as the fulfillment of the requirements elaborated in Chapter 2 are briefly summarized.

The approach establishes a *sphere* concept for a *distributed, cross-methodological* repository space consisting of two spheres for *spontaneous* and *planned* reuse. In its core, a central *ontology-based registry* serves for registering all kinds of artifacts throughout the repository space and provides holistic registration and search functionalities. Therefore, the *Web Engineering Reuse Ontology* was introduced as a *generic, homogenizing* semantic basis for the strongly heterogeneous Web Engineering reuse domain. It was formalized based on *Semantic Web standards* and technologies and provides well-defined *extension points*. As a result, the variety of existing Web Engineering methodologies can be systematically incorporated and thus effective, cross-methodological search strategies be realized. Besides common reuse-related concepts, the ontology places particular emphasis on capturing Web Engineering methodologies' *Resolution Strategies*, *Modeling Techniques* and *Software* for particular *Task Types* as well as the therefore required *Knowledge* or *Skills* respectively.

The *Web Engineering Reuse Ontology* establishes the foundation for novel *knowledge- and inference-based search strategies* which include *stakeholder skills* as an integral search facet. Thus, adequate *Resolution Strategies* and related artifacts for a particular *Task Type* and a particular *Stakeholder* audience or *Skill Set* respectively can be found. Furthermore, it enables finding existing *Artifacts* according to various contextual parameters, again including the current *Stakeholder* audience or required *Skill set* respectively. Beyond that, facilities for *browsing* through the registry space based on the relations specified in the ontology are provided.

The Web Engineering Reuse Sphere's *architectural reference framework* serves as technical support platform and guides the integration of existing (local and infrastructure-based) repositories and clients from heterogeneous Web Engineering methodologies. The non-invasive integration of local *ad-hoc repositories* combined with automated metadata extraction methodologies enables the transparent background registration of artifacts. In this way, artifacts become registered and thus

findable as soon as they are saved for the first time. Due to the resulting *coordinative support*, redundant development efforts can be significantly reduced.

In the context of current consolidation efforts towards interoperability between today's Web Engineering methodologies, e.g. the *MDWEnet Initiative*, the *Web Engineering Reuse Sphere* presents an ideal complement as enabler for real cross-methodological reuse. The presented approach in general and particularly the *Web Engineering Reuse Ontology* form a valuable contribution to the Web Engineering research discipline and received significant attention at the 8th International Conference on Web Engineering (ICWE'08).

The *Web Engineering Reuse Sphere's* cross-methodological nature and sound focus on stakeholder characteristics and skills makes it also an ideal complement to the *Web Engineering DSL Framework*. Stakeholders can autonomously use the advanced search strategies to determine and retrieve an adequate DSL and corresponding DIM notation, modeling software and related artifacts in accordance with their current task and individual skills. Hence, the *Web Engineering Reuse Sphere* forms the starting point for stakeholders and facilitates the DSL-based engineering process, thus making a substantial contribution to its efficiency and effectiveness.

8 Evaluation

Besides the successful theoretical evaluation of the presented solutions against the identified requirements catalog, their actual utility and adequacy have also been examined in practice and empirical studies. First of all, comprehensive technical implementations of the *Workflow DSL* including the *Model Transformation Framework*, the *Dialog DSL* and the *Web Engineering Reuse Sphere* were performed and establish the basis for further studies. Thereupon, they were successfully applied in various real-world scenarios including

- the collaborative reuse-oriented development of the *MWRG Research Site* (MWRG 2009b),
- the currently ongoing re-implementation of the new *MWRG Homepage* (MWRG 2009a),
- the implementation of a workflow-based Web application supporting the *MWRG students advising process* (Buck 2007; Setiawan 2009),
- various prototypical example implementations of Web-based workflows, supporting e.g. the *application for leave and travel reimbursement* processes,
- the construction of advanced Web-based dialogs in the *practical course 'Web Engineering'* in the winter term 2008/09, and
- the model- and component-based development of page-flow-based portal features for the *KIT Employee Portal* and the *KIT Students Portal* in the context of the *KIM project* (cf. Section 8.2).

The experiences gained thereby served as beneficial input for the continuous improvement of the presented approaches. Furthermore, nineteen publications at international workshops, conferences and journals allowed for intensive discussions and valuable feedback by researches from the Web Engineering community and adjacent research areas.

Beyond that, an *empirical evaluation* of Workflow DSL core concepts based on real-world process models from the *KIM project* was performed (cf. Section 8.1). In addition, the Dialog DSL approach's efficiency and effectiveness, particularly with regard to strong stakeholder involvement, were examined in several *formal empirical experiments* presented in 8.3.

8.1 Empirical Evaluation of Workflow DSL Concepts

The goal of this evaluation was to determine whether the concepts contained in the Workflow DSL's *Core Elements Set (CES)*, introduced in Section 5.4.2, achieve a sufficient coverage of real-world business processes. In addition, the adequacy of the *Activity Building Blocks (ABB)* catalog (cf. Section 5.2.2) for the Web-based processing of real-world business processes was analyzed in this context. The evaluation should provide quantitative answers to the following questions:

- **(Q1):** To which extent does the Workflow DSL's Core Elements Set (CES) provide sufficient coverage of real-world business process models?
- **(Q2):** To which extent does the Workflow DSL's Activity Building Block (ABB) catalog provide sufficient coverage for realizing real-world business processes as Web-based workflows?

The evaluation was conducted based on 64 Petri net-based business process models which originated from the project "Karlsruhe's Integrated Information Management (KIM)" (Juling 2005). They cover the domains of event and exam management at the University of Karlsruhe (TH). The models comprise a combined total of 1479 modeling constructs which results in an average of 23 constructs per model. The business process models were originally modeled for analysis and documentation purposes. Thus, they lack formal correctness at some points as well as technical workflow aspects in general.

8.1.1 Expressiveness of the CES in Real-World Process Models

Regarding (Q1), the available set of business process models was analyzed and each identified modeling construct classified according to its CES affiliation. Therefore, a multi-step classification methodology was used (cf. Figure 8-1).

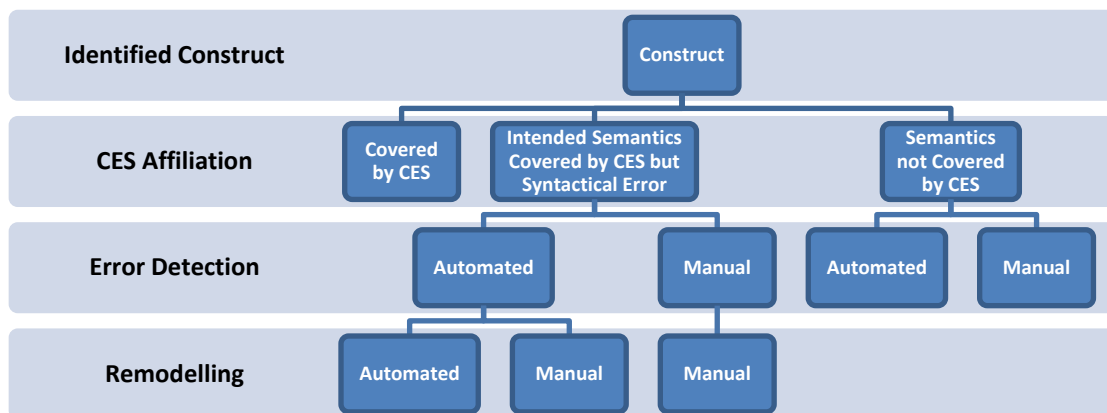
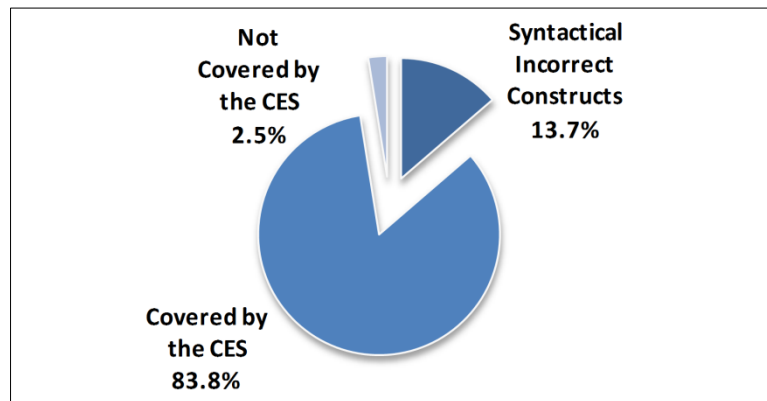


Figure 8-1: Classification Methodology for CES Evaluation

For each identified construct in a business process model, it is determined whether it is covered by the CES or not. Due to modeling errors, a construct which is semantically covered by the CES could require remodeling in order to be syntactically correct. In this case, it is differentiated whether the modeling error or the actual semantic intention respectively could be detected automatically or not. Automatically detectable errors are further classified into automatically resolvable errors and errors that require human remodeling. For constructs whose semantic is not covered by the CES, it is analyzed whether their modeling syntax enables their automated detection or requires manual detection. Manual error detection is required for cases where the incorrect Petri net-based modeling syntax corresponds to a different CES concept. Similarly, manual error resolution is required if multiple ways of resolution exist.

Figure 8-2 illustrates the obtained result after the first classification level. 83.8% of all modeling constructs in the analyzed business process models are immediately covered by the Workflow DSL's CES, 13.7% of the constructs require syntactical remodeling and 2.5% are not covered by the CES.



Constructs (before Remodeling)	Count
Covered by the Workflow DSL's Core Elements Set	1240
Semantically covered, but syntactically incorrect	202
Not Covered by the Workflow DSL's Core Elements Set	37

Figure 8-2: CES Evaluation Result before Remodeling

While the directly obtained degree of CES coverage for the analyzed process models already indicates a good applicability, it is further improved by remodeling the syntactical incorrect constructs. Their rather high fraction (13.7%) stems from the fact that the examined models were originally modeled for documentation purposes and no model verification mechanisms were applied. Analyzing the various syntactical errors showed that 4 out of 14 error patterns make up 91.85% (185 out of 202). These error patterns are primarily related with the non-compliant modeling of the *Workflow Data* concept. As the errors are due to a different modeling variant than expected by the Workflow DSL, it can be considered very likely that most of

these errors would not have occurred if the process models were designed in the context of the Workflow DSL.

After the error resolution via remodeling, 97.5% of the examined business process model's constructs are covered by the Workflow DSL's CES whereas only 2.5% remain uncovered. Table 8-1 summarizes the frequency distribution of the identified CES concepts, syntactical modeling errors and not covered modeling constructs.

Table 8-1: Frequency of Workflow Concepts Before and After the Error Resolution by Remodeling

Identified Constructs	Count (Percentage)	
	Before Remodeling	After Remodeling
Constructs Covered by CES Concepts		
Workflow Process	64 (4.33%)	64 (4.14%)
Activity (including hierarchical activities)	393 (26.57%)	429 (27.75%)
Start Node	63 (4.26%)	63 (4.08%)
End Node	58 (3.92%)	58 (3.75%)
Sequence	140 (9.47%)	154 (9.96%)
AND-Split	20 (1.35%)	30 (1.94%)
AND-Join	16 (1.08%)	27 (1.75%)
XOR-Split	10 (0.68%)	13 (0.84%)
XOR-Join	11 (0.74%)	14 (0.91%)
OR-Split	0 (0.00%)	0 (0.00%)
OR-Join	0 (0.00%)	0 (0.00%)
Structured Loop (Do-While-Loop)	4 (0.27%)	17 (1.10%)
Structured Loop (While-Do-Loop)	5 (0.34%)	11 (0.71%)
Workflow Data	0 (0.00%)	173 (11.19%)
Participant	118 (7.98%)	118 (7.63%)
Application	338 (22.85%)	338 (21.86%)
Syntactically Incorrect Constructs		
14 different error patterns	202 (13.66%)	0 (0.00%)
Constructs Not Covered by CES Concepts		
10 different patterns	37 (2.50%)	37 (2.50%)

The deviation between the concepts *Activity* and *Application* originates from hierarchical activities encapsulating complete sub-processes and thus not being assigned with an application. The divergence between *Start Node*, *End Node* and *Workflow Process* is due to irresolvable modeling errors as well as constructs not

covered by the CES. Beyond that, it should be noted that the occurrence of one control-flow concept usually implies one *Activity*, one *Application* and possibly one *Workflow Data* instance. As a consequence, the proportion of non-control flow concepts is much higher than the proportion of control-flow concepts. This ratio is even further multiplied by *Split*- or *Join*-typed control flow concepts. A detailed presentation of the particular error patterns and modeling constructs not covered by the CES can be found in (Setiawan 2009).

The observed frequency distribution of CES-based workflow concepts in the examined business process models as well as the fraction of uncovered constructs is illustrated in Figure 8-3. The low percentage of not covered constructs corresponds with similar studies, particularly if the reasons therefore and the rather unintuitive ratio between control-flow and non-control-flow concepts explained above are taken into account. For example, a recent study on the frequency distribution of BPMN construct usage arrived at the conclusion that average BPMN models use less than 20% of the available vocabulary, i.e. about nine core concepts or symbols respectively (Zur Muehlen and Recker 2008).

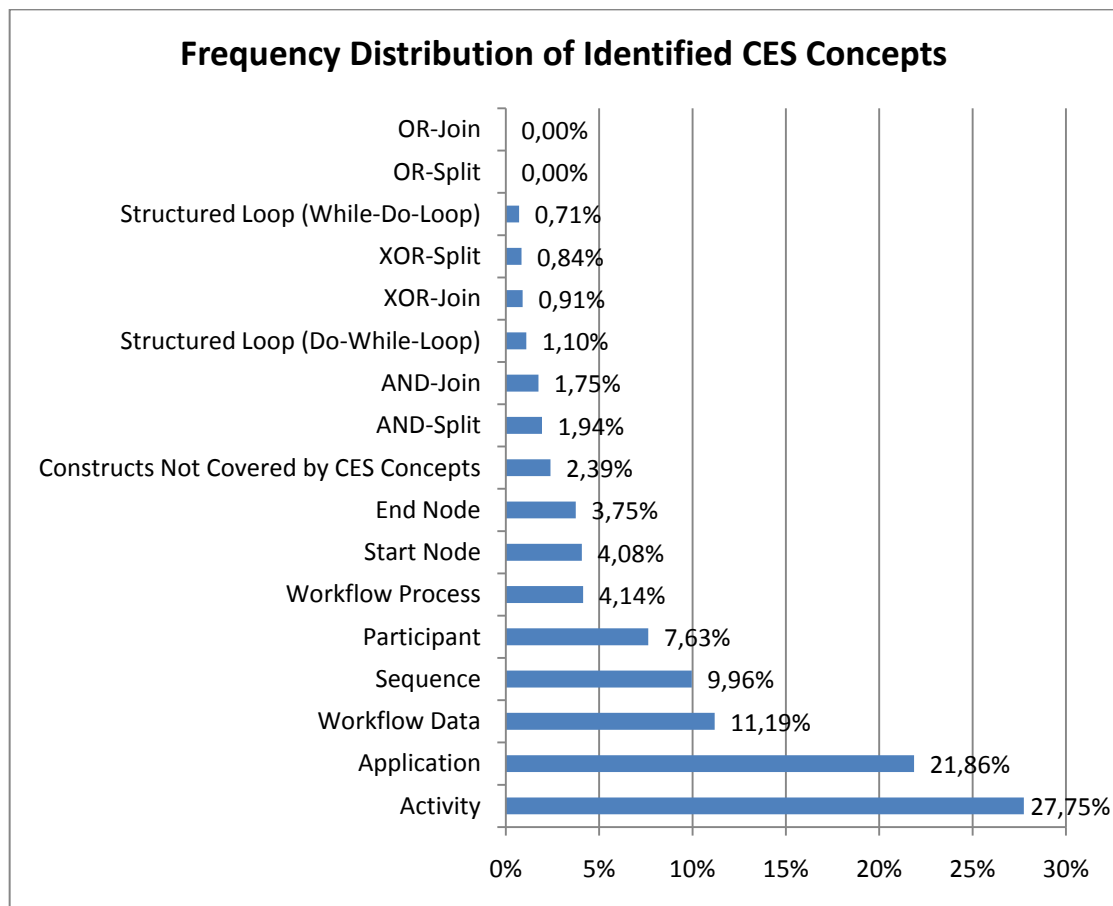


Figure 8-3: Observed Frequency Distribution of Identified CES Concepts in the Evaluated Business Process Models

As the examined process models focus primarily on a particular business domain and were modeled by only a small group of analysts with basic to intermediate modeling skills, the obtained results cannot be directly generalized. However, regarding the

CES' significant degree of coverage (97.5%) in the examined models as well as the fact that similar studies point in the same direction, the Workflow DSL's reasonable applicability to real-world scenarios can be considered confirmed.

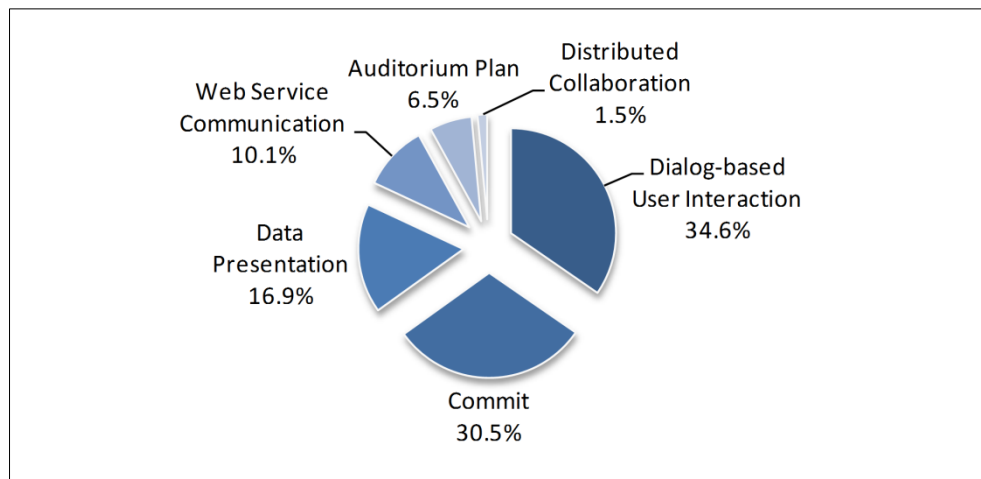
8.1.2 Coverage of Real-World Process Activities by the ABB Catalog

With regard to the evaluation question (Q2), all identified business process activities were analyzed whether a single or a combination of the defined ABBs could serve for their adequate Web-based processing. Furthermore, it was analyzed whether additional ABBs are required and should be incorporated in the catalog.

Figure 8-4 illustrates the found distribution of workflow activity types and their coverage by the Workflow DSL's ABB catalog. The results show that 69.5% of the activities can be effectively supported by Web-based user interfaces or Web Service communication whereas the remaining 30.5% are performed offline, e.g. marking and handing out exams or holding a lecture, and were thus mapped to the *Commit* ABB. Regarding the remaining ABBs in the Workflow DSL's catalog, *Dialog-based User Interaction* makes up the most frequently used ABB (34.6%). Thereafter, the *Data Presentation* ABB follows with 16.9% and the *Web Service Communication* ABB is used by 10.1% of all activities. It should be noted that the latter fraction covers only activities which are exclusively realized by the *Web Service Communication* ABB. In fact, this ABB is heavily used in combination with the other two ABBs: 91.5% of the activities realized by the *Dialog-based User Interaction* ABB and 98.3% of the activities realized by the *Data Presentation* ABB require a preceding and/or succeeding Web service invocation for retrieving or storing data.

Beyond that, two new ABB candidates were identified. A *Distributed Collaboration* ABB could realize Web-based, audiovisual-enabled meetings for distributed participants. It could be used by 1.5% of the examined activities and could substitute offline face-to-face meetings which would otherwise be covered by the *Commit* ABB. The *Auditorium Plan* forms a domain-specific specialization of the *Data Presentation* ABB and is used for visualizing graphical auditorium plans including detailed information regarding capacity, equipment etc. It is used by 6.5% of the examined activities and could be realized by the *Data Presentation* ABB and a specialized configuration set.

In summary, the evaluation confirmed that the Workflow DSL's ABB catalog provides full coverage for the examined 338 activities. Thus, their high degree of reusability as well as their generic and well-defined specification was approved. Furthermore, it was observed that the degree of real Web-based activity processing could be further increased by incorporating an ABB for *Distributed Collaboration*. A specialized variant of the *Data Presentation* ABB for Auditorium Plans could decrease redundant development efforts.



