Karlsruher Institut für Technologie

Karlsruhe Reports in Informatics 2012,21
Edited by Karlsruhe Institute of Technology,
Faculty of Informatics
ISSN 2190-4782

Palladio Days 2012

Proceedings

8–9 November 2012
Universität Paderborn, Germany

Steffen Becker, Jens Happe,
Anne Koziolek, Ralf Reussner (Editors)

2012

KIT – University of the State of Baden-Wuerttemberg and National Research Center of the Helmholtz Association

Fakultät für **Informatik**

Palladio Days 2012

Proceedings

8–9 November 2012
Universität Paderborn, Germany

Steffen Becker, Jens Happe,
Anne Koziolek, Ralf Reussner (Editors)

Karlsruher Institut für Technologie
Fakultät für Informatik
Bibliothek
Postfach 6980
76128 Karlsruhe

ISSN 2190-4782

Preface

The prediction of software quality (e.g. performance, reliability and maintainability) based on software architectures is useful in many software development scenarios, such as support for design decisions, resource dimensioning or scalability analysis.

The open source tool Palladio can be seen as a "software architecture simulator". Palladio includes a metamodel for specifying software architectures (Palladio Component Model, PCM), a simulator (SimuCom), and a set of analytical solvers to gather simulation data on different software quality attributes. By its flexible design, extensive documentation, and high number of industrial case studies, Palladio is a mature platform to be utilised by other developers and scientists to explore further possibilities of modelling and simulating architectures. There are several dimensions of building on Palladio: extending Palladio for specific application domains, such as embedded systems, adding analyses for additional quality metrics (such as maintainability) or using Palladio for non-software architectures (e.g., production plants or logistics).

Therefore, the Palladio Days 2012 have the goal to bring together practitioners using Palladio and researchers who intend to work on Palladio as well as those who drive the Palladio project or who would like to learn about Palladio and its latest improvements.

Specific topics of this year's Palladio Days is *Performance Prediction of Software in Dynamic (On-the-Fly) Contexts*. In general, we seek reports on applications and extensions of Palladio in academic or scientific contexts.

Programme Committee Chairs

- Steffen Becker, University of Paderborn
- Jens Happe, SAP Research Karlsruhe
- Anne Koziolek, University of Zurich
- Ralf Reussner, Karlsruhe Institute of Technology/FZI Forschungszentrum Informatik

Programme Committee

- Babora Buhnova, University of Brno
- Wilhelm Hasselbring, University of Kiel
- Samuel Kounev, Karlsruhe Institute of Technology
- Heiko Koziolek, ABB Research
- Klaus Krogmann, FZI Forschungszentrum Informatik
- Heike Wehrheim, University of Paderborn

Organizers

- Matthias Becker, University of Paderborn, Germany
- Steffen Becker, University of Paderborn, Germany
- Sebastian Lehrig, University of Paderborn, Germany

Programme

Thursday, 8 November

9.00 Developer Meeting

12.00 Lunch Break

13.00 Opening and Welcome (Organizers)

13.15 **Keynote**

Jun.-Prof. Dr. Oliver Hummel, Karlsruhe Institute of Technology
"Reuse and Beyond: Innovative Software Retrieval Approaches"

14.00 Break

14.30 **Palladio Days Paper Session**

Max E. Kramer, Zoya Durdik, Michael Hauck, Jörg Henss, Martin Küster, Philipp Merkle and Andreas Rentschler
"Extending the Palladio Component Model using Profiles and Stereotypes"

Robert Heinrich, Jörg Henss and Barbara Paech
"Extending Palladio by Business Process Simulation Concepts"

Jens Frieben and Henning Heutger
"Case Study: Palladio Based Modular System For Simulating PLC Performance"

16.00 Break

16.30 Heinz-Nixdorf Museum

19.00 Dinner and Socializing

Friday, 9 November

9.00 **Special Focus Talks**

Andreas Brunnert and Christian Vögele
"Applying the Palladio tool in a SOA Project"

Benjamin Klatt and Christoph Rathfelder
"Predicting Event-Based Communication with Palladio"