Activity Building Block Type	Count
Total Dialog-based User Interaction	117
<i>Dialog-based User Interaction</i>	10
<i>Dialog-based User Interaction combined with WS Communication</i>	107
Total Data Presentation	57
<i>Data Presentation</i>	1
<i>Data Presentation combined with WS Communication</i>	56
Web Service Communication	34
Commit	103
Distributed Collaboration	5
Auditorium Plan	22

Figure 8-4: Identified Workflow Activity Types and their Coverage by the Workflow DSL's Activity Building Block Catalog

8.2 Workflow DSL Concepts Applied in the KIM Project⁶

The project „Karlsruhe's Integrated Information Management (KIM)” pursues the goal of increasing the excellence in teaching at the Karlsruhe Institute of Technology (KIT) (Juling 2005). Therefore, it strives for a continuous and sound integration of relevant legacy systems and data as well as for increasing the accessibility and transparency of related business processes. The KIM project's technical realization is based on a Service-Oriented Architecture (SOA) model which was termed 'KIM integrated Service Oriented Architecture (KIM iSOA)' (Freudenstein, Liu, Majer et al.

⁶ Parts of this section have been published in (Freudenstein, Nussbaumer, Majer et al. 2007)

2006; Freudenstein, Majer and Maurer 2006). Figure 8-5 gives an overview of the KIM iSOA, its four core layers and cross-cutting concerns.

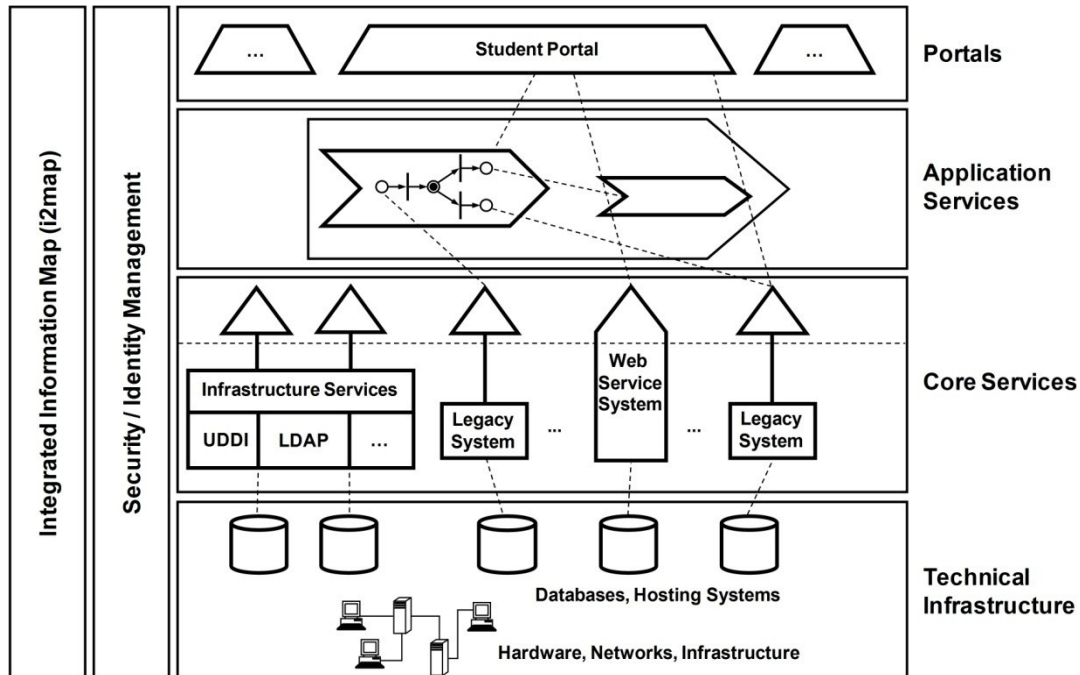


Figure 8-5: The KIM iSOA

Within the depicted layer model, Web-based portals present central and uniform access points to relevant information and business processes for diverse audiences. In the course of the KIM project, a KIT Students portal and a KIT Employee portal were developed (Allerding, Buck, Freudenstein et al. 2008). As both portals strongly rely on data and operations encapsulated in Web services located on the *Core* and *Application Services* layers (Freudenstein, Majer, Maurer et al. 2007), a multitude of Web service integration scenarios had to be implemented. Therefore, portal components realizing the service communication as well as the rendering of appropriate interaction and presentation structures were required (Freudenstein, Majer and Nussbaumer 2008).

Analyzing the various integration scenarios showed that their requirements can be quite complex and span across a variety of functional aspects: presentation and interaction aspects as well as aspects in the fields of data and service communication. While simple integration scenarios comprise only a parameterized service communication followed by the presentation of the received data, much more complex user interaction sequences of dialogs, service communication and data presentation are found in practice. Given this complexity and the emerging variety of Web services in medium and large SOA-based systems which have to be made accessible to users, an efficient approach for the integration of services in portals was required. Developing dedicated portal components for each single integration scenario turns out to be too cost- and time-consuming, aggravates operations and maintenance and the enforcement of quality standards.

8.2.1 FSM-based Modeling of User Interaction Workflows using ABBs

To this end, a novel approach for modeling the user interaction with Web services including a technological support framework for its application within existing portal systems was developed. This approach was developed in 2006 and formed a valuable first step towards the *Workflow DSL* approach presented in Chapter 5 of this thesis.

The approach considers integration scenarios as *user interaction (UI) workflows* composed of generic activity building blocks. Therefore, the *Workflow DSL's* catalog of *Activity Building Blocks (ABBs)* introduced in Section 5.2.2 was successfully adopted. UI workflow models can either be derived from business process models or, due to their simple and intuitive modeling notation, designed from scratch with strong stakeholder collaboration (Freudenstein, Nussbaumer, Majer et al. 2007). The resulting models are executed by a generic portal component. Thus, realizing complex Web service integration scenarios in portals is reduced to composing ABBs along a UI workflow.

The modeling notation for UI workflows is based on Finite State Machines (FSM). FSMs were chosen as they are more appropriate for modeling flexible navigational behavior than more rigid sequence-oriented process modeling approaches. A user view (e.g. a search form) is thus represented by a *state* and the user navigation between views by triggering events (e.g. clicking on a button) corresponds to *transitions*. The ABBs are used for specifying *entry actions* for the particular states. Figure 8-6 depicts an example model for the 'Course Registration' integration scenario which represents a portal feature supporting students in the process of searching and registering for courses at the beginning of a semester. In this scenario, a Web service providing comprehensive course information based on a course management legacy system and another Web service providing access to course assignment data, i.e. which student has registered for which courses, are integrated.

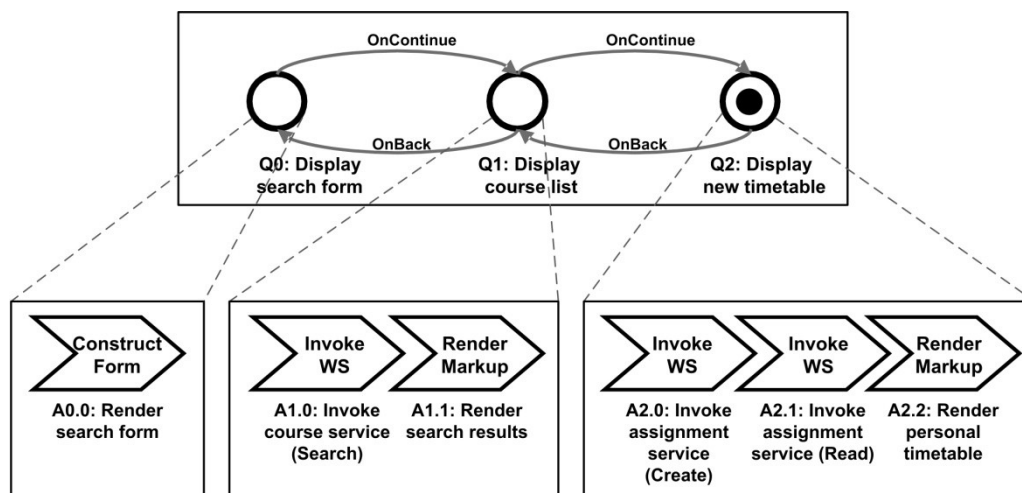


Figure 8-6: FSM-based Model of the 'Course Registration' Integration Scenario

The two-layered model of the "course registration" UI workflow can be formally defined in terms of a FSM as $W = (Q, \Sigma, \delta, q_0, F, A)$ with

$Q = \{Q_0, Q_1, Q_2\}$: Set of user views

$\Sigma = \{OnContinue, OnBack\} = \Sigma_{\text{default}}$: Set of events which can be triggered by a user. Event sets that are likely to recur again in the future are defined as normalized Σ clusters, thus easing reuse in the implementation phase.

δ : State transition function, i.e. possible navigation paths between the user views $\delta: Q \times \Sigma \rightarrow Q$

$q_0 = \{Q_0\}$: Initial user view

$F = \{Q_2\}$: Set of final user views

$A = \{a_{q,i} \mid q \in Q, i \in \mathbb{N}_0\} = \{a_{0,0}, a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, a_{2,2}\}$: Set of entry actions to be performed when entering state q

The set of user views consists of three states: In the first state, Q_0 , a search form for specifying the parameters for the course search is displayed to the user. Therefore, a ‘ConstructForm’ ABB (which corresponds to the *Dialog-based User Interaction* ABB) for generating the search form is executed ($a_{0,0}$) when entering Q_0 . Having filled out and submitted the form, whereby the event Σ_0 ‘OnContinue’ is triggered, the user arrives in state Q_1 , the search results list. When entering Q_1 , an ‘InvokeWS’ activity (which corresponds to the *Web Service Communication* ABB) is being executed ($a_{1,0}$) and runs a search against the course information Web service based on the search parameters defined in Q_0 . Afterwards, a ‘RenderMarkup’ activity (which corresponds to the *Data Presentation* ABB) renders the Web service response in form of a search results list ($a_{1,1}$). Using a corresponding button for activating the event Σ_1 ‘OnBack’ in Q_1 , the user can navigate back to the search form (Q_0). In Q_2 , the user has been registered for the selected course and her personal timetable including the new registration is being displayed. Therefore, three entry activities have to be executed when entering the state: First, an ‘InvokeWS’ activity accomplishes the registration for the selected course by creating a new registration record for the given course and student via the assignment Web service ($a_{2,0}$). Subsequently, the current list of course registrations for the given student is retrieved from the assignment Web service, again using an ‘InvokeWS’ activity ($a_{2,1}$). Finally, a ‘RenderPresentation’ activity uses the received assignment data and renders the student’s personal timetable ($a_{2,2}$).

Beyond multi-step UI workflows like this, also simple scenarios consisting only of only one step, e.g. invoking a Web service and rendering the result, can be realized with this approach.

8.2.2 Technical Framework for Executing UI Workflows in Web Portals

Figure 8-7 gives an overview of the technical framework’s architecture consisting of four layers: The bottom layer contains the Web services to be integrated in the portal. Above, the ‘UI Workflow’ layer comprises FSM-based workflow instances as described in the previous section. The ‘Data Exchange Service (DES)’ layer holds

mediating components decoupling workflows from the clients executing them. Therefore, a DES component offers a well defined interface to both parties based on the set of possible user events Σ , e.g. Σ_{default} . The top layer contains instances of a generic portal component which is able to instantiate all kinds of workflows and to send and receive events to or from them via the DES layer.

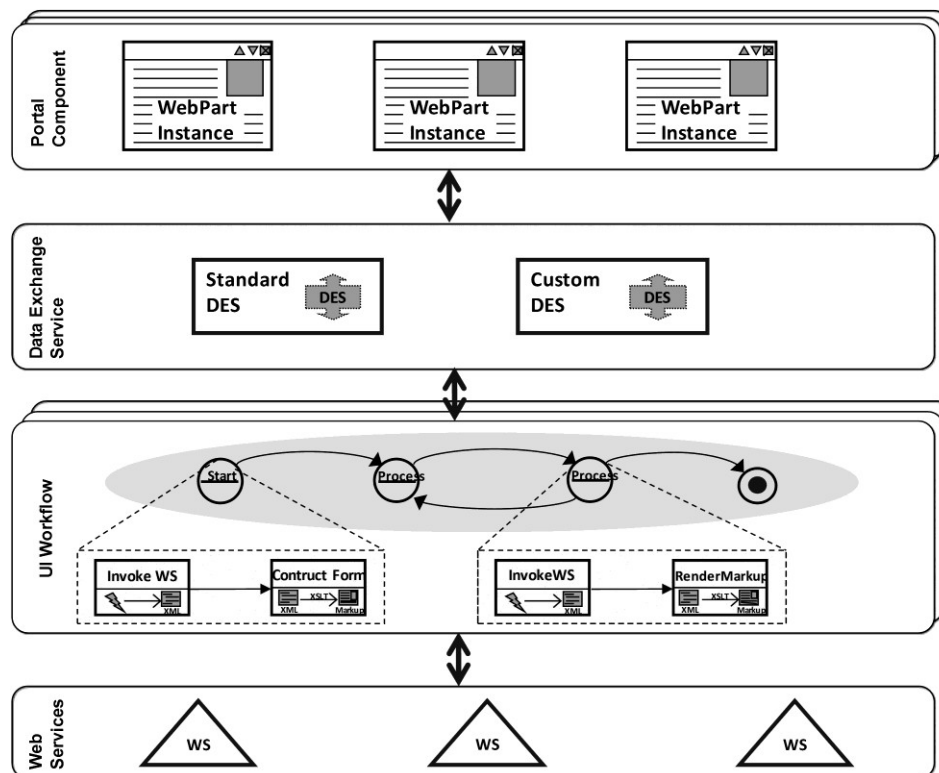


Figure 8-7: The Technical UI Workflow Integration and Execution Framework

The implementation used in the KIM project is based on the Microsoft Windows Workflow Foundation (WF) as workflow engine. The FSM-based UI workflows as well as the entry action sequences can be modeled very comfortably using a graphical editor within Visual Studio 2005 (Figure 8-8, 1+2). The ABBs were implemented as highly configurable software components, so-called 'Custom Activities'. In contrast to the ABB's implementation for the *Workflow DSL*, they are not used as autonomous portal components, but rather act as functional libraries which return markup to the UI workflow and thus ultimately to the generic portal component. When modeling an UI workflow, they can be easily integrated and configured via drag-and-drop and a dedicated property editor. Regarding the portal component layer, a generic 'Web Part' component for the Microsoft Office SharePoint Server 2007 was developed. It is configurable in terms of the UI workflow library to be executed and the DES component to be used for communicating with the UI workflow. This portal component is rather simple as its only functionality lies in receiving markup from the UI workflow via the DES, rendering it and sending back events triggered by a user, again via the DES. Hence, portal components for other portal systems could be easily implemented. To this end, e.g. for non .NET-compatible platforms, the Windows Workflow Foundation supports the automated encapsulation and exposure of workflows via Web service endpoints.

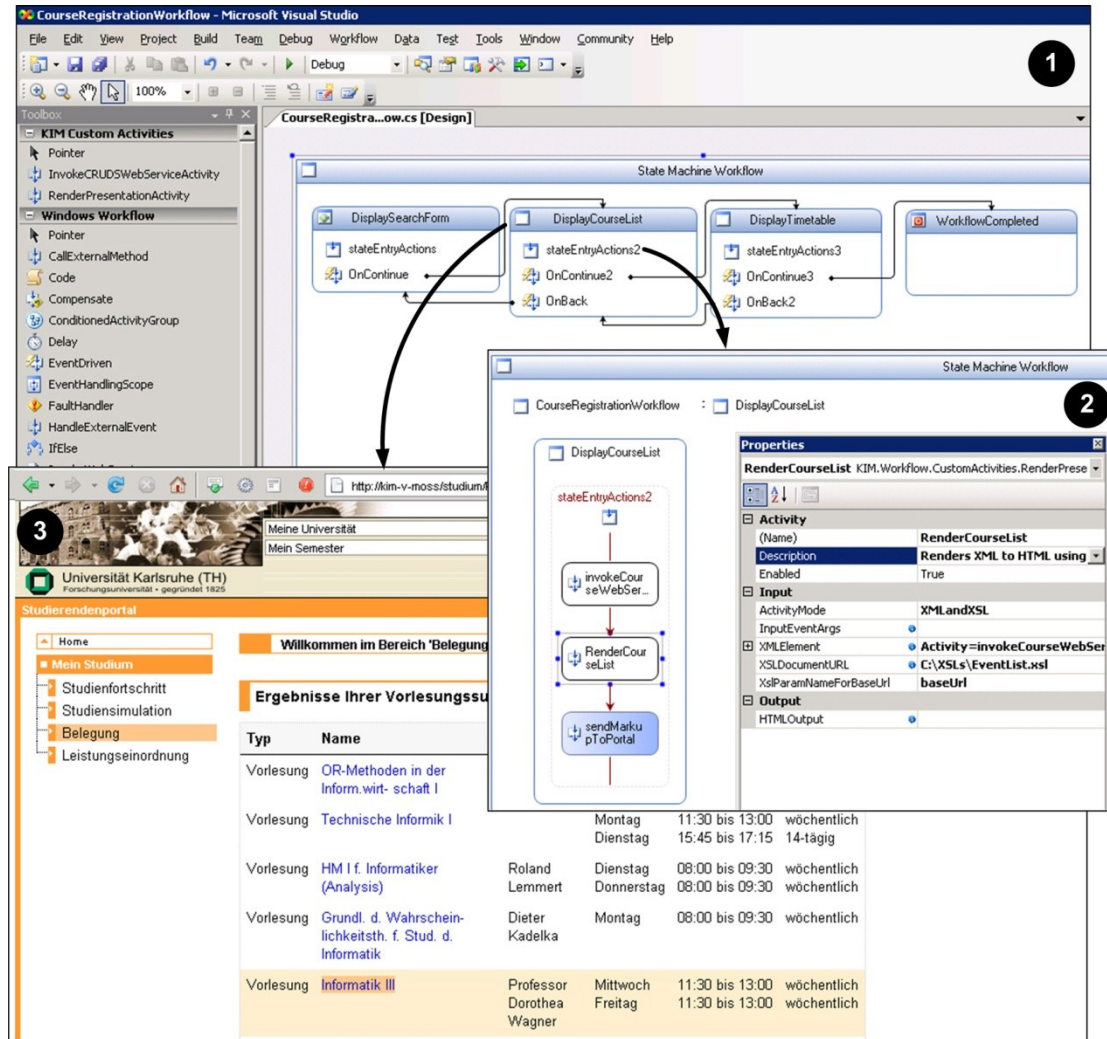


Figure 8-8: UI Workflow Modeling in Visual Studio 2005 (1: FSM, 2: Entry Actions) and its Execution by an UI Workflow WebPart Instance in the Students Portal (3)

8.2.3 Experiences

To date, numerous portal features were developed using the presented approach, e.g. features for applying for a business cell phone, changing passwords, registering an email address, or performing a self-assignment to the KIT's competence fields. Thereby, the high efficiency and flexibility when realizing new integration scenarios or adapting existing ones were identified as the approach's main advantages. This is particularly due to the sound combination of model- and component-based concepts which can also be found in the *Workflow DSL* approach. In this regard, the catalog of highly reusable *Activity Building Blocks (ABB)* presents a core pillar of both approaches and strongly contributes to development efficiency and effectiveness.

Beyond that, as expected, the UI workflow model's FSM-based visualization turned out to be much more comprehensible for new colleagues and stakeholders than purely code-based developed features. The two-layered modeling approach fosters

clarity and analogy with the behavior observed in the portal. Similar to the *Workflow DSL* approach, the focus lies on the various states and transitions between them and technical complexity is reduced to the ABB's configuration facilities. Likewise, as in the *Workflow DSL* approach, the process model is graphically visible and rather self-documenting instead of being weaved into comprehensive code.

Furthermore, only in few cases, special entry actions requiring manual coding were needed. Thus, the ABBs can reduce the set of required development skills to Web standards like XSLT and HTML and abstract from programming languages and APIs specific to a particular portal system. This was particularly beneficial for developers focusing on other aspects, e.g. Web service development. Due to the presented modeling approach and particularly the ABB catalog, they were enabled to autonomously realize Web service-based portal features.

This advantage analogically applies to the *Workflow DSL* approach which even enables stakeholders to autonomously contribute to the solution being built by using notations and tools they are familiar with. In the context of the *Workflow DSL* approach, which is a successor of the UI workflow approach presented in this section, the ABBs were implemented as DSLs including dedicated modeling notations and editors. For example, due to the *Dialog DSL* which corresponds to the *Dialog-based User Interaction ABB*, advanced Web-based dialogs can be completely specified on a model basis. Thus, in the great majority of scenarios, no manual coding is required at all.

8.3 Formal Empirical Evaluation of the Dialog DSL

In order to gain sustainable experiences of the Dialog DSL approach's efficiency and adequacy for heterogeneous stakeholders with diverse backgrounds, two empirical evaluations were conducted. Based on a formal experiment, the Dialog DSL's efficiency in developing complex Web-based dialogs and in adopting changes was analyzed. The Dialog DSL's modeling notation was evaluated based on a survey focusing on its applicability for validating, modifying and creating dialog models by heterogeneous stakeholders.

8.3.1 Experimental Evaluation of Development and Change Efficiency

The experiment's goal was to analyze the influence of a dialog development methodology on the efficiency of development and change adoption. As formal experiments are characterized by a high level of execution and measurement control, the obtained results can be well generalized within the experimental conditions (Wohlin, Runeson, Höst et al. 2000). Besides the Dialog DSL, a second adequate development methodology for Web-based dialogs was used as a basis for comparison in the experiment. This counterpart should be a widely-used

development methodology which supports both visual and code-based development styles. For this reasons, the ASP.NET Framework supported by the development environment Microsoft Visual Studio 2005 was used. The focus of the evaluation was exclusively put on implementing dialog behavior including usability best practices and dialog appearance in terms of user controls and layout. Other aspects like dialog processing or detailed visual design were explicitly left out due to comparability reasons. In a preliminary experiment, a significant falsification of results in favor of the Dialog DSL originating from these factors was observed. For example, the Dialog DSL provides comprehensive support for automated XML- or Web service-based dialog processing whereas ASP.NET requires extensive manual implementation. Considering that specialized frameworks could be possibly adopted in this regard, these factors were omitted in order to achieve more meaningful and comparable experimental results.

The experiment was structured based on the *Goal/Question/Metric (GQM)* approach (Basili, Caldiera and Rombach 1994) as shown in Table 8-2.

Table 8-2: GQM Plan for the Experimental Evaluation of the Dialog DSL's Efficiency

Goal 1	Empirical evaluation of the Dialog DSL approach's efficiency from the developer perspective
Question Q1.1	How efficiently can complex Web-based dialogs be developed using the Dialog DSL approach compared to ASP.NET?
Metric M1.1.1	$\frac{avgTime_{Dialog\ DSL}}{avgTime_{ASP.NET}}$
Metric M1.1.2	#Errors: Number of errors in the developed dialogs
Question Q1.2	How efficiently can changes be incorporated into existing, complex dialogs using the Dialog DSL approach compared to ASP.NET?
Metric M1.2.1	$\frac{avgChangeTime_{Dialog\ DSL}}{avgChangeTime_{ASP.NET}}$

Accordingly, the following hypotheses were formulated:

- *Null hypothesis:*

$$H_{1,0}: avgTime_{DialogDSL} = avgTime_{ASP.NET}$$

Alternative hypothesis:

$$H_{1,1}: avgTime_{DialogDSL} < avgTime_{ASP.NET}$$

- *Null hypothesis:*

$$H_{2,0}: avgChangeTime_{DialogDSL} = avgChangeTime_{ASP.NET}$$

Alternative hypothesis:

$$H_{2,1}: avgChangeTime_{DialogDSL} < avgChangeTime_{ASP.NET}$$

➤ *Null hypothesis:*

$$H_{3,0}: Eff(DialogDSL) = Eff(ASP.NET)$$

Alternative hypothesis:

$$H_{3,1}: Eff(DialogDSL) > Eff(ASP.NET)$$

Thereby, the total time including development, incorporation of changes as well as compensation times for errors is considered.

The selection of subjects for the experiment corresponds to a *convenience sampling*. Therefore, eight students from the practical course on Web Engineering (winter term 2008/09) at the Karlsruhe Institute of Technology (KIT) served as participants. When applying for the course, they did not know about the experiment. Their interest in the Web Engineering discipline and the course itself were the sole aspects for their inclusion in the experiment. No further selection was performed. The participation in the experiment was rewarded with a special certification.

Throughout the practical course, all participants received extensive training and completed exercises in Web standards and technologies and particularly in ASP.NET and Visual Studio 2005. This included also the development of a comprehensive Web-based dialog using ASP.NET and Microsoft Visual Studio 2005. Based on their performance during the first 3 months, the participants were rated by the course advisors and distributed into two balanced groups which is a proven methodology (Prechelt 2001). Table 8-3 illustrates the resulting allocation.

Table 8-3: Performance-Based Subject Allocation into Two Balanced Groups

Subjects	Treatment 1: ASP.NET	Treatment 2: Dialog DSL	Performance in the practical course on WebE (between -2 and +2)
1	X		2
2		X	0.5
3	X		-2
4	X		2
5		X	0
6	X		0
7		X	0.5
8		X	1

Group 1 (Dialog DSL) includes subjects 2, 5, 7, and 8. Group 2 (ASP.NET) includes subjects 1, 3, 4, and 6.

Group	Subjects	Average Performance Level
Group 1 (Dialog DSL)	2, 5, 7, 8	0.875
Group 2 (ASP.NET)	1, 3, 4, 6	1

The experiment design was successfully evaluated regarding *internal validity*, *external validity*, *construct validity* and *conclusion validity* (Chouchane 2009). Only minor improvements regarding the selection and rating of the participants by the advisors were identified and should be considered in subsequent experiments. For example, a more heterogeneous subject population, not only covering graduate students but also various types of industry practioners would be desirable. While the

selected study group enabled high conclusion validity, it results in a reduced external validity, i.e. the degree to which the results can be generalized for the whole software industry.

During the execution of the experiment, each participant had to develop an identical Web-based dialog using the assigned methodology. The dialog was a comprehensive, multi-step travel booking dialog including selection-dependent inputs, hint and help texts and input validations. It was precisely specified based on screenshots, annotations and textual descriptions. In the second part of the experiment, each participant had to adopt a predefined change in the developed dialog, i.e. a large dialog unit should be divided into several smaller units which had to be connected by appropriate navigation facilities. The complete experiment material can be found in (Chouchane 2009).

Figure 8-9 illustrates the experiment processes for both treatments. In the beginning, every participant had to fill out a questionnaire regarding relevant skills and experiences. As the Dialog DSL group had no prior knowledge of the Dialog DSL approach, a short introduction including a 15 minutes trial period regarding the usage of the Dialog DSL's Web-based editor followed. The subsequent experiment process was identical for both groups. Both experiments were conducted consecutively on the same day.

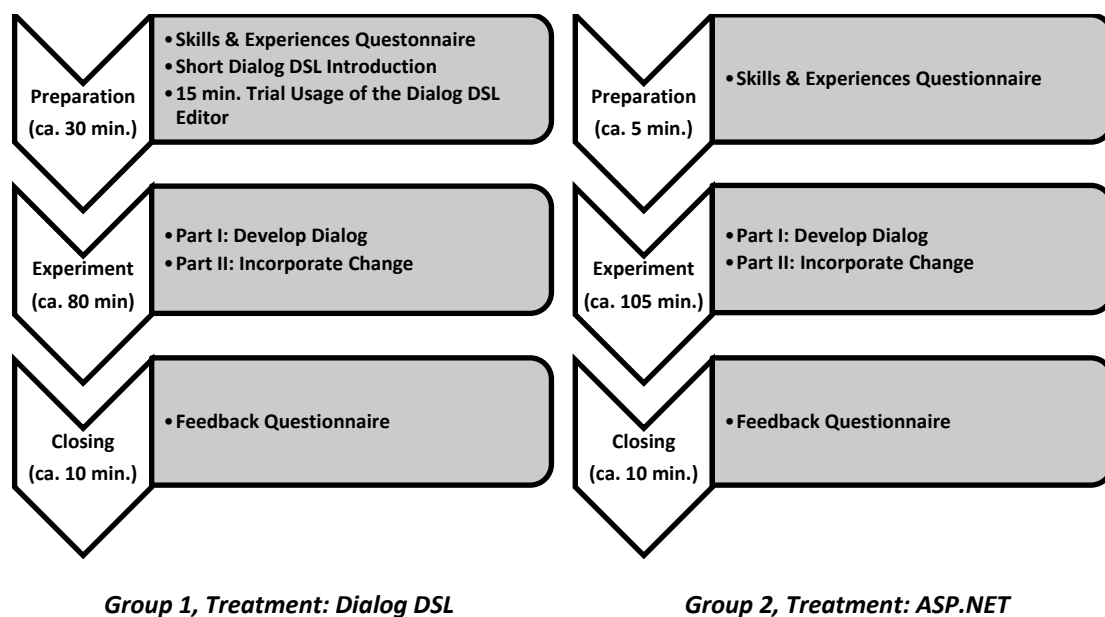


Figure 8-9: Experiment Processes for the Two Groups or Treatments Respectively

Figure 8-10 and Figure 8-11 illustrate the self-assessment-based skills of both groups which were derived from the skills questionnaire which was filled out before the beginning of the experiment. The results reflect the fact that all students received extensive ASP.NET and Visual Studio training during the practical course. Furthermore, it turned out that the ASP.NET group stated overall better skills than the Dialog DSL group. This is partly due to the fact that the ASP.NET group received an additional 'Dialog-development with ASP.NET and Visual Studio 2005' tutorial one

and a half day before the experiment which was strongly tailored to the skills needed for the experiment.

This skill difference forms a good precondition as it further contributes to the experiment's validity. The ASP.NET group can thus be considered representatively skilled whereas the Dialog DSL group corresponds to the Dialog DSL's goal of empowering poorly skilled stakeholders as well. To this end, the Dialog DSL group comprised even an office administrative assistant without an IT or development background.

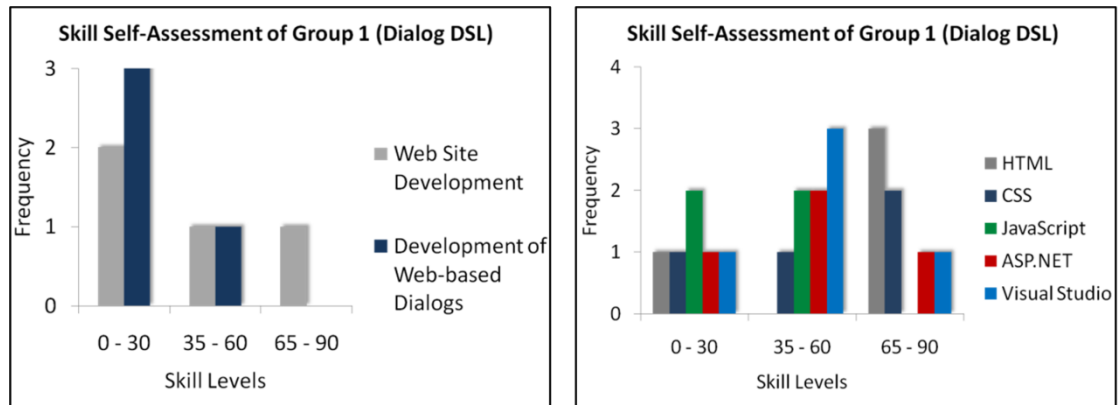


Figure 8-10: Distribution of Skills in Group 1
(Based on Self-Assessment via Initial Skills & Experiences Questionnaire)

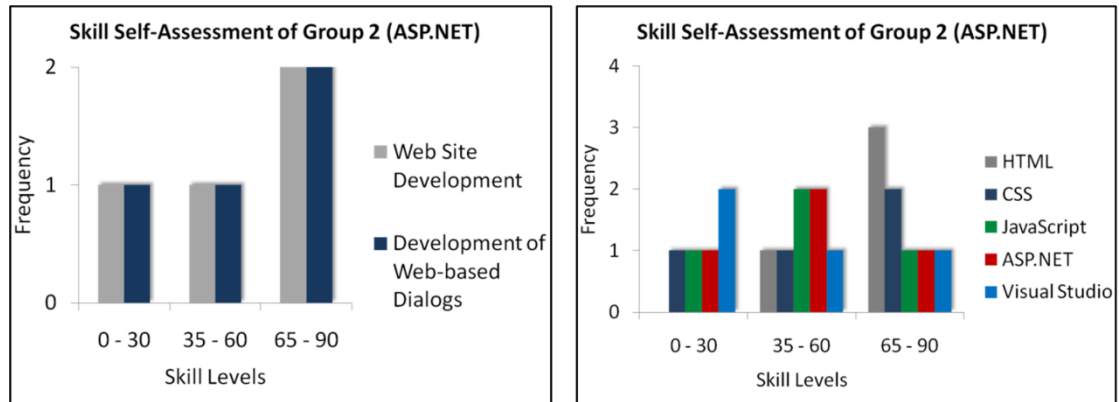


Figure 8-11: Distribution of Skills in Group 2
(Based on Self-Assessment via Initial Skills & Experiences Questionnaire)

Table 8-4 shows the measured dialog development times as well as derived descriptive statistical measures. Figure 8-12 illustrates the measured development times of both groups accordingly. Based on the calculated means for both approaches, the Dialog DSL approach turned out to be 2.6 times more efficient with respect to dialog development. Furthermore, the standard deviation among the subjects which used the Dialog DSL is lower which further underlines the achieved results. Taking into account the lower overall skill level of the Dialog DSL group, this presents an excellent result for the Dialog DSL approach.

Table 8-4: Measured Dialog Development Time and Derived Statistical Measures

Group / Treatment	Subject	Dialog Development Time (hh:mm m)						
		Measured Dev. Time	Median	Mean	Variance	Standard Deviation	Variation Coefficient	Range
1 (Dialog DSL)	2	00:47	00:37:30	00:39:15	29,58	5,44	13,86%	00:12
	5	00:35						
	7	00:36						
	8	00:39						
2 (ASP.NET)	1	01:47	01:42:00	01:41:15	45,58	6,75	6,67%	00:13
	3	01:47						
	4	01:37						
	6	01:34						

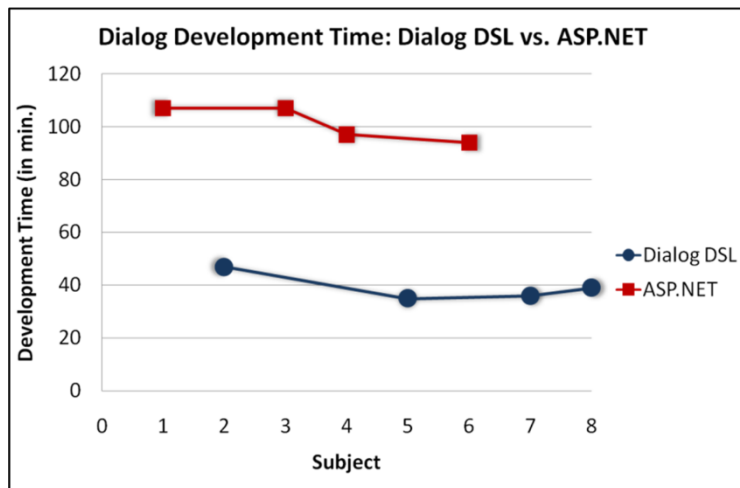


Figure 8-12: Dialog Development Times using the Dialog DSL Approach vs. ASP.NET

Table 8-5 shows the measured times for adopting changes to the developed dialog as well as derived descriptive statistical measures. Figure 8-13 illustrates the measured times for all subjects of both groups accordingly. Based on the calculated medians for both approaches, the Dialog DSL approach turned out to be 2.4 times more efficient with respect to change adoption. Furthermore, the standard deviation among the subjects which used the Dialog DSL is almost negligible which emphasizes the Dialog DSL’s superiority. Considering the lower overall skill level of the Dialog DSL group, this also presents an excellent result in favor of the Dialog DSL approach.

Table 8-5: Measured Change Adoption Time and Derived Statistical Measures

Group / Treatment	Subject	Change Adoption Time (hh:mm m)						
		Measured Change Time	Median	Mean	Variance	Standard Deviation	Variation Coefficient	Range
1 (Dialog DSL)	2	00:06	00:05:00	00:05:15	0,25	0,5	9,52%	00:01
	5	00:05						
	7	00:05						
	8	00:05						
2 (ASP.NET)	1	00:13	00:12:00	00:14:30	51,67	7,19	49,57%	00:14
	3	00:25						
	4	00:09						
	6	00:11						

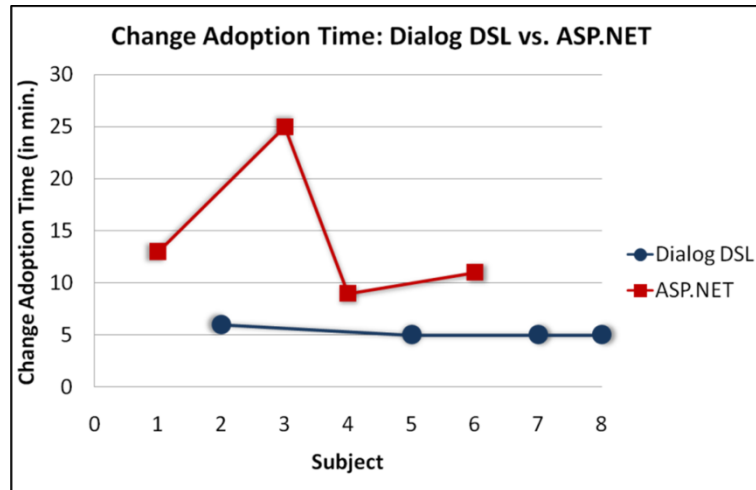


Figure 8-13: Change Adoption Time using the Dialog DSL Approach vs. ASP.NET

During the analysis of the developed dialogs, minor errors in the participant's deliverables were identified. In order to include these errors into the evaluation, compensation times for the various error types were defined and added to the actual measured times (Chouchane 2009). The ratio between the average error rate of the ASP.NET and the Dialog DSL group evaluates to 3.26. Thus, the Dialog DSL showed also a positive influence on the error rate in the experiment. Based thereupon, regarding hypothesis H_3 , the effective experiment time was calculated as sum of the development time, the change adoption time and the error compensation times. The resulting effective experiment times are shown in Table 8-6.

To conclude this section, the hypothesis H_3 shall be exemplarily tested using a *t-test* (Montgomery 1997) which compares the means μ of the two samples under the assumption that both originate from normal distributions with similar variances. Even though these assumptions are rarely satisfied in practice, the *t-test* is considered still robust (Briand, Emam and Morasca 1996). The *t-test* is conducted as follows.

1. Calculate t_0

$$t_0 = \frac{\bar{x} - \bar{y}}{S_p \sqrt{\frac{1}{n} + \frac{1}{m}}}$$

with

$$S_p = \sqrt{\frac{(n-1)S_x^2 + (m-1)S_y^2}{n+m-2}}$$

and \bar{x}, \bar{y} as means and S_x^2, S_y^2 as individual variances of the samples.

2. The null hypothesis $H_0: \mu_x \leq \mu_y$ is rejected in favor of the alternative hypothesis $H_1: \mu_x > \mu_y$ if

$$t_0 > t_{\alpha, n+m-2}$$

with $t_{\alpha, n+m-2}$ corresponding to the upper α percentage point of the t distribution with $n + m, -2$ degrees of freedom. In this regard, α corresponds to the level of statistical significance and $1 - \alpha$ accordingly to the test confidence. For the following calculation, $\alpha = 5\%$ was used. The distribution is tabulated for example in (Montgomery 1997).

Regarding the performed experiment, the alternative hypothesis $H_{3,1}: Eff(DialogDSL) > Eff(ASP.NET)$ introduced above can be rewritten to $H'_{3,1}: \mu_{eff,ASP.NET} > \mu_{eff,DialogDSL}$. Thus, a t-test was conducted to see if the null hypothesis $H'_{3,0}: \mu_{eff,ASP.NET} \leq \mu_{eff,DialogDSL}$ can be rejected in favor of $H'_{3,1}$.

Table 8-6 illustrates the calculation of the t-test based on the effective experiment times which include the dialog development time, change adoption time and error compensation time. According to its result, i.e. $t_0 = 8,90 > 2,447 = t_{\alpha, n+m-2}$, the null hypothesis $H'_{3,0}$ can be rejected in favor of $H'_{3,1}$ with a confidence of 95%. Thus, the Dialog DSL's superior efficiency was successfully confirmed.

Table 8-6: Effective Experiment Times and Calculation of the T-Test

Hypothesis Test for H_3 : Effective Time (Development, Change Adoption, Error Compensation)								
Samples	ASP: X (n = 4)				Dialog DSL: Y (m = 4)			
Effective Experiment Time (in min.)	124,7	138,5	106,7	108	55,5	40	41	44
Mean μ	119,46				45,13			
Degrees of Freedom (n+m-2)	6							
Variance $S^2_{x/y}$	228,19				50,73			
t-Test: t_0	8,90							
$t_{\alpha/2,6}$ with $\alpha = 5\%$	2,447							

8.3.2 Survey-based Evaluation of Stakeholder Adequacy

The Dialog DSL approach pursues the goal of enabling stakeholders to autonomously understand, validate, modify and even create Web-based dialogs or their corresponding models respectively. Particularly stakeholders without IT backgrounds and having sparse or no IT skills at all shall be efficiently addressed and involved.

The degree of achievement of these goals, summarized under the term *stakeholder adequacy*, was evaluated based on an empirical survey focusing on the Dialog DSL's modeling notation. The study was designed according to the various types of stakeholder activities and specified in form of a Goal/Question/Metric plan which is depicted in Table 8-7.