Matthias Becker
"SimuLizar: Design-Time Modeling and Performance Analysis of Self-Adaptive Systems"

10.30 Break

11.00 *Jörg Henß*
"OMPCM - An OMNet++ simulator for Palladio"

11.30 Discussion: PCM 3.4 Release

11.45 Open Discussion

Contents

Extending the Palladio Component Model using Profiles and Stereotypes

Max E. Kramer*, Zoya Durdik*, Michael Hauck†, Jörg Henss*, Martin Küster†,
Philipp Merkle* and Andreas Rentschler*
*Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
†FZI Research Center for Information Technology, Karlsruhe, Germany

***Abstract*—Extending metamodels to account for new concerns has a major influence on existing instances, transformations and tools. To minimize the impact on existing artefacts, various techniques for extending a metamodel are available, for example, decorators and annotations. The Palladio Component Model (PCM) is a metamodel for predicting quality of component-based software architectures. It is continuously extended in order to be applicable in originally unexpected domains and settings. Nevertheless, a common extension approach for the PCM and for the tools built on top of it is still missing. In this paper, we propose a lightweight extension approach for the PCM based on profiles and stereotypes to close this gap. Our approach is going to reduce the development effort for new PCM extensions by handling both the definition and use of extensions in a generic way. Due to a strict separation of the PCM, its extension domains, and the connections in between, the approach also increases the interoperability of PCM extensions.**

I. Introduction & Motivation

Domain-specific languages (DSLs), are designed for particular purposes. Their vocabulary is usually restricted to the terms used in their specific domains. On the one hand, this is their biggest strength: for a focused DSL, you can define the semantics more easily. Furthermore, narrow DSLs are easier to learn. On the other hand, being specific and concise is limiting, too. The expressiveness of a DSL might not be sufficient to cover all aspects of a problem that people would like to treat in the future. Additional DSLs need to be defined for the different viewpoints to cover the additional aspects. However, combining existing DSLs is difficult: First, the languages need to be combined on the abstract syntax level, making one DSL dependent on the other. Second, the tools that are generated for the concrete syntaxes need to be integrated. In general, this is a hard task.

A well-established approach to solve the problem of combining DSLs is to define a *core* metamodel with extensions designed as decorator models around it. Having a well-defined core, the conciseness and type-safety of the DSL are not corrupted. It is, however, the responsibility of the author of the extension to provide tool support that integrates the editors for the core DSL and the editors for the extension. The central concept is that the editors for the core model do not know about any decorator and can exist without any extension. Writing, maintaining and using additional editors for the extensions is a repetitive and time-consuming task. Here, we identify the need for a mechanism enabling general extensions of the DSL with a reusable set of tools for the definition of the extensions.

Another way to define extensions are annotations that seek to solve the problem by being very flexible. They allow to add arbitrary information to model elements. These additions must be interpreted at runtime by transformations processing the model. As type-safety is lacking, checking the syntax of any annotation is not possible at development time in EMF-based modeling environments, which is an unfortunate fact.

The abovementioned problems hold for the Palladio Component Model (PCM) [1] [2], too. While being a DSL for performance- and reliability prediction on the architecture level, a variety of additions to the core language have been introduced. Additions include information attached to connectors as performance completions, design patterns that are employed by the system or design decisions that have been made while designing the system structure or the deployment. All these additions have been defined outside the core of the PCM, raising the need for editors and adaptation of tools that process models with additions.

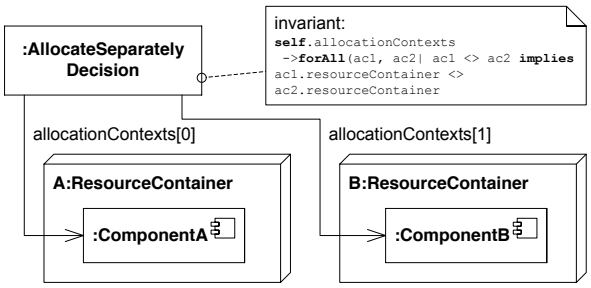

Fig. 1. Illustration of an architectural decision to allocate components separately applied to the deployment of PCM components (without stereotypes)

Fig. 1 exemplifies a design decision decorating a PCM allocation model. The element AllocateSeparatelyDecision indicates that the architect decided to deploy ComponentA and ComponentB on different hardware nodes A and B. The corresponding invariant formalises this decision through OCL.