Table 8-7: GQM Plan for the Evaluation of the Dialog DSL's Stakeholder Adequacy

Goal 2	Empirical evaluation of the Dialog DSL modeling notation's stakeholder adequacy
Question Q2.1	How adequate is the Dialog DSL's modeling notation for the autonomous <i>creation</i> of dialog models by stakeholders?
Metric M2.1.1	Corresponding paper-based exercises with objective, score-based rating.
Metric M2.1.2	Stakeholder Adequacy Scale (SAS) enabling a subjective rating by participants.
Question Q2.2	How adequate is the Dialog DSL's modeling notation for the autonomous <i>incorporation of changes</i> into existing dialog models by stakeholders?
Metric M2.2.1	Corresponding paper-based exercises with objective, score-based rating.
Metric M2.2.2	Stakeholder Adequacy Scale (SAS) enabling a subjective rating by participants.
Question Q2.3	How adequate is the Dialog DSL's modeling notation for the autonomous <i>validation</i> of dialog models by stakeholders?
Metric M2.3.1	Corresponding paper-based exercises with objective, score-based rating.
Metric M2.3.2	Stakeholder Adequacy Scale (SAS) enabling a subjective rating by participants.

Each question is measured both on an objective and subjective scale. Regarding the former, the survey contained specific exercises addressing the following areas:

- Verifying statements about a given dialog model
- Incorporating given changes into a given dialog model
- Creating a dialog model according to a textual specification

In order to suppress possible effects originating from the Dialog DSL's Web-based editor, the survey was designed and conducted purely paper-based, thus focusing exclusively on the modeling notation itself.

With the purpose of obtaining also a subjective rating of the Dialog DSL's modeling notation by the survey participants after having completed the exercises, a so-called *Stakeholder Adequacy Scale (SAS)* was developed and is depicted in Table 8-8. It is based on the ideas of the System Usability Scale (SUS) (Brooke 1996; Tullis and Albert 2008) which is widely used for the subjective evaluation of electronic office systems. The calculation of the SAS score based on the ratings of a participant is conducted as follows: The ratings of questions 1, 3, 5, 7 and 9 are assigned with the score $5-s$ where s is the rating assigned by the participant. Accordingly, the remaining questions 2, 4, 6, 8 and 10 are assigned with the score $s-1$. Each score is multiplied by 2.5 so that the overall SAS score lies between 0 and 100.

Table 8-8: Stakeholder Adequacy Scale (SAS) for the Subjective Rating of the Dialog DSL's Modeling Notation by Survey Participants (translated from German)

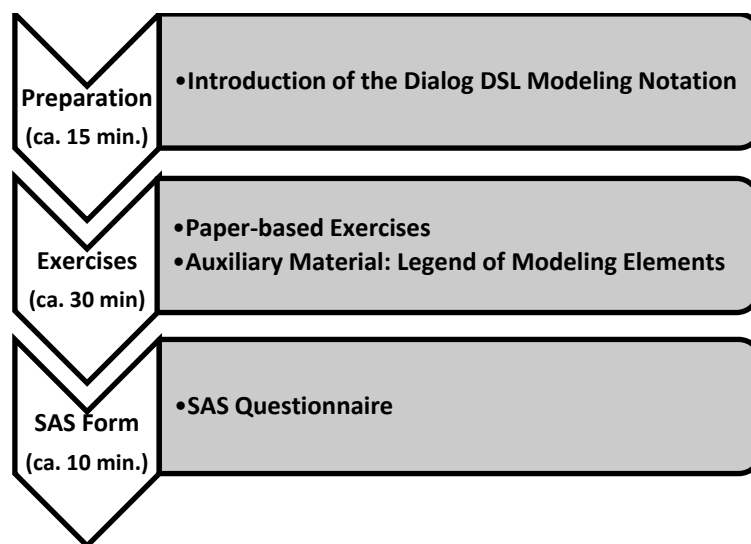
1. I have understood all notation elements.	<p style="text-align: center;"><i>Strongly agree</i> <i>Strongly disagree</i></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 20%; height: 20px;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> </table>						1	2	3	4	5
1	2	3	4	5							
2. I consider the notation elements to be unnecessary complex.	<p style="text-align: center;"><i>Strongly agree</i> <i>Strongly disagree</i></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 20%; height: 20px;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> </table>						1	2	3	4	5
1	2	3	4	5							
3. I found it easy to employ the notation elements for solving the exercises.	<p style="text-align: center;"><i>Strongly agree</i> <i>Strongly disagree</i></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 20%; height: 20px;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> </table>						1	2	3	4	5
1	2	3	4	5							
4. I was not able to complete the exercises without frequent questions and support by an expert.	<p style="text-align: center;"><i>Strongly agree</i> <i>Strongly disagree</i></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 20%; height: 20px;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> </table>						1	2	3	4	5
1	2	3	4	5							
5. I consider the various notation elements reasonable and necessary.	<p style="text-align: center;"><i>Strongly agree</i> <i>Strongly disagree</i></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 20%; height: 20px;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> </table>						1	2	3	4	5
1	2	3	4	5							
6. In my opinion, the various notation elements were difficult to distinguish from each other.	<p style="text-align: center;"><i>Strongly agree</i> <i>Strongly disagree</i></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 20%; height: 20px;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> </table>						1	2	3	4	5
1	2	3	4	5							
7. I consider it easy to remember and employ the various notation elements, even without a legend.	<p style="text-align: center;"><i>Strongly agree</i> <i>Strongly disagree</i></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 20%; height: 20px;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> </table>						1	2	3	4	5
1	2	3	4	5							
8. The modeling notation's successful usage requires a lot of previous knowledge.	<p style="text-align: center;"><i>Strongly agree</i> <i>Strongly disagree</i></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 20%; height: 20px;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> </table>						1	2	3	4	5
1	2	3	4	5							
9. I felt very confident in employing the various notation elements.	<p style="text-align: center;"><i>Strongly agree</i> <i>Strongly disagree</i></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 20%; height: 20px;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> </table>						1	2	3	4	5
1	2	3	4	5							
10. I experienced difficulties in using the various notation elements.	<p style="text-align: center;"><i>Strongly agree</i> <i>Strongly disagree</i></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 20%; height: 20px;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> <td style="width: 20%;"></td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> </table>						1	2	3	4	5
1	2	3	4	5							

With respect to the survey's validity, the involvement of subjects with heterogeneous, preferably non-IT-related backgrounds presented a key factor. A set of eight subjects adequately representing this requirement could be won for the survey. Table 8-9 shows their occupations, educational backgrounds and ages. Only one participant had few software development skills in PHP and no participant at all had ever developed a Web site or Web-based dialog. Thus, they formed a well-suited subject population for the survey.

Table 8-9: Survey Participants, their Occupation or Educational Background and Age

Subject	Occupation / Educational Background	Age
1	Undergraduate student of business administration	21
2	Trained retail saleswoman	25
3	Management consultant, diploma in business administration	26
4	Teacher	28
5	Hotline operator, no apprenticeship	46
6	Student of pedagogy with major in early childhood studies	22
7	Graduate student of electrical engineering	26
8	Graduate student of computer science	29

The survey execution process consisted of three parts (cf. Figure 8-14). First, a short introduction of the Dialog DSL's modeling notation was given. Subsequently, the participants worked on five paper-based exercises. The first three exercises addressed the aspect of model creation, the fourth exercise dealt with the incorporation of changes into an existing model and the fifth exercise covered the aspect of model validation. The participants were provided with a one-page legend of the Dialog DSL's modeling elements. No participant needed more than 30 minutes for accomplishing the exercises. In the third step, the participants filled out the Stakeholder Adequacy Scale (SAS) form described above. The complete survey material can be found in (Chouchane 2009).

**Figure 8-14:** Survey Process

After the survey, the performances of the eight participants in the exercises as well as their subjective ratings stated in the SAS forms were analyzed. Table 8-10 summarizes the measured results including descriptive statistical measures. For each subject and exercise type, the achieved absolute score and relative success rate are given. Furthermore, the table shows each participant's combined overall score and

success rate. Besides these objective measures, the table also indicates each participant’s rating in the SAS form which represents her subjective perception of the Dialog DSL’s modeling notation’s adequacy.

Table 8-10: Survey Results including Exercise Performances and SAS Rating

Subject	Type of Exercise						Overall Score (max. 300)	Overall Succ. Rate (%)	SAS (%)
	Create Model (E1-E3)		Incorporate Changes (E4)		Validate Model (E5)				
	Total Score	Success Rate (%)	Total Score	Success Rate (%)	Total Score	Success Rate (%)			
1	160	100	40	100	100	100	300	100,00	100,00
2	120	75	40	100	80	80	240	80,00	85,00
3	155	96,9	40	100	90	90	285	95,00	95,00
4	135	84,4	40	100	80	80	255	85,00	82,50
5	140	87,5	40	100	100	100	280	93,33	92,50
6	105	65,6	30	75	100	100	235	78,33	77,50
7	125	78,1	35	87,5	100	100	260	86,67	82,50
8	137	85,6	40	100	80	80	257	85,67	90,00
Mean	134,63	84,14	38,13	95,31	91,25	91,25	264,00	88,00	88,13
Median	136,00	85,00	40,00	100,00	95,00	95,00	258,50	86,17	87,50
Variance	325,41	127,37	13,84	86,50	98,21	98,21	508,00	56,44	56,70
St. Deviation	18,04	11,29	3,72	9,30	9,91	9,91	22,54	7,51	7,53
Var. Coeff. (%)	13,40	13,41	9,76	9,76	10,86	10,86	8,54	8,54	8,54
Range	55,00	34,40	10,00	25,00	20,00	20,00	65,00	21,67	22,50

Figure 8-15 illustrates the average success rates for the various task types. The success rates lie over 80 % for all task types which can be considered an excellent result. While the incorporation of changes was the most successful task type (95.31 %), the exercises concerning the autonomous creation of models were still in 84.14 % of all cases accomplished correctly. Across all task types, the participants reached a combined average success rate of 88 %. Taking into account the fact that the participants had no previous knowledge of the Dialog DSL and of Web development in general, the obtained results are even more appealing.

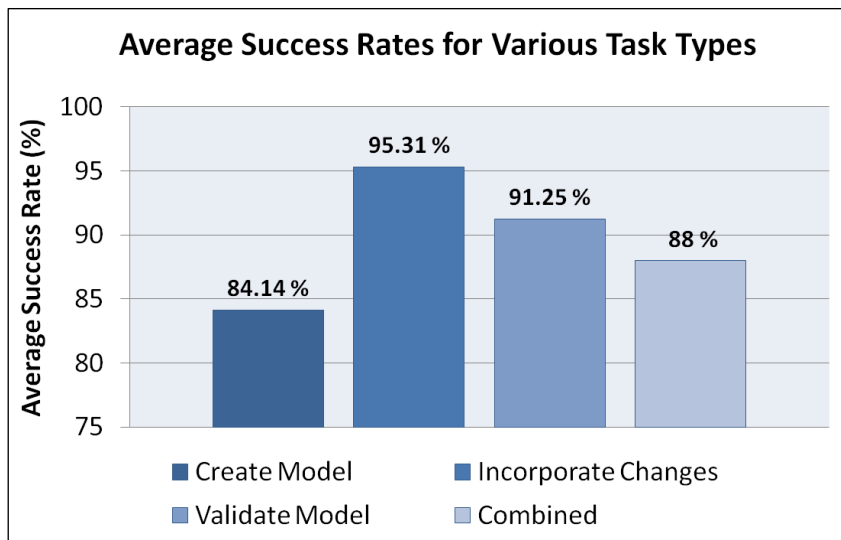


Figure 8-15: Average Success Rates for Various Task Types

The subjective ratings based on the Stakeholder Adequacy Scale (SAS) which was performed by the participants after the experiment are illustrated in Figure 8-16.

Again, the obtained results are overall very positive. The eight participants rated the Dialog DSL modeling notation's adequacy over 75 % which results in an average rating of 88.13 %.

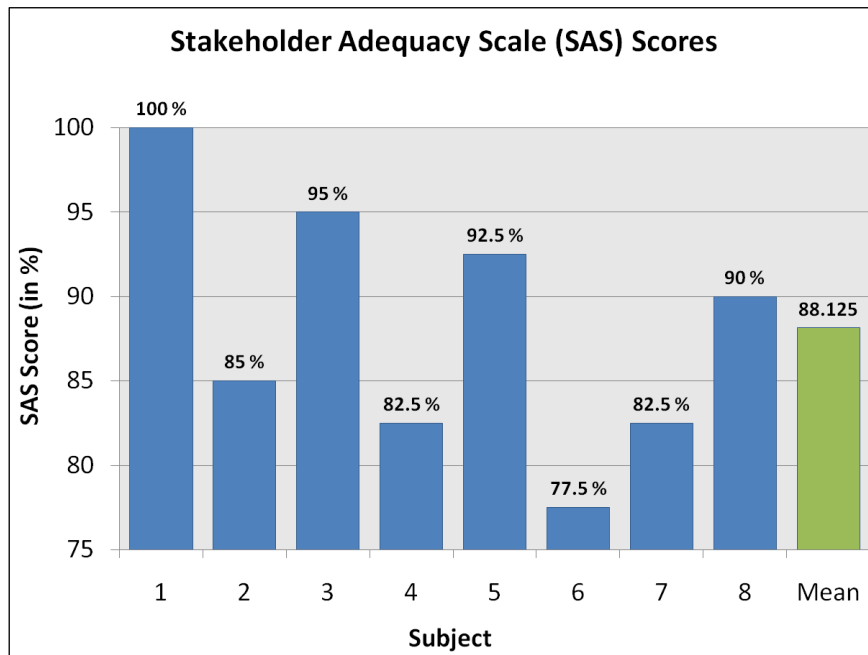


Figure 8-16: Stakeholder Adequacy Scale (SAS) Scores Awarded by the Participants

In summary, the Dialog DSL's modeling notation's excellent adequacy for enabling stakeholders to autonomously understand, validate, modify and even create dialog models could be successfully confirmed in the conducted survey. This was measured and confirmed both on an objective and subjective scale. The fact that the involved participants were totally inexperienced in Web and dialog development and had, with one exception, completely non-IT-related educational backgrounds fosters the validity of the obtained results. The observed positive trend could be further confirmed by additional empirical evaluations with more subjects and diverse dialog types used in the exercises.

9 Conclusion and Outlook

At the beginning of this thesis, the following research question was posed: *How can workflow-based Web applications be constructed in close collaboration with stakeholders in an efficient and effective way?*

In order to clarify this question, a detailed analysis of the considered problem domain was conducted. First of all, the continuous and strong involvement of stakeholders throughout the development process was introduced as a cross-cutting key requirement. While it was identified as a crucial success factor by numerous empirical studies, it is still not sufficiently addressed by existing scientific and commercial approaches. Thereafter, the particular characteristics of workflow-based Web applications were analyzed and key requirements concerning their efficient and effective construction elaborated. Thus, technical and methodological requirements as well as key challenges for effective stakeholder collaboration were identified. Subsequently, the characteristics of Web-based dialogs as a core pillar for the Web-based processing of workflow activities were examined. Based thereupon, crucial technological, methodological and stakeholder-oriented requirements an adequate development approach should address were identified. The preceding analysis repeatedly highlighted effective reuse as an important factor for the efficient construction of workflow-based Web applications and Web-based dialogs. Consequently, a sound elaboration of challenges and corresponding requirements for effective reuse in the Web Engineering domain followed.

Based on the resulting requirements catalog, a systematic in-depth analysis of the current state of the art including representative scientific and commercial approaches was performed. The analysis was structured along the dimensions workflow, dialog, and reuse and arrived at the conclusion that existing solutions do not achieve a sufficient fulfillment of the identified requirements. It was found that the following major problem areas still hinder an efficient and effective construction of workflow-based Web applications with stakeholders:

- *No holistic consideration of workflow and user interface aspects*
- *Insufficient support for advanced Web-based dialogs*
- *Constricted, proprietary reuse approaches*
- *Heavy-weight, inflexible development methodologies*
- *Restrictive developer-centricity*

Against this background, this thesis introduced novel models, systems and methodologies which explicitly address the identified requirements and open challenges. The contributions are structured in four core pillars and their evaluation:

Web Engineering DSL Framework: In view of the heavy-weight, monolithic, and developer-centric modeling approaches in the Web Engineering discipline, the Web Engineering DSL Framework presents a novel alternative. It establishes the conceptual foundation for continuously and intensely involving stakeholders throughout the development process by enabling them to autonomously validate, modify, and even specify parts of the solution. The framework suggests using a multitude of highly-focused Domain-Specific Languages (DSLs) for the various aspects of a Web application. The specification of a DSL allows for providing various modeling notations and corresponding editors, each of them tailored to the characteristics of an individual stakeholder audience and process stage. The resulting DSL programs serve as instrumentation for a DSL-specific software component which executes them by adapting its behavior accordingly. In conclusion, Web applications can be constructed in an evolutionary way by composing these DSL components and configuring them with DSL programs in form of stakeholder-tailored models.

Workflow DSL: Designed in accordance with the Web Engineering DSL Framework, the Workflow DSL enables the stakeholder-oriented and fully model-based development of workflow-based Web applications. The Workflow DSL bridges the gap between existing workflow execution platforms and the need for Web-based user interfaces for the efficient and effective processing of human tasks. It was shown how the XML Process Definition Language (XPDL), a widely-adopted workflow specification standard, can be systematically extended towards capturing the Web-based realization of workflow activities. To this end, a catalog of highly reusable Activity Building Blocks (ABBs) addressing the concerns dialog-based user interaction, data presentation and Web service communication was introduced. For each ABB, the minimal required configuration set in order to specify the desired behavior was elaborated and served as conceptual foundation for the metamodel-based extension of the XPDL standard. The resulting formalized schema of the Workflow DSL forms a novel, standard-based foundation for the holistic specification of both workflow execution and Web-based user interface aspects.

The Workflow DSL fosters the effective collaboration with stakeholders by allowing them to use standardized modeling notations and tools according to their individual skills and preferences. Thereby, technical complexity is hidden as far as possible and the modeling focus shifted towards the business process' structure. In order to enable such a cross-notational modeling on a single shared Workflow DSL program and thus achieving a novel degree of model continuity and integrity, the existing heterogeneity of business process modeling standards has to be overcome. Facing this so-far unsolved challenge, a novel model transformation framework striking a new path by introducing the Core Elements Set (CES) concept was presented. The CES defines a set of common business process and workflow concepts which abstracts from individual notations and languages. Thus, the CES establishes the conceptual basis for achieving semantic congruence between heterogeneous business process and workflow modeling languages. Although the CES cannot provide full coverage for all theoretical possible modeling constructs, empirical

evaluations showed that it achieves sufficient coverage for the great majority of scenarios occurring in practice. Based on the CES, a systematically and non-invasively extensible framework for lossless, bilateral model transformations was presented. The successful realization of the vision of real cross-notational modeling was exemplified by four modeling notations, their corresponding standardized serialization formats and supporting tools: The Business Process Modeling Notation (BPMN), UML 2.0 Activity Diagrams, Petri Nets and a custom table-based notation for early requirements engineering activities. Besides these horizontal transformations, also a vertical transformation to an executable workflow language was presented.

The Workflow DSL's technical support platform realizes the execution of model transformations and the fully-automated construction of workflow-based Web applications according to a given Workflow DSL program. Its service-oriented design establishes a sound foundation for federative scenarios and multimodal participation. The automated construction process and the consistent propagation of changes as well as the DSL-supported detailed design of the Web-based user interfaces at runtime foster an agile and evolutionary development process.

Dialog DSL: Complex but nonetheless effective dialog-based user interaction forms a core pillar for the Web-based processing of workflow activities. To this end, the Dialog DSL as a novel, fully model-based approach for the efficient and usability-oriented construction of advanced Web-based dialogs was presented. Simplicity formed a key principle in the DSL's design in order to enable stakeholders to autonomously validate, modify and create dialogs or their models respectively.

Furthermore, the Dialog DSL inherently focuses on usability aspects and related best practices as core features of advanced dialogs and facilitates their efficient model-based incorporation. The strong focus on usability already at design time as well as its excellent adequacy for stakeholders presents a significant advancement of the current state of the art. While the great majority of today's solutions still pursue a paper-like, predominantly technically- and appearance-oriented design approach, the Dialog DSL shifts the focus to usability and particularly dynamic behavior. Thus, the potentials of Web-based dialogs are effectively utilized and cognitive overload avoided.

Due to the automated dialog generation facilities as well as the rapid, fully model-based roundtrip engineering supported by a Web-based editor, the Dialog DSL enables an agile and evolutionary development process. Consequently, the Dialog DSL achieves significant efficiency gains compared to existing approaches. Adequate model transformations accomplish the client-specific adaptation and rendering of dialog models according to the W3C XForms standard and allow for the flexible incorporation of additional markup formats.

Web Engineering Reuse Sphere: Efficient and effective reuse across the diversity of existing Web Engineering methodologies combined with strong stakeholder involvement present the main contributions of the Web Engineering Reuse Sphere. It uniquely considers stakeholder characteristics and skills as key factors for reuse effectiveness, i.e. the capability to understand, evaluate and subsequently modify and use reusable artifacts. The introduced Web Engineering Reuse Ontology

establishes a semantic foundation for homogenizing the variety of heterogeneous Web Engineering methodologies and artifacts. Therefore, it provides well-defined extension points and was formalized based on Semantic Web standards. The technical integration of existing artifact repositories and client applications is guided by a supplemental architectural reference framework. By integrating also local ad-hoc repositories, both planned and spontaneous reuse scenarios are supported which in turn improves the coordination of hitherto unrecognized redundant development efforts.

Due to the ontology-based registry, the Web Engineering Reuse Sphere provides novel, cross-methodological search strategies which include stakeholder skills as an integral search facet. Thus, stakeholders are enabled to efficiently find adequate resolution strategies and artifacts for a given task type and in accordance with their individual skills and knowledge. In view of current consolidation activities towards model interoperability in the Web Engineering discipline, the Web Engineering Reuse Sphere forms a valuable contribution as enabler for real cross-methodological reuse. Beyond that, it also facilitates the DSL-based Web Engineering process by assisting stakeholders in finding adequate DSLs, modeling notations, software and related artifacts.

Evaluation: This thesis presented novel solutions for the efficient and effective construction of workflow-based Web applications which successfully address the identified requirements and hitherto unsolved challenges. Comprehensive technical implementations of the presented approaches allowed for their practical evaluation in various scenarios. Thus, their applicability and significant benefits for developing real-world applications could be observed. Furthermore, the achieved results were published in nineteen publications at international workshops, conferences and journals and intensely discussed with researchers from the Web Engineering community and adjacent research areas.

In addition, formal empirical evaluations of the presented core concepts and methodologies were conducted. Regarding the Workflow DSL, the applicability of the novel Core Elements Set (CES) concept and the Application Building Blocks (ABB) catalog were examined based on a comprehensive set of real-world business process models. The study arrived at the conclusion that the CES achieves 97.5 % coverage of the occurring modeling constructs and that 100% of the workflow activities could be realized with the ABB catalog. This confirms the Workflow DSL's excellent applicability for the fully model-driven and cross-notational construction of workflow-based Web applications based on real-world business process models.

The Dialog DSL approach was empirically examined concerning its development efficiency and stakeholder adequacy. A formal experiment substantiated that the Dialog DSL achieves significant efficiency gains by a factor of 2.6 compared to existing approaches. Furthermore, a survey-based evaluation confirmed its modeling notation's adequacy for stakeholders with heterogeneous, none-IT-related educational backgrounds. The participating stakeholders achieved an objective average success rate of 88% with respect to creating, modifying and validating dialog models. Their subjective perception of the modeling notation's stakeholder adequacy averaged 88.13 %. In summary, the Dialog DSL approach forms a

fundamental contribution to the current state of the art in terms of efficiency and stakeholder involvement.

The presented models, systems and methodologies establish also a sound basis for future work. While the Workflow DSL provides sufficient support for basic federative scenarios, more advanced settings could require additional considerations, e.g. regarding distributed workflow transaction management (Wenzel 2009; Wenzel, Freudenstein and Nussbaumer 2009). Furthermore, as the empirical evaluation showed, domain-specific specializations of the introduced ABBs can further improve the approach's efficiency and should thus be further examined. To this end, the Web Engineering DSL Framework provides an adequate conceptual foundation for their specification and evolution. The service-oriented and distributed nature of workflow-based Web applications drives the need for adequate approaches addressing their operation at a consistent level of quality. In this context, particular emphasis has to be placed on capturing and evaluating the complex interdependencies between relevant services, systems and stakeholders throughout their complete lifecycle (Majer, Nussbaumer and Gaedke 2008; Majer, Nussbaumer and Freudenstein 2009).

The Dialog DSL considers usability as a core feature of advanced Web-based dialogs, particularly in the context of workflow-based Web applications. In addition to its inherent accentuation of usability best practices and the facilitation of their model-based realization, a proactive usability validation at design time would present an ideal complement. Therefore, usability best practices could be conceptualized as rules based on the Dialog DSL's formalized schema and continuously evaluated at design time. Furthermore, measuring the success of the Dialog DSL's focus on usability in term of the approach's influence on the resulting dialog's usability could provide interesting insights for its continuous advancement.

The Web Engineering Reuse Ontology achieves a homogenization of Web Engineering methodologies not only regarding solely reuse-related aspects, but also with regard to their respective artifacts, modeling techniques, tools, methodologies and knowledge. Thus, it presents also a valuable input for consolidation activities in the Web Engineering discipline like the MDWEnet initiative. Thus, a continuous alignment would be desirable in order to enable mutual benefits and particularly to accomplish the long-term vision of cross-methodological interoperability and reuse.

List of Figures

Figure 1-1: Structure of the Thesis	9
Figure 2-1: Simplified Excerpt from the “Business Trip” Business Process	15
Figure 2-2: Various Business Process Modeling Notations and Tools	19
Figure 3-1: Overview of Steps, Tools and Results of the WebML Methodology for Lightweight Web-enabled Workflows. Taken from: (Brambilla 2006).....	31
Figure 3-2: Overview of IBM Business Process Management Products. Taken from: (IBM Corp. 2006)	34
Figure 3-3: Overview of the Development Process based on the IBM Platform. Taken from: (Brown, Johnston, Larsen et al. 2005).....	36
Figure 3-4: Abstract Widget Ontology of OOHDM/SHDM. Taken from: (Moura and Schwabe 2004)..	37
Figure 3-5: Screenshot of IBM Lotus Forms Designer	41
Figure 4-1: Overview of the Evolutionary and Reuse-Oriented DSL-based Web Engineering Approach	53
Figure 4-2: Projection of the Domain-Specific Model (DSM) onto the Domain Interaction Model (DIM) Taxonomy	54
Figure 4-3: Complete Overview of the Presented Contributions for the Stakeholder-Oriented Construction of Workflow-based Web Applications	58
Figure 5-1: Overview of the Evolutionary Workflow DSL Process Model	64
Figure 5-2: Overview of the XPDL Process Definition Metamodel. Taken from: (Shapiro, Marin, Brunt et al. 2005).....	67
Figure 5-3: Schematized Overview of Relevant XPDL Application Types and the Web-Specific Extensions Introduced by the Workflow DSL	73
Figure 5-4: XPDL Type Declaration and Data Field Specification within a Workflow DSL Program	74
Figure 5-5: Workflow Activity Definition within a Workflow DSL Program.....	75
Figure 5-6: Application Definition for a Web-based Expense Report Dialog within a Workflow DSL Program	76
Figure 5-7: Multi-Notational Modeling of a Shared Workflow DSL Program.....	77
Figure 5-8: Initial Draft of the Business Process using the Simple Sequence Only (SSO) DIM and Microsoft Word 2007	78
Figure 5-9: Transformed BPMN Representation of Workflow DSL Program	80

Figure 5-10: Added Process Section in the BPMN DIM Representation and the Corresponding Workflow DSL Program's XML Representation	81
Figure 5-11: UML 2.0 Profile for the Workflow DSL's UML 2.0 Activity DIM	83
Figure 5-12: Extending the Business Process using the UML 2.0 Activity DIM and IBM Rational Software Architect	84
Figure 5-13: Excerpts from the XMI-based Serialization corresponding to the Business Process Section Added using the UML 2.0 Activity DIM	85
Figure 5-14: Workflow Modeling using the Petri Nets DIM and INCOME 2010	88
Figure 5-15: PNML Excerpt resulting from the Application Configuration for the CreateExpenseReport Activity	88
Figure 5-16: Workflow Modeling: Role Assignment in INCOME 2010	89
Figure 5-17: Role Assignment in PNML	89
Figure 5-18: Workflow Modeling: Data Object Specification in INCOME 2010	90
Figure 5-19: Data Object Specification in PNML	90
Figure 5-20: Model Transformation Strategies between various DIMs: (a) Peer-to-Peer, (b) Ring, (c) Strategy based on an Intermediate Schema	92
Figure 5-21: Overview of the Workflow DSL's Model Transformations following the Intermediate Schema-based Model Transformation Strategy	93
Figure 5-22: Transitive Closure of the Model Transformation Graph	94
Figure 5-23: CES-based Model-to-Model Transformation Strategy	96
Figure 5-24: The CES Concepts Activity (a) / Silent Activity (b) and Sequence (c)	100
Figure 5-25: The CES Concepts Start Node (a) and End Node (b)	100
Figure 5-26: The CES Concepts AND-Split (a) and AND-Join (b)	100
Figure 5-27: The CES Concepts XOR-Split (a) and XOR-Join (b)	101
Figure 5-28: The CES Concepts OR-Split and OR-Join as Composition of an AND-Split/-Join as well as multiple XOR-Split/-Join structures nested therein	101
Figure 5-29: The CES Concept Structured Loop as While-Do-Loop (a) and Do-While-Loop (b)	102
Figure 5-30: Pseudo Code of the Model Traversal and Transformation Algorithm	104
Figure 5-31: XSLT-based Implementation of the Stack Technique	104
Figure 5-32: Pseudo Code of the Layout Algorithm for the Petri Net DIM	105
Figure 5-33: Graph-Structured vs. Block-Structured Specification Styles	109
Figure 5-34: The Parallel Pattern in XPDL and XOML	110
Figure 5-35: The Decision Pattern in XPDL and XOML	110
Figure 5-36: Referencing Separately Defined Declarative Rule Conditions	111
Figure 5-37: The Structured Loop Pattern (variant: While-Do-Loop) in XPDL and XOML	111
Figure 5-38: XOML DependencyProperty for XPDL TypeDeclaration and DataField	112
Figure 5-39: The Technical Platform of the Workflow DSL	115
Figure 5-40: Excerpt of the Transform Web Service's Public Interface	116
Figure 5-41: Achieving Interchangeable Transformation Engines based on the Strategy Design Pattern	117
Figure 5-42: Interaction between Clients, the Transform Service and Various Engines	118

Figure 5-43: List of Available Transformations in the Transformation Manager	120
Figure 5-44: Public Interface of the Workflow Web Service	122
Figure 5-45: Assembly of Application Building Blocks According to the Application Specification in the Workflow DSL Program	123
Figure 5-46: Screenshot of the Workflow List in the Business Trip Example Scenario	124
Figure 5-47: Transitions between the Currently Visible Subordinated Components within the Workflow SBB's Inner Presentation Place	126
Figure 5-48: DIM-to-DSM Transformation via the Transform Manager	127
Figure 5-49: Configuring a Workflow SBB Instance in the WSLS Framework.....	128
Figure 5-50: Workflow Execution – The Create Expense Report Activity.....	129
Figure 5-51: Support for Cross-Organizational Web-Based Workflow Scenarios	130
Figure 6-1: Overview of the Evolutionary Dialog Engineering Methodology	137
Figure 6-2: Simplified Excerpt from the Dialog DSL's Domain-Specific Model.....	139
Figure 6-3: The Choice Interaction Structure as Petri Net Transition Template	141
Figure 6-4: The Sequence Interaction Structure as Petri Net Transition Template	142
Figure 6-5: Binding Interaction Elements via Corresponding User Control Symbols to Data Elements and Defining Semantic Groups	142
Figure 6-6: Pagination of a Dialog Partition via the Sequence Interaction Structure	144
Figure 6-7: Overview of the Dialog DSL's Technical Platform	145
Figure 6-8: Partitions & Transitions Design in the Web-Based DIM Editor	147
Figure 6-9: Usability-Oriented Appearance Design in the Web-Based DIM Editor	147
Figure 6-10: Initial Configuration of a Dialog SBB Instance	149
Figure 6-11: Rendered Web-based Travel Expense Report Dialog Incorporating Choice (1, 3) and Sequence (5) Interaction Structures.....	150
Figure 7-1: The Web Engineering Reuse Sphere	154
Figure 7-2: Simplified Overview of the Ontology	156
Figure 7-3: Ontology Excerpt with Instances for WebML and UWE.....	157
Figure 7-4: Ontology Concepts related to Knowledge and Stakeholders (Simplified Excerpt)	158
Figure 7-5: Ontology Concepts Related to Artifact, Methodology, Process and Product (Simplified Excerpt).....	159
Figure 7-6: Ontology Concepts Resolution Strategy, Modeling Technique & Software and their Instantiation for the WebML Methodology (Simplified Excerpt)	161
Figure 7-7: Ontology Concepts Resolution Strategy, Modeling Technique & Software and their Instantiation for the Web Engineering DSL Framework (Simplified Excerpt).....	162
Figure 7-8: Overview of an Artifact's Relations.....	164
Figure 7-9: The Reference Architecture Framework.....	166
Figure 7-10: The Reuse View in the WSLS Framework.....	167
Figure 7-11: SPARQL Query for Determining all Modeling and Software Skills related to the Task 'Design Business Process' across all Web Engineering Methodologies.....	169
Figure 7-12: The 'Search for Resolution Strategy' Wizard.....	170
Figure 7-13: The 'Search for Resolution Strategy' Wizard.....	171

Figure 7-14: The ‘Search for Existing Artifact’ Wizard.....	172
Figure 7-15: Search Results with Browsing and Filtering Facilities.....	173
Figure 8-1: Classification Methodology for CES Evaluation.....	178
Figure 8-2: CES Evaluation Result before Remodeling	179
Figure 8-3: Observed Frequency Distribution of Identified CES Concepts in the Evaluated Business Process Models.....	181
Figure 8-4: Identified Workflow Activity Types and their Coverage by the Workflow DSL’s Activity Building Block Catalog	183
Figure 8-5: The KIM iSOA.....	184
Figure 8-6: FSM-based Model of the ‘Course Registration’ Integration Scenario.....	185
Figure 8-7: The Technical UI Workflow Integration and Execution Framework	187
Figure 8-8: UI Workflow Modeling in Visual Studio 2005 (1: FSM, 2: Entry Actions) and its Execution by an UI Workflow WebPart Instance in the Students Portal (3).....	188
Figure 8-9: Experiment Processes for the Two Groups or Treatments Respectively	192
Figure 8-10: Distribution of Skills in Group 1 (Based on Self-Assessment via Initial Skills & Experiences Questionnaire).....	193
Figure 8-11: Distribution of Skills in Group 2 (Based on Self-Assessment via Initial Skills & Experiences Questionnaire).....	193
Figure 8-12: Dialog Development Times using the Dialog DSL Approach vs. ASP.NET	194
Figure 8-13: Change Adoption Time using the Dialog DSL Approach vs. ASP.NET	195
Figure 8-14: Survey Process	199
Figure 8-15: Average Success Rates for Various Task Types	200
Figure 8-16: Stakeholder Adequacy Scale (SAS) Scores Awarded by the Participants	201

List of Tables

Table 2-1: Table-based Elicitation of Tasks and Roles of a Business Process Excerpt.....	18
Table 3-1: Evaluation of State of the Art Approaches against the Presented Requirements Catalog – Dimensions Workflow and Dialog	48
Table 3-2: Evaluation of the State of the Art against the Presented Requirements Catalog – Dimension Reuse	48
Table 3-3: Legend of Rating Symbols	49
Table 3-4: Overview of the Requirements Catalog from Chapter 2.....	49
Table 5-1: The Core Elements Set (CES)	97
Table 5-2: Mapping of CES Concepts to XOML Elements.....	108
Table 5-3: Complete Overview of Mappings between DIMs and the DSM (Part 1/2)	113
Table 5-4: Complete Overview of Mappings between DIMs and the DSM (Part 2/2)	114
Table 6-1: Multi-Step Transformation of Dialog Models into Executable Markup	145
Table 8-1: Frequency of Workflow Concepts Before and After the Error Resolution by Remodeling	180
Table 8-2: GQM Plan for the Experimental Evaluation of the Dialog DSL’s Efficiency	190
Table 8-3: Performance-Based Subject Allocation into Two Balanced Groups.....	191
Table 8-4: Measured Dialog Development Time and Derived Statistical Measures	194
Table 8-5: Measured Change Adoption Time and Derived Statistical Measures	194
Table 8-6: Effective Experiment Times and Calculation of the T-Test.....	196
Table 8-7: GQM Plan for the Evaluation of the Dialog DSL’s Stakeholder Adequacy.....	197
Table 8-8: Stakeholder Adequacy Scale (SAS) for the Subjective Rating of the Dialog DSL’s Modeling Notation by Survey Participants (translated from German).....	198
Table 8-9: Survey Participants, their Occupation or Educational Background and Age.....	199
Table 8-10: Survey Results including Exercise Performances and SAS Rating.....	200

Bibliography

- Acerbis, R., A. Bongio, M. Brambilla and S. Butti (2007). *WebRatio 5: An Eclipse-based CASE Tool for Engineering Web Applications*. In Proceedings of 7th International Conference on Web Engineering (ICWE 2007). Como, ItalySpringer, Heidelberg, ISBN: 978-3-540-73596-0.
- Allerding, F. (2007). *Modellgetriebene Entwicklung dynamischer Benutzerschnittstellen im Web Engineering*. Diploma Thesis, Institute of Telematics, University of Karlsruhe (TH), Karlsruhe, 30.04.2007. 115 pages.
- Allerding, F., J. Buck, P. Freudenstein, T. Höllrigl, W. Juling, B. Keuter, B. Klosek, S. Link, F. Majer, A. Maurer, M. Nussbaumer, D. Ried and F. Schell (2008). *Integriertes Service-Portal zur Studienassistenz*. In Proceedings of INFORMATIK 2008, 38. Jahrestagung der Gesellschaft für Informatik. Munich, Germany, 08.-13.09.2008. ISBN: 978-3-88579-228-4.
- Alter, A. (2007). *50 Technologies: Where CIOs are Spending Their Money*, CIO Insight, 14 Februray 2007, from: <http://www.cioinsight.com/c/a/Research/50-Technologies-Where-CIOs-are-Spending-Their-Money/>.
- Arango, G. (1989). *Domain Analysis: From Art Form to Engineering Discipline*. ACM SIGSOFT Software Engineering Notes 14(3): 152-159. ISSN: 0163-5948
- Arsanjani, A. (2004). *Service-oriented Modeling and Architecture*. IBM developerWorks International Business Machines (IBM) Corporation. Retrieved 16.04.2009, from <http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/>.
- Basili, V. R., G. Caldiera and H. D. Rombach (1994). *Goal Question Metrics Paradigm*. In: Encyclopedia of Software Engineering. J. Marciniak (ed.), John Wiley and Sons. Vol. 1: 528-532.
- Batory, D., B. Lofaso and Y. Smaragdakis (1998). *JTS: Tools for Implementing Domain-Specific Languages*. In Proceedings of Fifth International Conference on Software Reuse. Victoria, British Columbia, Canada, 2-5 June 1998. IEEE Computer Society Press, ISBN: 978-0818683770.