Of course, the decision model that is attached to the PCM architecture model can be edited separately. The links to the PCM elements can be defined outside any PCM-related editor. But for an architecting process it is somewhat counterintuitive to have different model editors for the same information, which is architecture and architectural decisions. Therefore, we see the need for a generic mechanism that enables additions to the PCM without the need to re-generate any graphical or textual

Fig. 2. UML stereotype example (from [3], p. 663)

editor for PCM elements. The example above illustrates the situation we have in mind for extensions. A domain metamodel (decisions) exists independent of the PCM metamodel. The bridge between the two models, i.e. adding decision-related information to PCM elements, needs to be established by a profile-based extension mechanism.

For a generic extension mechanism, we identified five major requirements: It has to support *type-safe* extensions that can be created in a *non-invasive* way, so that the PCM does not have to be adapted to account for extensions. Furthermore, the mechanism has to be *lightweight* in the sense that extending the PCM has to be easier than creating a complete and separate metamodel. In addition, the mechanism has to be *flexible* and *intuitive* so that it is possible to create small as well as complex extensions using tools and notations that are familiar to many PCM developers. As a last requirement, extensions have to be *composable* so that new extensions can rely on existing ones.

This paper presents a PCM extension approach that addresses these requirements through profiles and stereotypes as known from the Unified Modeling Language (UML) [3]. It takes ideas from decorator models, but brings it to a level where PCM models can be extended without re-generating editors.

The rest of the paper is structured as follows: Section II provides the foundations for our work. The main features and concepts of our approach are presented in detail in Section III. Section IV discusses the technical realization of our extension approach. Four different examples of PCM extensions that would profit from our approach are presented in Section V. The paper is completed with a discussion of related work in Section VI and conclusions in Section VII.

II. Foundations

In this section, we first give an introduction on the profile concept of the UML for extending the UML metamodel. We then explain an approach based on the Eclipse Modeling Framework (EMF) called EMF profiles. This approach adopts the UML profiles concept to DSLs based on EMF and appears to be suitable for providing a lightweight extension mechanism for PCM models.

A. UML Profiles

The UML provides a mechanism called *Profiles* for extending metaclasses to adapt them for different purposes [3]. By using the profile mechanism, user-defined UML extensions called *stereotypes* can be defined that carry specialized semantics. During the last years, several UML profiles have been standardized by the OMG and are included in common UML tools.

Fig. 2, taken from the UML standard [3], shows the UML element Interface extended by a stereotype Remote. With this extension, remote interfaces are metamodelled indicating an interface with specific properties for remote usage. With appropriate tooling, remote interfaces can appear as basic building blocks, such as conventional UML elements. Note that the UML stereotype element inherits from the UML class element. Thus, associations and attributes can be used to further specify the properties of the stereotype.

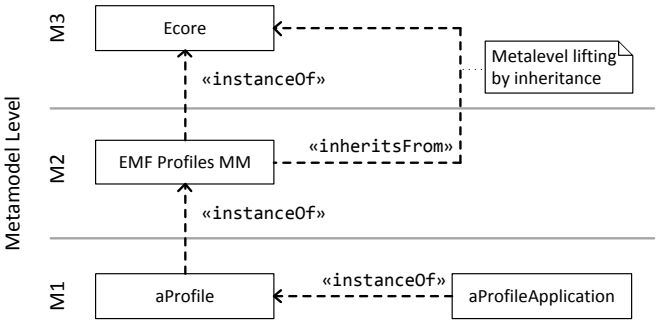

Fig. 3. EMF Profiles architecture (adapted from [5])

The UML *Profile* element is a special UML package used for collecting all UML stereotypes defined for a certain domain or platform (e.g. embedded systems, JEE, etc.).

B. EMF Profiles

To apply the profile mechanism to EMF-based metamodels, and to overcome issues with the UML profiles concept, Langer et al. developed an approach called EMF Profiles [4], [5].

Fig. 3 shows how EMF Profiles extend the metamodelling language Ecore with a profile mechanism. EMF Profiles provides a metamodel for defining profiles consisting of stereotypes. This metamodel inherits from Ecore metamodel elements. Aa profile inherits from Ecore EPackage, a stereotype inherits from EClass. A profile model is then defined on the same layer as a metamodel defined with Ecore (such as the PCM). The instance of a profile, i.e. a profile application, resides on the same layer as an instance of the metamodel.

EMF Profiles comes with a tool that supports the specification of profiles as well as the application of profiles in generated EMF and GMF editors. Besides adopting the concept of profiles from UML, this approach and the corresponding tooling facilitate the extension of EMF models by providing additional functionality. This includes the issue of storing profiles in a separate container (in UML, profile applications are directly included into the UML model), as well as a clean-up function to automatically delete stereotype instances referencing deleted model elements. As a stereotype can be used to extend multiple metaclasses, it provides a powerful mechanism for specifying extensions of EMF-based DSLs.

We decided to adopt EMF Profiles for implementing a lightweight extension mechanism for PCM models which is described in more detail in the following sections.

III. Concept

In this section, we explain the proposed PCM extension approach on a conceptual level. First, we outline the complete

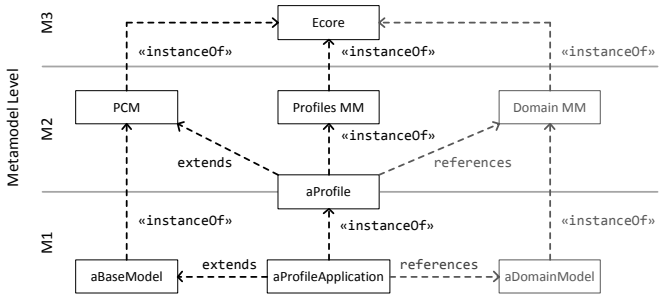

Fig. 4. Models, layers and relations of a profile application extending a PCM instance and referencing a domain model

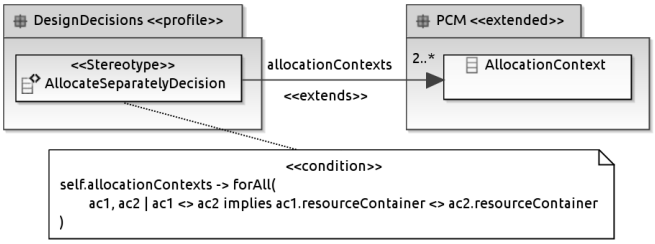

Fig. 5. Example design decision profile for the PCM showing the stereotype AllocateSeparatelyDecision and its context.

approach and present all involved artefacts. Then, we explain in detail how an extension engineer defines an extension profile. Last, we switch to the process of applying an extension profile at runtime. Both extension steps are presented in a generic manner and illustrated using the PCM profile for design decisions as introduced before.

A. Overview

We propose a practical extension mechanism for Palladio to handle PCM extensions in a common way. In our vision developers of the Palladio Bench are enabled to extend the PCM by defining a profile that consists of stereotypes. In order to provide users of Palladio the possibility to use the extension profile, its developer has to register it using an Eclipse extension point. Afterwards users will directly be able to add information to models, because stereotypes provided in a profile can be applied in a generic way. Note that our approach only provides a way to add information to elements of the PCM. It does not address the use of this information nor is it concerned with extension possibilities for other parts of the Palladio Bench.