- Baumeister, H., N. Koch and L. Mandel (1999). *Towards a UML Extension for Hypermedia Design*. In Proceedings of UML'99: The Unified Modeling Language - Beyond the Standard. Fort Collins, USA Springer Verlag
- BEA Systems (2007). *State of the Portal Market 2007: Portals and the Power of Participation*, White Paper. 42 pages, San Jose, USA, 25 May 2007, from: <http://www.oracle.com/technology/pub/articles/dev2arch/2008/03/state-of-portal-market.html>.
- Bechhofer, S., F. v. Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider and L. A. Stein (2004). *OWL Web Ontology Language Reference*, World Wide Web Consortium (W3C), W3C Recommendation. 10 February 2004, from: <http://www.w3.org/TR/owl-ref/>.
- Berglund, A., S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie and J. Siméon. (2007, 23 January 2007). *XML Path Language (XPath) 2.0*. W3C Recommendation, World Wide Web Consortium (W3C). Retrieved 19.03.2009, from <http://www.w3.org/TR/xpath20/>.
- Bieberstein, N., S. Bose, M. Fiammante, K. Jones and R. Shah (2006). *Service-Oriented Architecture (SOA) Compass: Business Value, Planning, and Enterprise Roadmap*, IBM Press. 232 pages.
- Blechar, M. J. (2007). *IBM's Federated Metadata Management Strategy*, Gartner, Inc., Research Report. 10 pages, Stamford, CT, USA, G00147616, 16 April 2007
- Boldyreff, C., D. Nutter and S. Rank (2002). *Active Artefact Management for Distributed Software Engineering*. In Proceedings of Proc. of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment. IEEE Computer Society
- Bos, B., T. Çelik, I. Hickson and H. W. Lie. (2007, 19 July 2007). *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. W3C Candidate Recommendation, World Wide Web Consortium (W3C). Retrieved 19.03.2009, from <http://www.w3.org/TR/CSS21/>.
- Böttger, M. (2008). *Ein ontologiebasiertes Wiederverwendungs-Rahmenwerk für das kollaborative Web Engineering*. Diploma Thesis, Institute of Telematics, University of Karlsruhe (TH), Karlsruhe, 30.04.2008. 132 pages.
- Boyer, J., T. Bray and M. Gordon (1998). *Extensible Forms Description Language (XFDL) 4.0*, World Wide Web Consortium (W3C), W3C Note. September 2, 1998, from: <http://www.w3.org/TR/NOTE-XFDL>.
- Boyer, J. M., M. Dubinko, J. Leigh L. Klotz, D. Landwehr, R. Merrick and T. V. Raman. (2007, 29 October 2007). *XForms 1.0 (Third Edition)*. W3C Recommendation, World Wide Web Consortium (W3C). from <http://www.w3.org/MarkUp/Forms/specs/XForms1.0.ThirdEdition/index-all.html>.
- Bozzon, A., S. Comai, P. Fraternali and G. T. Carughi (2006). *Conceptual Modeling and Code Generation for Rich Internet Applications*. In Proceedings of

- International Conference on Web Engineering 2006 (ICWE 2006). Menlo Park, USA
- Brambilla, M. (2006). *Generation of WebML Web Application Models from Business Process Specifications*. In Proceedings of International Conference on Web Engineering (ICWE) 2006. Menlo Park, California, USA
- Brambilla, M., S. Ceri, S. Comai, P. Fraternali and I. Manolescu (2003). *Specification and Design of Workflow-Driven Hypertexts*. Journal of Web Engineering (JWE) 1(2): 163-182. ISSN: 1540-9589
- Brambilla, M., S. Ceri, P. Fraternali and I. Manolescu (2006). *Process Modeling in Web Applications*. ACM Transactions on Software Engineering and Methodology (TOSEM) 15(4): 360 - 409. ISSN: 1049-331X
- Brambilla, M., S. Comai, P. Fraternali and M. Matera (2008). *Designing Web Applications with WebML and WebRatio*. In: Web Engineering - Modelling and Implementing Web Applications. G. Rossi, O. Pastor, D. Schwabe and L. Olsina (eds.). London, UK, Springer Verlag Ltd.: 221-261. ISBN: 978-1-84628-922-4
- Briand, L., K. E. Emam and S. Morasca (1996). *On the Application of Measurement Theory in Software Engineering*. Journal on Empirical Software Engineering 1(1): 61-88. ISSN: 1382-3256
- Brickley, D. and L. Miller. (2007). *FOAF Vocabulary Specification 0.91*. Retrieved 12.02.2008, from <http://xmlns.com/foaf/spec/>.
- Brooke, J. (1996). *SUS - A Quick and Dirty Usability Scale*. In: Usability Evaluation in Industry. P. W. Jordan, B. Thomas, B. A. Weerdmeester and A. L. McClelland (eds.). London, Taylor and Francis: 252. ISBN: 978-0748404605
- Brown, A. W., S. K. Johnston, G. Larsen and J. Palistrant (2005). *SOA Development Using the IBM Rational Software Development Platform: A Practical Guide*, International Business Machines (IBM) Corporation, White Paper. 36 pages, Somers, NY, September 2005, from: <ftp://ftp.software.ibm.com/software/rational/web/whitepapers/G507-0956-00.pdf>.
- Buck, J. (2007). *Modellgetriebene Entwicklung Workflow-basierter Web Anwendungen*. Diploma Thesis, Institute of Telematics, University of Karlsruhe (TH), Karlsruhe, 28 February 2007. 114 pages.
- Cantara, M., F. Biscotti and A. Raina (2007). *Forecast: Portal, Process and Middleware Software, Worldwide, 2006-2011*, Gartner, Inc., Research Report. Stanford, CT, USA, G00148362, 27 April 2007
- Carter, S. (2007). *The New Language of Business: SOA & Web 2.0*, IBM Press / Pearson. 299 pages. ISBN: 013195654X
- Ceri, S., P. Fraternali and A. Bongio (2000). *Web Modeling Language (WebML): A Modeling Language for Designing Web Sites*. In Proceedings of 9th International World Wide Web Conference (WWW). Amsterdam, Netherlands

- Ceri, S., P. Fraternali and M. Matera (2001). *WebML Application Frameworks: a Conceptual Tool for Enhancing Design Reuse*. In Proceedings of WWW10 Workshop Web Engineering. Hong Kong
- Charette, R. N. (2005). *Why Software Fails*. IEEE Spectrum 42(9): 42-49. ISSN: 0018-9235
- Chouchane, L. (2009). *Empirische Evaluation der Effektivität der Stakeholder-Einbindung durch den Dialog DSL-Ansatz*. Study Thesis, Institute of Telematics, University of Karlsruhe (TH), Karlsruhe, 27.03.2009. 146 pages.
- Christensen, E., F. Curbera, G. Meredith and S. Weerawarana (2001). *Web Services Description Language (WSDL) 1.1*.
- Clark, J. (1999). *Extensible Stylesheet Language Transformations (XSLT), Version 1.0*. W3C Recommendation, World Wide Web Consortium (W3C). Retrieved 20.03.2009, from <http://www.w3.org/TR/xslt>.
- Cockburn, A. (2006). *Agile Software Development*, Addison-Wesley Professional. ISBN: 978-0321482754
- Cowan, D. D. and C. J. P. Lucena (1995). *Abstract Data Views: An Interface Specification Concept to Enhance Design for Reuse*. Software Engineering, IEEE Transactions on 21(3): 229-243.
- Czarnecki, K. and U. W. Eisenecker (2000). *Chapter 3: Domain Engineering and Object-Oriented Analysis and Design*. In: Generative Programming. Methods, Tools and Applications: Methods, Techniques and Applications. Amsterdam, Addison-Wesley Longman. ISBN: 0201309777
- Czarnecki, K. and S. Helsen (2003). *Classification of Model Transformation Approaches*. In Proceedings of OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture. Anaheim, CA, USA, 27 October 2003.
- De Medeiros, A. P., D. Schwabe and B. Feijo (2005). *Kuaba Ontology : Design rationale representation and reuse in model-based designs*. In Proceedings of Proc. of 24th International Conference on Conceptual Modeling. Klagenfurt, Austria
- Dean, M. and M. Paolucci (2008). *Proceedings of "Semantic Web in Use" Track*. In: Proceedings of the Seventh International Semantic Web Conference (ISWC 2008). A. P. Sheth, S. Staab, M. Dean et al (eds.). Karlsruhe, Germany, Springer Verlag, Berlin / Heidelberg. ISBN: 978-3-540-88563-4
- Deshpande, Y., S. Murugesan, A. Ginige, S. Hansen, D. Schwabe, M. Gaedke and B. White (2002). *Web Engineering*. Journal of Web Engineering 1(1): 3-17.
- Deursen, A. v., P. Klint and J. Visser (2000). *Domain-Specific Languages: An Annotated Bibliography*. ACM SIGPLAN Notices 35(6): 26-36.
- Dublin Core Metadata Initiative. (2008). *Dublin Core Metadata Initiative RDF Schemas*. Retrieved 12.02.2008, from <http://dublincore.org/schemas/rdfs/>.

- Ecma International (2006). *Standard ECMA-376: Office Open XML File Formats - First Edition*, Ecma International, ECMA-376, December 2006, from: <http://www.ecma-international.org/publications/standards/Ecma-376.htm>.
- Erl, T. (2005). *Service-Oriented Architecture - Concepts, Technology, and Design*, Prentice Hall. 792 pages. ISBN: 978-0131858589
- Escalona, M. J. and N. Koch (2004). *Requirements Engineering for Web Applications – A Comparative Study*. Journal of Web Engineering 2(3): 193-212. ISSN: 1540-9589
- Fasbinder, M. (2007a, 24 October 2007). *Business Process Standards, Part 2: How the Standards are Used in WebSphere Products*. IBM developerWorks IBM Corporation. Retrieved 03.03.2009, 2009, from http://www.ibm.com/developerworks/websphere/library/techarticles/0710_fasbinder2/0710_fasbinder2.html.
- Fasbinder, M. (2007b, 05 December 2007). *What's new in WebSphere Integration Developer V6.1*. IBM developerWorks IBM Corporation. Retrieved 03.03.2009, 2009, from http://www.ibm.com/developerworks/websphere/library/techarticles/0712_fasbinder_wid/0712_fasbinder.html.
- Fasbinder, M. (2007c, 05 December 2007). *What's new in WebSphere Business Modeler V6.1*. IBM developerWorks IBM Corporation. Retrieved 03.03.2009, 2009, from http://www.ibm.com/developerworks/websphere/library/techarticles/0712_fasbinder/0712_fasbinder.html.
- Feigenbaum, L. and T. Heath. (2009). *International Semantic Web Conference 2009 - Semantic Web In Use Track - Call for Papers*. Semantic Web Science Association. Retrieved 09.03.2009, from http://iswc2009.semanticweb.org/wiki/index.php/ISWC_2009_Semantic_Web_In_Use_Track.
- Fensel, D. (2003). *The OntoWeb Ontology Homepage*. Retrieved 08.02.2008, from <http://www.ontoweb.org/Ontology/index.html>.
- Fialho, A. and D. Schwabe (2007). *Enriching Hypermedia Application Interfaces*. In Proceedings of 6th International Workshop on Web-Oriented Software Technologies (IWWOST'07). Como, Italy
- Fowler, M. (2005, 12.06.2005). *Language Workbenches: The Killer-App for Domain Specific Languages?*, from <http://www.martinfowler.com/articles/languageWorkbench.html>.
- Frakes, W. B. and T. P. Pole (1994). *An Empirical Study of Representation Methods for Reusable Software Components*. IEEE Transactions on Software Engineering 20(8): 617.
- Fransen, J. (2003). *Customizing the Microsoft Office 2003 Research Task Pane*. Retrieved 12.02.2008, from [http://msdn2.microsoft.com/en-us/library/aa159647\(office.11\).aspx](http://msdn2.microsoft.com/en-us/library/aa159647(office.11).aspx).

- Freeman, P. (1983). *Reusable Software Engineering: Concepts and research directions*. In Proceedings of The Workshop on Reusability in Programming. Newport, RI, USA
- Freudenstein, P., M. Boettger and M. Nussbaumer (2008). *Efficacious Reuse Support as Enabler for Cross-Methodological Web Engineering with Stakeholders*. In Proceedings of 8th International Conference on Web Engineering (ICWE2008). New York, USA, 14-18 July 2008. Institute of Electrical and Electronics Engineers (IEEE), ISBN: 978-0-7695-3261-5.
- Freudenstein, P., J. Buck, M. Nussbaumer and M. Gaedke (2007). *Model-driven Construction of Workflow-based Web Applications with Domain-specific Languages*. In Proceedings of Third International Workshop on Model-Driven Web Engineering (MDWE 2007), in conjunction with Seventh International Conference on Web Engineering (ICWE2007). Como, Italy CEUR Workshop Proceedings, ISSN 1613-0073., ISBN: ISSN 1613-0073.
- Freudenstein, P., L. Liu, F. Majer, A. Maurer, C. Momm, D. Ried and W. Juling (2006). *Architektur für ein universitätsweit integriertes Informations- und Dienstmanagement*. In Proceedings of Tagungsband zur INFORMATIK 2006 - Informatik für Menschen, 36. Jahrestagung der Gesellschaft für Informatik. Dresden, October 2006.
- Freudenstein, P., F. Majer and A. Maurer (2006). *SOA in der Praxis - eine Referenzarchitektur*. dot.net-magazin(11/2006): 22-27. ISSN: 1619-7933
- Freudenstein, P., F. Majer, A. Maurer, D. Ried and W. Juling (2007). *Wiederverwendungsorientierte Dienste für Universitäten*. In Proceedings of INFORMATIK 2007, 37. Jahrestagung der Gesellschaft für Informatik. Bremen, Germany, September 2007.
- Freudenstein, P., F. Majer and M. Nussbaumer (2008). *Agile WebPart-Entwicklung*. dot.net-magazin(11/2008): 92-95. ISSN: 1619-7933
- Freudenstein, P. and M. Nussbaumer (2008a). *Constructing Advanced Web-based Dialog Components with Stakeholders - a DSL Approach*. In Proceedings of 8th International Conference on Web Engineering (ICWE2008). New York, USA, 14-18 July 2008. Institute of Electrical and Electronics Engineers (IEEE), ISBN: 978-0-7695-3261-5.
- Freudenstein, P. and M. Nussbaumer (2008b). *The Dialog DSL: Rapid Development of Advanced Web-based Dialogs with Stakeholders*, University of Karlsruhe (TH), Technical Report. 14 pages, Karlsruhe, Germany, 2008-6 - ISSN: 1432-7864, from: <http://digbib.ubka.uni-karlsruhe.de/volltexte/1000008071>.
- Freudenstein, P., M. Nussbaumer, F. Allerding and M. Gaedke (2008). *A Domain-specific Language for the Model-driven Construction of Advanced Web-based Dialogs*. In Proceedings of 17th International World Wide Web Conference (WWW2008). Beijing, China, 21-25 April 2008. Association for Computing Machinery, ISBN: 978-1-60558-085-2.
- Freudenstein, P., M. Nussbaumer, F. Majer and M. Gaedke (2007). *A Workflow-Driven Approach for the Efficient Integration of Web Services in Portals*. In

- Proceedings of IEEE International Conference on Services Computing 2007 (SCC 2007). Salt Lake City, Utah, USA, 9-13 July 2007. IEEE Computer Society, ISBN: 0-7695-2925-9.
- Gaedke, M., M. Nussbaumer and J. Meinecke (2005). *WSLS: An Agile System Facilitating the Production of Service-Oriented Web Applications*. In: Engineering Advanced Web Applications. S. C. M. Matera (ed.), Rinton Press: 26-37. ISBN: 1-58949-046-0
- Gaedke, M. and J. Rehse (2000). *Supporting Compositional Reuse in Component-Based Web Engineering*. In Proceedings of 2000 ACM Symposium on Applied Computing (SAC 2000). Villa Olmo, Como, Italy, 19.-21.03.2000. ACM
- Gamma, E., R. Helm, R. Johnson and J. Vlissides (1995). *Design patterns: elements of reusable object-oriented software*. Reading, Mass., Addison-Wesley. xv, 395 pages. ISBN: 0201633612 (acid-free paper)
- Ginige, A. and S. Murugesan (2001). *Guest Editors' Introduction: Web Engineering: An Introduction*. IEEE MultiMedia 8(1): 14-18. ISSN: 1070-986X
- Gootzit, D., G. Phifer, R. Valdes, N. Drakos, A. Bradley, K. Harris, D. Sholler, M. Pezzini, Y. V. Natis, B. Gassman, D. M. Smith, D. W. Cearley, R. W. Schulte, S. Prentice, N. Gall, W. Clark and A. Lapkin (2008). *Hype Cycle for Web and User Interaction Technologies, 2008*, Gartner, Inc., Research Report. Stanford, CT, USA, G00159447, 7 July 2008
- Haase, P. (2007). *Part III - Ontology Evolution*. In: Semantic Technologies for Distributed Information Systems. Karlsruhe, Universitätsverlag Karlsruhe: 226. ISBN: 3866441002
- Hailpern, B. and P. Tarr (2006). *Model-driven development: The good, the bad, and the ugly*. IBM Systems Journal 45(3): 451-461. ISSN: 0018-8670
- Hailstone, R., R. Illsley, T. Jones and A. Kellett (2007). *SOA Platforms*, Butler Direct Limited, 322 pages, BG-0041, June 2007
- Havey, M. (2006). *Keeping BPM Simple for Business Users*. BP Trends Business Process Trends. Retrieved 23.03.2009, from <http://www.bptrends.com/publicationfiles/01-06-ART-KeepingBPMSimple-Havey.pdf>.
- Hennicker, R. and N. Koch (2000). *A UML-based Methodology for Hypermedia Design*. In Proceedings of Third International Conference on the Unified Modeling Language (UML'2000). York, UKSpringer Verlag, ISBN: 354041133X.
- Hennicker, R. and N. Koch (2001). *Modeling the User Interface of Web Applications with UML*. In Proceedings of Practical UML-Based Rigorous Development Methods Workshop at the UML 2001. Köllen Druck+Verlag
- Herman, J. (2004). *A process for creating the business case for user experience projects*. In Proceedings of Conference on Human Factors in Computing Systems. Vienna, AustriaACM, New York, USA, ISBN: 1-58113-703-6
- Heskett, J. L., W. E. Sasser and L. A. Schlesinger (1997). *The Service Profit Chain*. New York, USA, Free Press. 320 pages. ISBN: 0684832569

- Hewlett-Packard Development Company. (2003). *Homepage of the Jena Semantic Web Framework*. SourceForge.net. Retrieved 17.04.2009, from <http://jena.sourceforge.net/>.
- Hill, J. B., J. Sinur, D. Flint and M. J. Melenovsky (2006). *Gartner's Position on Business Process Management (2006)*, Gartner, Inc., Research Report. 26 pages, Stanford, CT, USA, G00136533, 16 February 2006
- Hornung, T., A. Koschmider and J. Mendling (2006). *Integration of heterogeneous BPM Schemas: The Case of XPD and BPEL*. In Proceedings of The 18th Conference on Advanced Information Systems Engineering (CAISE '06), Forum Proceedings. Luxembourg, 5-9 June 2006. CEUR Workshop Proceedings - CEUR-WS.org
- IBM. (2006). *IBM WebSphere MQ Workflow Homepage*. Retrieved 08.02.2009, 209, from <http://www-306.ibm.com/software/integration/wmqwf/>.
- IBM Corp. (2006). *Making Business Better: Business Process Management with SOA*. IBM Corporation. Retrieved 02.03.2009, 2009, from ftp://ftp.software.ibm.com/software/websphere/pdf/2007WSB11257-USEN-00_BPM_brochure_0921.pdf.
- IBM Corp. (2008a). *IBM Lotus Forms Homepage*. International Business Machines (IBM) Corporation. Retrieved 04.03.2009, from <http://www.ibm.com/software/lotus/forms/>.
- IBM Corp. (2008b). *IBM Lotus Forms Server 3.5 - Webform Server Homepage*. International Business Machines (IBM) Corporation. Retrieved 05.03.2009, from <http://publib.boulder.ibm.com/infocenter/forms/v3r5m0/index.jsp?topic=/com.ibm.form.webform.doc/toc.html>.
- IBM Corp. (2008c). *IBM Lotus Forms Turbo Homepage*. International Business Machines (IBM) Corporation. Retrieved 05.03.2009, from <http://www.ibm.com/software/lotus/products/forms/turbo/>.
- IBM Corp. (2008d). *IBM Rational Asset Manager Homepage*. International Business Machines (IBM) Corporation. Retrieved 06.03.2009, from <http://www.ibm.com/software/awdtools/ram/>.
- ISO/IEC 29500:2008 (2008). *Information technology – Document description and processing languages - Office Open XML file formats.*, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), ISO/IEC 29500:2008, from: <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>.
- Joran, J. (1954). *Universals in Management Planning and Controlling*. The Management Review 43(11): 748-761.
- Jordan, D. and J. Evdemon. (2007). *Web Services Business Process Execution Language Version 2.0*. OASIS Standard, Organization for the Advancement of Structured Information Standards (OASIS). Retrieved 18.03.2009, from <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html>.