In Fig. 4 we show how a profile and its application are related to other models. A profile is an instance of the profiles metamodel, which is based on the metamodel presented by Langer et al. [4]. This metamodel conforms to the Ecore metamodelling language used by EMF and the PCM. A profile instance (aProfile) extends the PCM by specifying which specific elements of the PCM can be extended using stereotypes. If necessary, a profile instance can connect instances of its stereotypes to a domain model instance (aDomainModel). It is optional to refer to such an instance of a metamodel representing the extension domain (aDomainMM), but to show full capabilities we depict the corresponding metamodel, model and relations in Fig. 4. When a PCM instance (aBaseModel) is extended, an instance of the profile (aProfileApplication) is created to represent the profile and stereotypes applications. Such a profile instance can refer to other models conforming to the PCM or to models representing the extension domain. We placed the profile application instance on the border of the first two model levels to highlight the two roles it is playing. On the one hand, the profile application is instantiated by profile applications in M1. On the other hand, the profile application itself is an instance of the profile metamodel in M2.

B. Extension Profile Definition

The first step in extending the PCM using our approach is the definition of an extension profile using the graphical profile editor. We are developing this profile editor based on the editor presented by Langer et al. [4]. As shown in Fig. 5, this editor is in turn based on the standard graphical editor for Ecore metamodels that is part of the EMF. For the design decision extension, for example, a developer has to create a profile that contains at least one stereotype for each possible decision type. In Fig. 5 we present how a part of this profile could look like. The profile contains everything that is necessary to enable users to specify the design decision mentioned in the introduction. This design decision states that two or more components have to be allocated to separate resource containers. The stereotype AllocateSeparatelyDecision of the DesignDecisions profile is linked to the metaclass AllocationContext of the PCM. This is done using a special extends relation. It states that this stereotype can only be applied to instances of the metaclass AllocationContext or to instances of its subclasses. Note that these semantics of extends relations between stereotypes and metaclasses are different from inheritance relationships among metaclasses. In our example, the extends relation is specified with a multiplicity of 2..*. Therefore, possible applications are restricted to cases where at least two entities have been selected. An additional restriction for applications of the AllocateSeparatelyDecision stereotype is specified using an OCL expression. This restriction is shown in the graphical profiles editor as an attached note. It states that the stereotype can only be applied to AllocationContexts that correspond to distinct resource containers.

With the profile editor it is also possible to define stereotype attributes and references as well as dependencies between stereotypes. These three possibilities are not used in our small design decision example and therefore only explained generically: Stereotype attributes are graphically specified in the same way metaclass attributes are defined using the graphical ecore editor. Such attributes have to be of primitive type and correspond to tagged values of UML profiles. References to complex-typed elements of additional domain models are graphically defined in the same way metaclass references are defined in EMF. Dependencies between stereotypes can be used to specify that a stereotype s1 can only be applied if another stereotype s2 has already been applied. These dependencies are graphically defined using a special depends on relation. We

introduce this new dependency concept for stereotypes with our approach as it is not part of UML or EMF Profiles.

The second and last step in defining a PCM profile is the registration of the profile. For this registration we use Eclipse's extension mechanism and define an extension point. This extension point requires extensions to provide a profile name and the path to the profile model itself. Every profile registered for the extension point is retrieved automatically by the Palladio Bench during start-up. Based on this, the application of profiles to regular PCM instances and all related tasks can be handled in a generic way as explained in the following subsection.

C. Profile Application

A profile registered for the extension point as described above is automatically available in the tree-based editor and in the graphical editor of Palladio. Whenever a user selects entities in an editor, stereotypes of extensions can be applied if the corresponding conditions are met. For this purpose, a menu is automatically created for each extension based on the information provided by the extension point registration and the profile itself. During runtime it is decided whether a stereotype can be applied with respect to the extends relations and OCL conditions defined in the profile. If a stereotype can be applied, the corresponding menu entry is activated.

During the application of a stereotype, a properties view showing the values of the stereotype application is opened. Using this view the user can enter values for attributes of primitive type. For complex-typed references elements of existing models can be selected. For each model and each applied profile the values for attributes and references of applied stereotypes are stored in a separate profile application file. This is an important difference to UML profiles and one of the reasons why parts of the Palladio bench can ignore profile applications if they are unaffected by them. The described possibility to edit stereotype application values is provided out-of-the-box and available for all profiles. An extension developer can also replace the standard view for editing stereotype applications with a graphical editor. Such a custom editor for stereotype applications could be used to conceal that elements are not part of the PCM.

When performing analysis, simulations or model transformations are executed in Palladio that may request extension information for input models using an API for our generic extension mechanism. The extension information obtained can be kept separate or can be used to create augmented versions of models in a pre-processing step. If a workflow in the Palladio Bench does not explicitly request information for a certain profile it operates as if the profile is not applied. On the other hand, Palladio can be easily extended by different types of analyses, without having to change PCM core functionality (e.g. by using various eclipse extension points or workflow engine extensions). Providing such analyses is out of scope of this paper, but the proposed Palladio extension can be used to facilitate such analysis without changing the PCM core (and the PCM core metamodel).

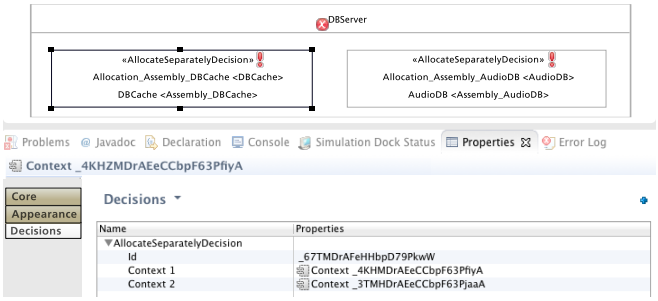

Fig. 6. Graphical editor showing a profile application and a validation error

IV. Implementation

The technical realization of the extension approach presented in this paper is going to reuse existing infrastructure from EMF and EMF Profiles. In this section we outline which steps are necessary to implement our approach as three artefacts: A graphical profile editor, an integration into Palladio's tree-based and graphical editors and an API for persisting and retrieving extension information.

A. Extension Profile Definition

The profiles editor of EMF Profiles provides the basic functionality needed to define the proposed PCM profiles. It is possible to define new stereotypes and to specify to which PCM elements can be applied. Furthermore, primitive-type attributes and complex-typed references can be added to stereotypes. Support for additional OCL constraints restricting the application of stereotypes is missing and has to be implemented for our approach. This could be done by integrating functionality of existing OCL-editors in order to provide syntax-highlighting, auto-completion and syntax checking. To ease the definition of big profiles, support for profile subpackages and inheritance among stereotypes could be added. EMF Profiles currently forces profile developers to choose either a graphical profile editor or a tree-based profile editor. In the future it would be helpful if a single profile can be edited with both editors as it is the case for Ecore metamodels.

B. Profile Application

Currently EMF Profiles provides only restricted tool-support for the application of profiles and stereotypes using existing tree-based and graphical model editors. It is possible to apply stereotypes to a model once a profile has been explicitly loaded by a user. Our idea is to perform an automatic retrieval of registered and applicable profiles based on the current modelling context. This can be done by realizing a profile registry mechanism using an Eclipse extension point. To incorporate support for profiles into the existing PCM graphical editors some additional effort is necessary. Stereotyped entities have to show applied stereotypes in guillemets. Validation of stereotype constraints has to be done while adding and editing stereotype applications. Fig. 6 shows for our design decision example how the integration into a graphical editor and the result of a failed validation may look like.

Setting values for attributes or references and the persistence of stereotype applications is not yet fully supported in EMF Profiles. Therefore, an editor for stereotype applications has to be implemented for our approach. In addition, the type, multiplicity and general OCL constraints defined in registered profiles have to be evaluated in a modified version of the stereotype application hook. As mentioned above, it is currently possible to restrict stereotype applications in the profile editor of EMF Profiles with respect to the type and number of elements to be extended. In the integration for the tree-based and graphical model editor, however, only the type restrictions are evaluated. For our approach, it is crucial that correct multiplicities are ensured. In cases where the lower bound of the multiplicity of an extends relationship is greater than one, it has to be ensured that the corresponding stereotype can only be applied to the desired number of elements. Upper bounds of extends relationships have to be checked similarly. At the moment it is not possible to set the values for attributes and references of stereotype applications in a dedicated view. For our approach, a property view that can be used to enter attribute values and to select referenced elements as described in the previous section has to be implemented. Moreover, a clean-up mechanism has to be introduced to remove dangling stereotype applications caused by model element deletions.