- Jouault, F. and W. Piers. (2009). *The Atlas Transformation Language (ATL) - User Guide*. The Eclipse Foundation. Retrieved 07.04.2009, from http://wiki.eclipse.org/ATL/User_Guide.
- Juling, W. (2005). *KIM Project Homepage*. University of Karlsruhe (TH). Retrieved 12.02.2008, from <http://www.kim.uni-karlsruhe.de/>.
- Kappel, G., B. Pröll, S. Reich and W. Retschitzegger (2006). *Web Engineering: The Discipline of Systematic Development*, John Wiley & Sons. ISBN: 0-470-01554-3
- Karlsruhe Institute of Technology. (2007). *Karlsruhe Institute of Technology (KIT) Homepage*. Retrieved 09.02.2009, 2009, from <http://www.kit.edu>.
- Karlsruhe Institute of Technology. (2009). *Form for requesting and granting business trips as well as for stating expenses*. Retrieved 06.02.2009, 2009, from http://www.zvw.uni-karlsruhe.de/download/U.KA_0038.pdf.
- Kieburtz, R. B., L. McKinney, J. M. Bell, J. Hook, A. Kotov, J. Lewis, D. P. Oliva, T. Sheard, I. Smith and L. Walton (1996). *A software engineering experiment in software component generation*. In Proceedings of 18th International Conference on Software Engineering. IEEE Computer Society Press
- Kindler, E. (2006). *The Petri Net Markup Language and ISO/IEC 15909-2: Concepts, Status, and Future Directions*. In Proceedings of Entwurf Komplexer Automatisierungssysteme (EKA 2006). Braunschweig, Germany, 29-31 May 2006.
- Kindler, E. (2007). *ISO/IEC 15909-2: Software and Systems Engineering – High-level Petri Nets – Part 2: Transfer Format, Working Draft Version 1.1.4*, International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC), 62 pages, ISO/IEC 15909, 2007-01-22, from: http://www.pnml.org/papers/cd_1.1.4.pdf.
- Kiss, C. (2007, 30 April 2007). *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0*. W3C Working Draft, World Wide Web Consortium (W3C). Retrieved 13.04.2009, from <http://www.w3.org/TR/CCPP-struct-vocab2/>.
- Klink, S., Y. Li and A. Oberweis (2008). *INCOME2010 - a Toolset for Developing Process-Oriented Information Systems Based on Petri Nets*. In Proceedings of First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems. Marseille, France, ISBN: 978-963-9799-20-2.
- Klyne, G. and J. J. Carroll (2004). *Resource Description Framework (RDF): Concepts and Abstract Syntax*, World Wide Web Consortium (W3C), W3C Recommendation. 10 February 2004, from: <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>.
- Knapp, A., N. Koch, F. Moser and G. Zhang (2003). *ArgoUWE: A CASE Tool for Web Applications*. In Proceedings of First International Workshop on Engineering Methods to Support Information Systems Evolution (EMSISE'03). Geneva

- Knapp, A., N. Koch, G. Zhang and H.-M. Hassler (2004). *Modeling Business Processes in Web Applications with ArgoUWE*. In Proceedings of 7th International Conference on the Unified Modeling Language (UML2004). Lisbon, PortugalSpringer Verlag
- Koch, N. and A. Kraus (2003). *Towards a Common Metamodel for the Development of Web Applications*. In Proceedings of Third International Conference on Web Engineering (ICWE'03). Oviedo, SpainSpringer Verlag, ISBN: 3-540-40522-4.
- Koch, N., A. Kraus, C. Cachero and S. Melia (2003). *Modeling Web Business Processes with OO-H and UWE*. In Proceedings of Third Int. Worskhop on Web-oriented Software Technology (IWWOST'03). Oviedo, Spain
- Kotoric, D. (2007). *Usability of Desktop Applications and Rich Internet Applications*. School of Information Systems and Information Technology, Faculty of Business and Law; Science and Technology, Deakin University, Melbourne, Australia, 22 November 2007. 163 pages.
- Kraus, A., A. Knapp and N. Koch (2007). *Model-Driven Generation of Web Applications in UWE*. In Proceedings of 3rd International Workshop on Model-Driven Web Engineering (MDWE 2007). Como, ItalyCEUR Workshop Proceedings, ISSN 1613-0073.
- Krueger, C. W. (1992). *Software reuse*. ACM Computing Surveys 24(131): 131-183.
- Lawrence, K., C. Kaler, A. Nadalin, M. Goodner, M. Gudgin, A. Barbir and H. Granqvist. (2007). *WS-SecurityPolicy v1.2*. OASIS Standard Specification, Organization for the Advancement of Structured Information Standards (OASIS). Retrieved 20.03.2009, from <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html>.
- Lawrence, K., C. Kaler, A. Nadalin, R. Monzillo and P. Hallam-Baker. (2006). *Web Services Security: SOAP Message Security 1.1*. OASIS Standard Specification, Organization for the Advancement of Structured Information Standards (OASIS). Retrieved 20.03.2009, from <http://docs.oasis-open.org/wss/v1.1/>.
- Linaje, M., J. C. Preciado and F. Sánchez-Figueroa (2007). *A Method for Model Based Design of Rich Internet Application Interactive User Interfaces*. In Proceedings of Seventh International Conference on Web Engineering. Como, ItalySpringer Verlag, Heidelberg, ISBN: 978-3-540-73596-0.
- Lockhart, H., S. Andersen, J. Bohren, Y. Sverdlov, M. Hondo, H. Maruyama, A. Nadalin, N. Nagaratnam, T. Boubez, K. S. Morrison, C. Kaler, A. Nanda, D. Schmidt, D. Walters, H. Wilson, L. Burch, D. Earl, S. Baja and H. Prafullchandra (2006). *Web Services Federation Language (WS-Federation) v.1.1*, BEA Systems, BMC Software, CA, Inc., IBM, Layer 7 Technologies, Microsoft, Novell, VeriSign, 124 pages, December 2006, from: <http://www-106.ibm.com/developerworks/webservices/library/ws-fed/>.
- Lucas, K., M. Adrian, R. Wang and D. Krauss (2007). *The State Of Enterprise Software Adoption In Europe*, Forrester Research, 15 pages, Cambridge, USA, 29 January 2007

- MacVittie, L. A. (2006). *XAML in a Nutshell*, O'Reilly Media. ISBN: 0596526733
- Madsen, K. H. (1999). *The Diversity of Usability Practices*. Communications of the ACM 42(5). ISSN: 0001-0782
- Majer, F., M. Nussbaumer and P. Freudenstein (2009). *Operational Challenges and Solutions for Mashups – An Experience Report*. In Proceedings of Second Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), held in conjunction with 18th International World Wide Web Conference (WWW 2009). Madrid, Spain, 20 April 2009. Association of Computer Machinery (ACM), ISBN: 978-1-60558-487-4.
- Majer, F., M. Nussbaumer and M. Gaedke (2008). *A Descriptive Approach for the Lifecycle Support of Distributed Web-based Systems*. In Proceedings of Fourth International Conference on Web Information Systems and Technologies (WEBIST 2008). Funchal, Portugal., 4-7 May 2008. INSTICC Press, ISBN: 978-989-8111-26-5.
- Matera, M., F. Rizzo and G. T. Carughi (2006). *Web Usability: Principles and Evaluation Methods*. In: Web Engineering. E. Mendes and N. Mosley (eds.). Heidelberg, Springer: 143-180.
- McConnell, S. (1996). *Chapter 33: Reuse*. In: Rapid Development. Redmond, Washington, USA, Microsoft Press: 527-538.
- McDonald, A. and R. Welland (2001). *Web Engineering in Practice*. In Proceedings of 4th Workshop on Web Engineering (in conjunction with 10th International World Wide Web Conference). Hong Kong, May 2001.
- McIlroy, M. D. (1968). *Mass Produced Software Components*. In Proceedings of Software Engineering; Report on a conference by the NATO Science Committee. Garmisch, Germany, October 1968. NATO Scientific Affairs Division, Brussels, Belgium
- Meinecke, J., M. Nussbaumer and M. Gaedke (2005). *Building Blocks for Identity Federations*. In Proceedings of Fifth International Conference for Web Engineering (ICWE2005). Sydney, AustraliaSpringer, ISBN: 3-540-27996-2.
- Mendes, E. and N. Mosley, Eds. (2006). *Web Engineering*. Heidelberg, Springer. ISBN: 3-540-28196-7.
- Mendling, J., C. P. d. Laborda and U. Zdun (2005). *Towards an Integrated BPM Schema: Control Flow Heterogeneity of PNML and BPEL4WS*. In Proceedings of Third Conference on Professional Knowledge Management (WM 2005). Kaiserslautern, GermanySpringer Verlag Heidelberg / Berlin, Germany
- Mendling, J., G. Neumann and M. Nüttgens (2005). *A Comparison of XML Interchange Formats for Business Process Modelling*. In: Workflow Handbook 2005. L. Fischer (ed.), The Workflow Management Coalition (WfMC) with Future Strategies Inc., Book Division: 185-198. ISBN: 0970350988
- Menzel, M., I. Thomas, C. Wolter and C. Meinel (2007). *SOA Security - Secure Cross-Organizational Service Composition*. In Proceedings of Stuttgarter

- Softwaretechnik Forum (SSF). Stuttgart, Germany, November 2007. Fraunhofer IRB-Verlag, ISBN: 978-3-8167-7493-8.
- Merrill Lynch (2006). *Merill Lynch CIO Spending Study - "What Are Your Top Spending Priorities"*, July 2006
- Microsoft Corp. (2006a). *Homepage of the Microsoft Office Visio 2007 Software Development Kit (SDK)*. Microsoft Corporation. Retrieved 03.04.2009, from [http://msdn.microsoft.com/de-de/library/ms409183\(en-us\).aspx](http://msdn.microsoft.com/de-de/library/ms409183(en-us).aspx).
- Microsoft Corp. (2006b). *Microsoft Office Visio 2007 XML Schema Reference*. Microsoft Corporation. Retrieved 25.03.2009, from <http://msdn.microsoft.com/en-us/library/aa721908.aspx>.
- Microsoft Corp. (2006c). *Microsoft Windows Workflow Foundation Homepage*. Retrieved 08.02.2009, 2009, from <http://www.microsoft.com/net/WFDetails.aspx>.
- Microsoft Corp. (2007). *MSDN Library - Using Workflow Markup*. Microsoft Corporation. Retrieved 31.03.2009, from <http://msdn.microsoft.com/en-us/library/ms735921.aspx>.
- Microsoft Corp. (2009). *ASP.NET AJAX - The Official Microsoft ASP.NET Site*. Microsoft Corporation, Inc. Retrieved 13.04.2009, from <http://www.asp.net/ajax/>.
- Montgomery, D. C. (1997). *Design and Analysis of Experiments*, John Wiley & Sons. 720 pages. ISBN: 978-0471157465
- Moura, S. S. d. and D. Schwabe (2004). *Interface Development for Hypermedia Applications in the Semantic Web*. In Proceedings of Second Latin American Web Congress. Ribeirão Preto, Brazil IEEE Computer Society Press, ISBN: 0-7695-2237-8.
- Mukerji, J. and J. Miller (2003). *MDA Guide Version 1.0.1*, Object Management Group (OMG), Object Management Group (OMG) Specification. 62 pages, omg/03-06-01, 12 June 2003, from: <http://www.omg.org/cgi-bin/doc?omg/03-06-01>.
- Murphy, B. (2006). *The Forrester Wave: e-Forms Software, Q2 2006*, Forrester Research, Inc., 14 pages, Cambridge, MA, USA, June 16, 2006
- Murugesan, S., Y. Deshpande, S. Hansen and A. Ginige (1999). *Web Engineering: A New Discipline for Development of Web-Based Systems* In Proceedings of First ICSE Workshop on Web Engineering, held in conjunction with 21st International Conference on Software Engineering (ICSE). Los Angeles, USA
- Murzek, M. and G. Kramler (2007). *Business Process Model Transformation Issues - The Top 7 Adversaries Encountered at Defining Model Transformations*. In Proceedings of Ninth International Conference on Enterprise Information Systems (ICEIS 2007). Funchal, Madeira - Portugal, June 2007. INSTICC Press, ISBN: 978-972-8865-90-0.
- MWRG. (2009a). *IT Management and Web Engineering Homepage*. IT Management and Web Engineering Research Group, Institute of Telematics, Karlsruhe

- Institute of Technology (KIT). Retrieved 19.04.2009, from <http://mwrg.tm.uni-karlsruhe.de>.
- MWRG. (2009b). *IT Management and Web Engineering Research Site*. IT Management and Web Engineering Research Group, Institute of Telematics, Karlsruhe Institute of Technology (KIT). Retrieved 13.04.2009, from <http://research.tm.uka.de/>.
- Neighbors, J. M. (1984). *The Draco Approach to Constructing Software from Reusable Components*. IEEE Transactions on Software Engineering 10(5): 564-574.
- Nielsen, J. (1993). *Usability Engineering*, Academic Press, Cambridge, MA, USA. ISBN: 0-12-518405-0
- Nielsen, J. (2005). *Forms vs. Applications*. Jakob Nielsen's Alertbox Retrieved 15.04.2009, from <http://www.useit.com/alertbox/forms.html>.
- Nielsen, J. (2008). *Bridging the Designer-User Gap*. Jakob Nielsen's Alertbox Retrieved 15.04.2009, from <http://www.useit.com/alertbox/designer-user-differences.html>.
- Nielsen, J. and T. K. Landauer (1993). *A mathematical model of the finding of usability problems*. In Proceedings of INTERACT '93 and CHI '93 conference on Human factors in computing systems. Amsterdam, The Netherlands ACM, ISBN: 0-89791-575-5
- Nielsen, J. and H. Loranger (2006). *Prioritizing Web Usability*. Berkeley, CA, USA, New Riders Press. 432 pages. ISBN: 0321350316
- Nunes, D. A. and D. Schwabe (2006). *Rapid prototyping of web applications combining domain specific languages and model driven design*. In Proceedings of 15th International World Wide Web Conference. Edinburgh, Scotland Association for Computing Machinery, ISBN: 1-59593-323-9.
- Nussbaumer, M. (2001). *Einsatz von Petri Netzen und Xforms zur Modellierung und Unterstützung von interaktiven Web-Anwendungen*. Diploma Thesis, Institute of Telematics, University of Karlsruhe (TH), Karlsruhe, 31.12.2001. 125 pages.
- Nussbaumer, M. (2008). *Entwicklung und Evolution dienstorientierter Anwendungen im Web Engineering (Dissertation)*. Karlsruhe, Universitätsverlag Karlsruhe. 212 pages. ISBN: 978-3-86644-208-5
- Nussbaumer, M., P. Freudenstein and M. Gaedke (2006a). *The Impact of DSLs for Assembling Web Applications*. Engineering Letters 13(2006): 387-396. ISSN: 1816-093X
- Nussbaumer, M., P. Freudenstein and M. Gaedke (2006b). *Stakeholder Collaboration - From Conversation To Contribution*. In Proceedings of 6th International Conference on Web Engineering (ICWE). SLAC, Menlo Park, California, 12-14 July, 2006. ACM
- Nussbaumer, M., P. Freudenstein and M. Gaedke (2006c). *Towards DSL-based Web Engineering*. In Proceedings of 15. International World Wide Web Conference (WWW). Edinburgh, UK ACM

- Nussbaumer, M., P. Freudenstein and M. Gaedke (2006d). *Web Application Development employing Domain-Specific Languages*. In Proceedings of International Conference on Software Engineering (SE2006). Innsbruck, Austria, 14.-16.02.2006. IASTED/ACTA Press, ISBN: 0-88986-574-4.
- O'Reilly, T. (2005). *What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software*. Online Article, Retrieved 18.10.2005, from <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>.
- Oberweis, A. (2008). *INCOME2010 Homepage*. Institute of Applied Informatics and Formal Description Methods (AIFB), University of Karlsruhe (TH). Retrieved 28.03.2009, from <http://www.aifb.uni-karlsruhe.de/Forschungsgruppen/BIK/income2010/>.
- Object Management Group. (2005a). *Reusable Asset Specification v2.2*. OMG Available Specification, Object Management Group. Retrieved 06.03.2009, from <http://www.omg.org/docs/formal/05-11-02.pdf>.
- Object Management Group (2005b). *Unified Modeling Language (UML) v2.0: Superstructure*, Object Management Group (OMG), OMG Specification. OMG formal/2005-07-04, August 2005, from: <http://www.omg.org/spec/UML/2.0/Superstructure/PDF>.
- Object Management Group (2007). *MOF 2.0/XMI Mapping, Version 2.1.1*, Object Management Group (OMG), OMG Available Specification. 120 pages, OMG formal/2007-12-01, December 2007, from: <http://www.omg.org/spec/XMI/2.1.1/PDF/index.htm>.
- Orozov, N. (2008). *Ein Transformations-Rahmenwerk zur modellgetriebenen Entwicklung webbasierter Workflows mit Stakeholdern*. Diploma Thesis, Institute of Telematics, University of Karlsruhe (TH), Karlsruhe, 09.05.2008. 108 pages.
- Palmer, N. (2006). *Understanding the BPMN-XPDL-BPEL Value Chain*. Business Integration Journal: 54-55.
- Pender, T. (2003). *Chapter 21 - Customizing UML Using Profiles*. In: UML Bible. Indianapolis, Indiana, USA, Wiley Publishing, Inc. : 687-723. ISBN: 0764526049
- Petri, C. A. (1962). *Kommunikation mit Automaten*. Dissertation, Technischen Universität Darmstadt, Darmstadt,
- Phifer, G. (2006). *The Fifth Generation of Portals Supports SOA and Process Integration*, Gartner, Stanford, CT, USA, G00137923, 1 March 2006
- Phifer, G., D. Gootzit, D. Sholler, R. Valdes, A. Bradley, N. Drakos, R. W. Schulte, Y. V. Natis, D. M. Smith, C. Abrams, S. Prentice, W. A. D. A. Filho, D. W. Cearley, R. E. Knox, N. Jones, N. Gall, B. J. Lheureux and L. F. Kenney (2007). *Hype Cycle for Web and User Interaction Technologies, 2007*, Gartner, Inc., Stanford, CT, USA, G00148212, 13 July 2007

- Prechelt, L. (2001). *Kontrollierte Experimente in der Softwaretechnik - Potenzial und Methodik*, Springer Verlag. 273 pages. ISBN: 3-540-41257-3
- Preciado, J. C., M. Linaje, S. Comai and F. Sanchez-Figueroa (2007). *Designing Rich Internet Applications with Web Engineering Methodologies*. In Proceedings of 9th IEEE International Workshop on Web Site Evolution (WSE 2007). Paris, France IEEE Computer Society Press, ISBN: 978-1-4244-1450-5.
- Preciado, J. C., M. Linaje, R. Morales-Chaparro, F. Sanchez-Figueroa, G. Zhang, C. Kroiß and N. Koch (2008). *Designing Rich Internet Applications Combining UWE and RUX-Method*. In Proceedings of Eighth International Conference on Web Engineering (ICWE'08). White Plains, New York, USA IEEE Computer Society, Washington DC, USA, ISBN: 978-0-7695-3261-5.
- Preciado, J. C., M. Linaje, F. Sanchez and S. Comai (2005). *Necessity of methodologies to model rich Internet applications*. In Proceedings of Seventh IEEE International Symposium on Web Site Evolution 2005. (WSE 2005). Budapest, Hungary IEEE Computer Society
- Pressman, R. S. (2005a). *Chapter 30: Component-based Development*. In: Software Engineering: A Practitioner's Approach. New York, McGraw-Hill: 499-626. ISBN: 0071238409
- Pressman, R. S. (2005b). *Part Three: Applying Web Engineering*. In: Software Engineering: A Practitioner's Approach. New York, McGraw-Hill: 499-626. ISBN: 0071238409
- Prieto-Diaz, R. (2003). *A faceted approach to building ontologies*. In Proceedings of IEEE International Conference on Information Reuse and Integration. Las Vegas, USA
- Progeny Systems. (2007). *Homepage of the FormFaces Framework*. Progeny Systems Corporation. Retrieved 13.04.2009, from <http://formfaces.com/>.
- Prud'hommeaux, E. and A. Seaborne. (2008, 15 Januar 2008). *SPARQL Query Language for RDF*. W3C Recommendation, World Wide Web Consortium (W3C). from <http://www.w3.org/TR/rdf-sparql-query/>.
- Puschmann, T. and R. Alt (2004). *Process Portals - Architecture and Integration*. In Proceedings of 37th Hawaii International Conference on System Sciences. Hawaii, USA, 25.01.2004. IEEE Computer Society, ISBN: 0-7695-2056-1
- Raman, T. V. (1997). *Auditory User Interfaces--Toward The Speaking Computer*, Kluwer Academic Publishers. ISBN: 0-7923-9984-6
- Recker, J. (2008). *BPMN Modeling - Who, Where, How and Why*. BP Trends Business Process Trends. Retrieved 24.03.2009, from <http://www.bptrends.com/publicationfiles/05-08-ART-BPMN%20Survey-Recker-JR%20final.pdf>.
- Recker, J. and A. Dreiling (2007). *Does It Matter Which Process Modelling Language We Teach or Use? An Experimental Study on Understanding Process Modelling Languages without Formal Education*. In Proceedings of

- Proceedings of the 18th Australasian Conference on Information Systems (ACIS 2007). Toowoomba, AustraliaThe University of Southern Queensland
- Rode, J., M. B. Rosson and M. A. P. Quinones (2006). *End User Development of Web Applications*. In: End User Development. H. Lieberman, F. Paternò and V. Wulf (eds.), Springer Verlag Heidelberg / Berlin: 161-182. ISBN: 978-1-4020-4220-1
- Roger S. Pressman (2005). *Part Three: Applying Web Engineering*. In: Software Engineering: A Practitioner's Approach. New York, McGraw-Hill: 499-626. ISBN: 0071238409
- Rossi, G., O. Pastor, D. Schwabe and L. Olsina, Eds. (2008). *Web Engineering - Modelling and Implementing Web Applications*. London, Springer.ISBN: 978-1-84628-922-4.
- Rossi, G. H., H. A. Schmid and F. Lyardet (2003). *Customizing Business Processes in Web Applications*. In Proceedings of 4th International Conference on E-Commerce and Web Technologies. September 2003. Springer Verlag
- Russell, N., A. H. M. Ter Hofstede, D. Edmond and W. M. P. Van der Aalst (2004a). *Workflow Data Patterns*, Queensland University of Technology, QUT Technical report. 75 pages, Brisbane, Australia, FIT-TR-2004-01, from: http://www.workflowpatterns.com/documentation/documents/data_patterns%20BETA%20TR.pdf.
- Russell, N., A. H. M. Ter Hofstede, D. Edmond and W. M. P. Van der Aalst (2004b). *Workflow Resource Patterns*, Eindhoven University of Technology, BETA Working Paper Series. 73 pages, Eindhoven, Netherlands, WP 127, from: <http://www.workflowpatterns.com/documentation/documents/Resource%20Patterns%20BETA%20TR.pdf>.
- Russell, N., A. H. M. ter Hofstede, W. M. P. Van der Aalst and N. Mulyar (2006). *Workflow Control-Flow Patterns: A Revised View*, BPMCenter.org, BPM Center Report. BPM-06-22,
- Russell, N., W. M. P. van der Aalst, A. H. M. ter Hofstede and P. Wohed (2006). *On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling*. In Proceedings of Third Asia-Pacific Conference on Conceptual Modelling (APCCM2006). Hobart, AustraliaAustralian Computer Society, Inc., Darlinghurst, Australia
- Schmid, A. (2006). *Transforming Graphical Models Using XSLT*. Study Thesis, Institute of Telematics, University of Karlsruhe (TH), Karlsruhe, 28 February 2006. 50 pages.
- Schmid, H. A. and G. Rossi (2004). *Modeling and Designing Processes in E-Commerce Applications*. IEEE Internet Computing 8(1): 19-27.
- Schmidt, M.-T. and G. Larsen (2007). *Federated Metadata Management with IBM Rational and WebSphere Software.*, International Business Machines (IBM) Corporation, White Paper. 24 pages, Somers, NY, USA, June 2007, from: ftp://ftp.software.ibm.com/software/rational/web/whitepapers/10709147_Rational_RAM_WP_ACC.pdf.