A last important area for the implementation of our extension approach is the persistence, validation and retrieval of stereotype application information. In EMF Profiles stereotype applications have to be explicitly persisted in publicly visible XMI-files. For our approach, stereotype application files have to be automatically validated and stored. Furthermore, these files should be hidden in the package explorer of the Palladio Bench as long as nothing contrary is specified by the extension or by plug-ins depending on it. This can be achieved by using a specific PCM project nature or project type. Last but not least, the implementation of our approach has to provide an API for querying and retrieving the stereotype application information in analyses, simulations and transformations.

V. Example Extensions

A number of examples from existing Palladio decorator models can serve as illustration of both the usefulness and applicability of the EMF profile-based extension mechanism that we introduced so far. The examples come from the performance engineering domain, such as the middleware completion V-C, and the specification of measurements V-D. Others are aligned with the use of PCM as an architecting tool: The recording of design decisions and patterns V-B, and finally the definition of security-related information to PCM elements V-A, which we will discuss in the following.

A. Security

In the EMERGENT project, initial work has been carried out in order to extend Palladio by an analysis of security and privacy attributes of a software architecture [6]. The approach aims at supporting the software architect or Quality of Service (QoS) analyst in specifying such attributes on the architecture level. By analysing the architecture, it helps answering questions related to security and privacy issues, such as “Is it possible for an attacker to get access to a certain kind of data?” A prototypical implementation of the approach has been developed using a decorator metamodel. This metamodel facilitates the specification of security-related annotations to a PCM model. The PCM model and the annotations are then transformed into an analysis model that can be solved by a protocol checker using predefined rules.

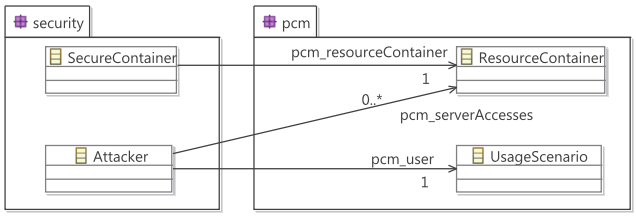

Fig. 7. Security PCM annotations using a decorator model

Fig. 7 shows an excerpt of the security annotation metamodel and the referenced PCM metamodel entities. In the figure, two security-related elements are shown that decorate the PCM metamodel. A SecureContainer can be used to indicate that a PCM ResourceContainer meets certain requirements for secure operation. In addition, the Attacker element can be used to model malicious users in the system. The approach uses the PCM UsageScenario to indicate such users, hence the according reference. For the Attacker element, references to PCM ResourceContainers can be provided to indicate on which servers the user has access to.

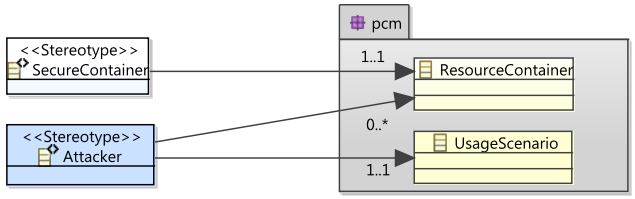

Fig. 8. Security PCM stereotypes using the EMF Profiles approach

Fig. 8 shows how the security extensions can be implemented by using the EMF Profiles approach. The elements SecureContainer and Attacker are defined as stereotypes residing in a security profile. In this example, we only use a subset of the Palladio security extensions. However, the remaining security extension elements can be modelled as stereotypes in a similar way. The example shows that, regarding the metamodel, the existing decorator approach can be replaced by using the EMF Profiles approach. By using common tooling as described in Section IV, modelling security attributes can be included in existing PCM editors. We plan to convert the complete existing security decorator model into a profile model and integrate the profile model into the existing security analysis toolchain, where the model is transformed into an analysis model which

is checked with predefined security rules. If common PCM editors support the profiles mechanism, no custom editor for the security extensions has to be implemented Instead of implementing a custom editor for the security extensions, common PCM editors supporting profiles could be reused, leading to decreased development efforts needed for providing editor support for the security extension.