- Schwabe, D., L. Esmeraldo, G. Rossi and F. Lyardet (2001). *Engineering Web applications for Reuse*. IEEE Multimedia 8(1): 20-31.
- Schwabe, D., G. Rossi and S. Barbosa (1996). *Systematic Hypermedia Design with OOHDM*. In Proceedings of ACM International Conference on Hypertext' 96. Washington, USA, March 1996.
- Schwinger, W. and N. Koch (2006). *Chapter 3: Modeling Web Applications*. In: Web Engineering: The Discipline of Systematic Development. G. Kappel, B. Pröll, S. Reich and W. Retschitzegger (eds.), John Wiley & Sons: 39-64. ISBN: 0-470-01554-3
- Selmi, S. S., N. Kraiem and H. B. Ghezala (2005). *Toward a Comprehension View of Web Engineering*. In Proceedings of 5th International Conference of Web Engineering (ICWE 2005). Sydney, AustraliaSpringer, ISBN: 3-540-27996-2.
- Setiawan, D. (2008). *Modellgetriebene Konstruktion workflow-basierter Web-Anwendungen mit UML 2.0 Aktivitätsdiagrammen*. Study Thesis, Institute of Telematics, University of Karlsruhe (TH), Karlsruhe, 31.01.2008. 156 pages.
- Setiawan, D. (2009). *Notationsübergreifende Konstruktion webbasierter Workflowanwendungen - Methodik und Evaluation*. Diploma Thesis, Institute of Telematics, University of Karlsruhe (TH), Karlsruhe, 31.01.2009. 136 pages.
- Shapiro, R., M. Marin, J. Brunt, W. Zurek, T. Stephenson, S. Bojanic and G. Gouri. (2005, 03.10.2005). *XML Process Definition Language (XPDL) 2.0 Specification*. Workflow Management Coalition. 2007.
- Silva, B. D. and A. Ginige (2007). *Meta-Model to Support End-User Development of Web Based Business Information Systems* In Proceedings of Seventh International Conference on Web Engineering (ICWE 2007). Como, ItalySpringer Verlag, Heidelberg / Berlin, ISBN: 978-3-540-73596-0.
- Sommerville, I. (2007a). *Chapter 18: Software Reuse*. In: Software Engineering, Pearson Education Ltd.: 415-438. ISBN: 0321313798
- Sommerville, I. (2007b). *Chapter 19: Component-based Software Engineering*. In: Software Engineering, Pearson Education Ltd.: 439-461. ISBN: 0321313798
- Stahl, T. and M. Völter (2006). *Model-Driven Software Development - Technology, Engineering, Management*, John Wiley & Sons. 444 pages. ISBN: 0470025700
- The Standish Group International. (1994-2008). *CHAOS Research*. Research Reports, from <http://www.standishgroup.com>.
- Thompson, H. S., D. Beech, M. Maloney and N. Mendelsohn. (2004). *XML Schema Part 1: Structures (Second Edition)*. W3C Recommendation, World Wide Web Consortium (W3C). 2009, from <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>.
- Tracz, W. (1990). *Where does reuse start?* ACM SIGSOFT Software Engineering Notes 15(2): 42-46. ISSN: 0163-5948

- Tullis, T. and B. Albert (2008). *Measuring the User Experience : Collecting, Analyzing, and Presenting Usability Metrics*, Morgan Kaufmann. 336 pages. ISBN: 978-0123735584
- Uschold, M. and M. King (1995). *Towards a Methodology for Building Ontologies*. In Proceedings of Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada
- Vallecillo, A., N. Koch, C. Cachero and S. Comai (2007). *MDWEnet: A Practical Approach to Achieving Interoperability of Model-Driven Web Engineering Methods*. In Proceedings of Third International Workshop on Model-Driven Web Engineering (MDWE'07). Como, Italy
- Van der Aalst, W. M. P. (1998). *The Application of Petri Nets to Workflow Management*. The Journal of Circuits, Systems and Computers 8(1): 21-66.
- Van der Aalst, W. M. P. (2007). *Trends in Business Process Analysis - From Verification to Process Mining*. In Proceedings of 9th International Conference on Enterprise Information Systems (ICEIS 2007). Madeira, Portugal Institute for Systems and Technologies of Information, Control and Communication (INSTICC)
- Van der Aalst, W. M. P., A. H. M. ter Hofstede, B. Kiepuszewski and A. P. Barros (2003). *Workflow Patterns*. Distributed and Parallel Databases 14(1): 5-51. ISSN: 0926-8782
- van der Aalst, W. M. P., A. H. M. ter Hofstede and M. Weske (2003). *Business Process Management: A Survey*. In Proceedings of International Conference on Business Process Management (BPM) 2003. Eindhoven, The Netherlands, 26-27 June 2003. Springer
- Van der Aalst, W. M. P., B. F. Van Dongen, C. W. Günther, R. S. Mans, A. K. Alves de Medeiros, A. Rozinat, V. Rubin, M. Song, H. M. W. Verbeek and A. J. M. M. Weijters (2007). *ProM 4.0: Comprehensive Support for Real Process Analysis*. In Proceedings of 28th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007). Siedlce, Poland, 25-29 June 2007. Springer Verlag, Berlin, Germany, ISBN: 978-3-540-73093-4.
- Vedamuthu, A. S., D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez and Ü. Yalçinalp. (2007). *Web Services Policy 1.5 - Framework*. W3C Recommendation, World Wide Web Consortium (W3C). Retrieved 20.03.2009, from <http://www.w3.org/TR/ws-policy/>.
- Verbeek, E. and W. M. P. Van der Aalst (2000). *Woflan 2.0: A Petri-net-based Workflow Diagnosis Tool*. In Proceedings of 21st International Conference on Application and Theory of Petri Nets (ICATPN 2000). Aarhus, Denmark, 26-30 June 2000. Springer Verlag Berlin/Heidelberg, Germany, ISBN: 3-540-67693-7.
- Wahli, U., V. Avula, H. Macleod, M. Saeed and A. Vinther (2007). *Business Process Management: Modeling through Monitoring Using WebSphere V6.0.2 Products*, International Business Machines (IBM) Corporation. 670 pages. ISBN: 0738489123

- Warmer, J. and A. Kleppe (2003). *The Object Constraint Language: Getting Your Models Ready for MDA*. Boston, MA, USA, Addison-Wesley. 240 pages. ISBN: 0-321-17936-6
- Warshall, S. (1962). *A Theorem on Boolean Matrices*. Journal of the ACM 9(1): 11-12. ISSN: 0004-5411
- Welie, M. v. and H. Trætteberg (2000). *Interaction Patterns in User Interfaces*. In Proceedings of Seventh Pattern Languages of Programs Conference. Illinois, USA
- Wenzel, F. (2009). *Transaction Management Challenges for Federated, Workflow-based SOA Applications*. Diploma Thesis, Institute of Telematics, University of Karlsruhe (TH), Karlsruhe, 31.03.2009. 106 pages.
- Wenzel, F., P. Freudenstein and M. Nussbaumer (2009). *Strengths and Weaknesses of WS-BusinessActivity for Cross-Organizational SOA Applications*. In Proceedings of Workshop on Principles of Engineering Service-oriented Systems (PESOS 2009), held in conjunction with 31st International Conference on Software Engineering. Vancouver, Canada, 16-24 May 2009.
- Weske, M., G. Vossen and F. Puhmann (2005). *Workflow and Service Composition Languages*. In: Handbook on Architectures of Information Systems. P. Bernus, K. Mertins and G. Schmidt (eds.), Springer Verlag, Berlin: 369-390. ISBN: 9783540266617
- White, S. A. (2006). *Business Process Modeling Notation (BPMN) Specification*, Object Management Group (OMG), OMG Final Adopted Specification. 308 pages, dtc/06-02-01, February 2006
- Wieggers, K. E. (2003). *Software Requirements*, Microsoft Press. 430 pages. ISBN: 0735618798
- Wireless Application Forum (2001). *User Agent Profile Specification v.2.0*, Wireless Application Forum, Ltd., Wireless Application Protocol. 86 pages, WAP-248-UAPROF-20011020-a, 20 October 2001, from: <http://www.openmobilealliance.org/tech/affiliates/wap/wap-248-uaprof-20011020-a.pdf>.
- Wohlin, C., P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell and A. Wesslén (2000). *Experimentation in Software Engineering - An Introduction*, Springer Verlag. 228 pages. ISBN: 978-0-7923-8682-7
- Workflow Management Coalition (1997). *The WfMC Glossary*. In: Workflow Handbook. P. Lawrence (ed.). New York, John Wiley and Sons: 385-421. ISBN: 0-471-96947-8
- Workflow Management Coalition. (2009). *XPDL Implementations Overview*. Workflow Management Coalition. Retrieved 18.03.2009, from <http://www.wfmc.org/xpdl-implementations.html>.
- Workflow Patterns Initiative. (2007). *Evaluation Results for XPDL version 2.0 against the Workflow Control-Flow Patterns*. Retrieved 18.03.2009, from <http://www.workflowpatterns.com/evaluations/standard/xpdl.php>.

- World Wide Web Consortium. (2007). *Mobile Web Initiative*. World Wide Web Consortium (W3C). Retrieved 27.04.2009, from <http://www.w3.org/Mobile/>.
- Wroblewski, L. (2008). *Web Form Design: Filling in the Blanks*. New York, USA, Rosenfeld Media. 226 pages. ISBN: 1-933820-25-X
- Wüstner, E., T. Hotzel and P. Buxmann (2002). *Converting Business Documents: A Classification of Problems and Solutions using XML/XSLT*. In Proceedings of Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002). Newport Beach, California, USA, 26-28 June 2002. IEEE Computer Society, Washington, DC, USA, ISBN: ISBN:0-7695-1567-3.
- Yglesias, K. P. (1998). *IBM's Reuse Programs: Knowledge Management and Software Reuse*. In Proceedings of Fifth International Conference on Software Reuse. Victoria, BC, Canada, June 2-5, 1998. IEEE Computer Society, Washington, DC, USA, ISBN: 0-8186-8377-5
- Yongbeom, K. and A. S. Edward (1998). *Software Reuse: Survey and Research Directions*. Journal of Management Information Systems 14(4): 113-147. ISSN: 0742-1222
- Zur Muehlen, M. and J. Recker (2008). *How Much Language is Enough? Theoretical and Practical Use of the Business Process Modeling Notation*. In Proceedings of 20th International Conference on Advanced Information Systems Engineering (CAiSE 2008). Montpellier, France, 16-20 June 2008. Springer Verlag, Heidelberg / Berlin, Germany, ISBN: 978-3-540-69533-2.
- Zur Muehlen, M., J. Recker and M. Indulska (2007). *Sometimes Less is More: Are Process Modeling Languages Overly Complex?* In Proceedings of EDOC2007 Workshops - Third International Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE 2007). Baltimore, Maryland, USA, 15 October 2007. IEEE Press

Index

A

Activity Building Blocks (ABBs) 59, 63, 64, **68**, 72, 123
evaluation 182, 185, 187
Agility 17, 46, 126, 150
Application assembly 55, 123, 126
ArgoUWE..... 33, 157

B

Business Process 13
Business Process Execution Language (BPEL) . 37, 66,
106, 109
Business Process Management (BPM) 7, 13, 34
Business Process Modeling Notation (BPMN). 19, 31,
36, 37, 58, 63, **79**, 96, 113

C

Commit ABB 69
Component-based software engineering 17, 56
Composite Capability Preferences Profile 150
Concern Configuration 64, 80, 85, 87, 107
Core Elements Set (CES) 95
CES-based transformation concept *See Model
Transformation Framework*
evaluation of expressiveness 178

D

Data Presentation ABB 70
DatadiagramML 81
Device-independent access .16, 24, 69, 70, 138, 143,
150
Dialog DSL 59, 69, **135**, 162
Appearance Design 138, 147
Appearance Modeling Tier 142
Domain Interaction Model (DIM) ... 136, **140**, 196
Domain-Specific Model (DSM)..... 136, **138**
evaluation of efficiency..... 189
evaluation of stakeholder adequacy..... 196
model transformations 143
Partition Design 137, 146
Partitions & Transitions Modeling Tier 141

process model..... 137
reuse 137, 149
Solution Building Block (SBB) 136, 149
stakeholder orientation 137, 140, 146
technical platform 145
Web-based DIM editor 146
Dialog Partition 136, **139**, 141, 142
Dialog-based user interaction 69, 135
challenges 22, 46
Dialog-based User Interaction ABB..... 88
DIM Taxonomy..... 54
Domain Abstract Representation (DAR)..... 55
Domain Interaction Model (DIM)..... 53
Domain-Specific Language (DSL)..... 52, *See Web
Engineering DSL Framework*
Domain-Specific Model (DSM) 53
DSL-based Web Engineering *See Web Engineering
DSL Framework*

E

End-User Development (EUD)..... 47
Enterprise Application Integration (EAI) 163, 165
Evaluation 177
Dialog DSL 189
formal experiment 189
survey 196
Workflow DSL 178, 183
Evolution 17, 46, 52, 55, 65, 138, 150
Evolutionary prototypes 21, 25, 129, 149
Extensible Stylesheet Language Transformations
(XSLT) 92, 102, 119

F

Federative scenarios 45, 129
Finite State Machine (FSM)..... 185

G

Goal/Question/Metric (GQM)..... 190, 196

H

HyperDE 30

- I**
- IBM Lotus Forms Designer 40
 IBM Lotus Forms Turbo..... 42
 IBM Rational Asset Manager..... 44
 IBM Rational Software Architect..... 19, 63, 82
 IBM WebSphere 34
 IBM's federated metadata management strategy. 44
 INCOME2010..... 19, 63, 86
 Interaction Elements..... 136, **138**, 140, 143
 Interaction Structures 136, **138**, 140, 141
- J**
- Jena Semantic Web Framework..... 165
- K**
- Karlsruhe's Integrated Information Management
 (KIM) 19, 177, 178, 183
 integrated Service Oriented Architecture (iSOA)
 183
- M**
- Microsoft Office SharePoint Server 2007..... 187
 Microsoft Visio 19, 63, 79
 Microsoft Windows Workflow Foundation (WF) 106,
 121, 187
 Microsoft Word..... 19, 63, 78
 Minimal physical configuration aspects..... 68
 Model consistency 20, 46, 74, 95
 Model continuity..... 76, 81, 126
 Model Transformation Framework..... 59, 77, **91**
 CES-based transformation concept 96, 99
 composite end-to-end transformations 94
 DIM mappings catalog 113
 graph-structured vs. block-structured 109
 horizontal transformation 93, 99
 layout algorithm 105
 technical platform 116
 transformation algorithm 102
 transformation strategies 91
 vertical transformation 93, 106
 Model-code gap 21, 46
 Model-Driven Architecture (MDA) 33
 Model-driven software development 17, 20
 Model-Driven Web Engineering initiative
 (MDWEnet) 26, 43, 158
 Multi-notational modeling 19, 36, **76**, 93, 95
- O**
- Object-Oriented Hypermedia Design Method
 (OOHDM) 26, 29, 37, 43
 Office Open XML..... 78
- P**
- Pareto principle..... 96
 Petri Net..... 19, 58, 63, **86**, 99, 113, 136, 140, 141
 Petri Net Markup Language (PNML) 86
- Process Intermediate Language 59, 62, **65**
- R**
- Rapid prototyping 24, 126, 149
 Requirements engineering..... 18, 76, 78
 Resource Description Framework (RDF) 165
 Reuse **153**
 challenges 17, 26, 46
 context..... 27, 137, 149, 153, 156, 163, 168
 coordination 27, 155
 cross-methodological ... 26, 43, 60, 158, 159, 160,
 168, 171
 Dialog DSL 137, 149
 stakeholder-oriented 60, 157, 158, 160, 162, 168,
 171
 Workflow DSL 64
 Rich User Experience (RUX) Method..... 38, 40
- S**
- Semantic Hypermedia Design Method (SHDM) 37
 Semantic Web 37, 46, 60, 165
 Service Component Architecture (SCA) 35
 Service orientation 12, 24, 62, 129, 131, 165, 183,
 185
 Simple Sequence Only (SSO) 58, 63, **78**, 113
 Solution Building Block (SBB) 54
 SPARQL Protocol and RDF Query Language 162, 165,
 169
 Stakeholder Adequacy Scale (SAS) 197, 200
 Stakeholder collaboration / involvement **51**
 challenges 11, 17, 24, 26, 47
 dialog development *See Dialog DSL*
 reuse *See Reuse*
 workflow development..... *See Workflow DSL*
 State of the Art..... 29, 45
 System Usability Scale (SUS) 197
- T**
- Transformations
 Dialog DSL *See Dialog DSL*
 Workflow DSL *See Model Transformation*
 Framework
 t-test 195
- U**
- UML Activity Diagrams..... 58, 63, **82**, 113
 UML-based Web Engineering (UWE) 32, 39, 157, 171
 Unified Modeling Language (UML) 19, 30, 32, 82
 Usability 23
 best practices..... 23, 148, 190
 usability-oriented design 141, 146, 148
 User Agent Profile (UAPProf) 150
 User Interaction Workflows..... 185
- W**
- Web application 7
 Web Engineering..... 7, 29

- Web Engineering DSL Framework.... **51**, 62, 136, 161
 - Development for Reuse 53
 - Development with Reuse..... 55
 - DSL components 53
 - DSL Librarian team role 56
 - DSL Reuse Repository 56
 - evolution..... 55
 - model transformations *See Model Transformation Framework*
 - technical platform 56
 - Web Engineering Reuse Ontology *See Web Engineering Reuse Sphere*
 - Web Engineering Reuse Sphere 60, **153**
 - distributed repositories 154
 - reference architecture framework 165
 - search and integration..... 162, 168, 171
 - semantic registry 154
 - sphere concept 154
 - stakeholder orientation *See Reuse*
 - storing and registering..... 163
 - Web Engineering Reuse Ontology 155, 165
 - Web Modeling Language (WebML) 26, 31, 38, 43, 157, 160, 171
 - Web Ontology Language (OWL)..... 155, 165
 - Web Service Communication ABB 71
 - Web service integration scenarios..... 184, 185
 - WebComposition 26, 43
 - WebComposition Service Linking System (WSLS) . 56, 123, 167, 173
 - WebRatio 31, 43, 157, 160, 171
 - Web-specific concerns 64, 66, **72**, 79
 - Workflow 13
 - User Interaction Workflow 185
 - Workflow activity types 15, 68
 - Workflow application..... 13
 - Workflow DSL..... 58, **62**
 - automated application construction 126
 - Business Process Modeling..... 64
 - Domain Interaction Models (DIMs) 62, **76**, 113
 - Domain-Specific Model (DSM)..... 62, **65**, 73, 113
 - evaluation 178, 183, 188
 - evolution..... 65, 74
 - model transformations *See Model Transformation Framework*
 - modeling notations / editors 62, 76
 - Physical Design & Execution 65
 - process model..... 58, **63**
 - reuse 64
 - Solution Building Block (SBB) 63, 123
 - stakeholder orientation 62, 64, 68, **76**
 - technical platform 115
 - Workflow Modeling 64, 87, 90
 - Workflow engine..... 15, 63, 121, 187
 - Workflow interaction channels..... 16, 121, 129
 - Workflow patterns..... 66, 67, 97
 - Workflow perspectives 45, 97
 - Workflow transaction management 207
 - Workflow-based Web Application 13
 - challenges 12, 45
- X**
- XAML..... 24, 106, 144
 - XForms 24, 41, 136, 139, 144
 - XML Metadata Interchange (XMI)..... 37, 84
 - XML Process Definition Language (XPDL).. 37, 62, **66**, 113
 - Web-specific extensions 72
 - XOML 106, 109
 - XSLT.....*See Extensible Stylesheet Language Transformations*