B. Design Patterns and Rationale

Design patterns are a subclass of design decisions, which can influence one or multiple architectural elements, such as components, provided and required roles, and connectors. In architectural diagrams, design patterns are commonly represented with *pattern roles* and *role connectors* [7]. For example, a Model-View-Controller (MVC) pattern [8] consists of three roles (*Model, View*, and *Controller*), and three connectors between them (*View-Model, Controller-View*, and *Controller-Model*). In the PCM this pattern would be represented through three component instances (*AssemblyContext* in PCM) and three connectors between them (*AssemblyConnector* and *OperationProvidedRole* in PCM). Consequently, the rationale for the existence of these PCM elements would be the decision to apply the MVC pattern. This decision shall be explicitly visible in the model to prevent accidental modification or deletion of the participating components and connectors.

We have implemented a metamodel to capture pattern design decisions for the PCM. An excerpt of it can be seen on Fig. 9 (for full version refer to [9]). The two levels of instantiation required for the design patterns can be seen. The meta-level defines that a pattern consists of roles and connectors between roles. First, the general pattern gets instantiated with a *PatternType* element, for example, MVC or Observer [8]. Such a type instance specifies the number of and the names for the roles and connectors, e.g. Observer pattern type has two roles (*Observant* and *Observee*), and one connector between them.

Second, a modelled system can contain multiple patterns of a certain pattern type. Therefore, a pattern gets instantiated with a *PatternApplication* element referencing the *PatternType*. At this step, pattern roles and role connectors are mapped to the components and component connectors in the PCM model.

Such a metamodel realisation has several drawbacks. First, the two instantiation levels increase the complexity and reduce the comprehensibility of the metamodel. Second, the pattern decoration model references multiple PCM elements, and evolution of the PCM may ripple though multiple elements of the pattern metamodel causing significant maintenance effort of the tool-chain. Finally, when working with the PCM models, which components are annotated with the pattern information, the information on these annotations is not visible unless the related plugins are installed. This might lead to undesired changes, e.g. deletion of a component that was part of a pattern.

These drawbacks of the current implementation could be resolved through the proposed generic PCM profile mechanism. The double instantiation would not be required anymore, and the metamodels will be decoupled, thus allowing for an independent co-evolution of the metamodels. Finally, the annotation information (although in a brief way) would be visible also without additional plugins, thus reducing the risk of unintended design decision violation.

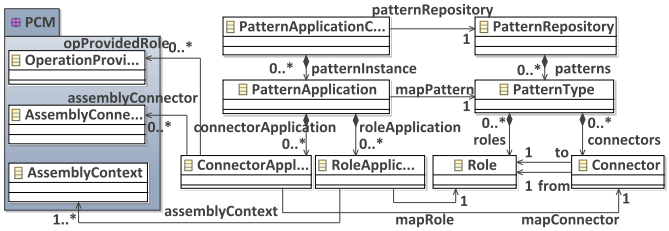

Fig. 9. Excerpt of the pattern metamodel: Decoration of the PCM

C. Performance Completions

Performance completions, initially proposed by Woodside et al. [10], is a technique to programmatically refine an architecture model, to have it better reflect performance influences arising from technological choices. Palladio offers a rich set of performance completions, implemented as in-place model transformations.

Completion transformations can be parameterised via mark models. A mark model is an external configuration model which decorates the architectural main model. For some completions in Palladio, parameter configurations are forming a feature-tree, reflecting the tree-like structure of options and possible combinations thereof. For instance, in order to predict the effect of middleware communication, Palladio provides several middleware-specific completions, each with its own set of configurable options.

As of today, Palladio's *Chilies Project* is one possible way for configuring completion transformations to be run as a preprocessing step prior to performance analysis. In Chilies, each transformation needs to be triggered manually. Elements are annotated outside of the graphical PCM editors, in a tree-based feature configuration editor.

Other completions, like the event-based middleware completion [11], introduce several new concepts by extending the Palladio metamodel with new classes. At the moment, extending PCM is done invasively, and new elements are further hard-coded into the GMF editors.

At the bottom line, current integration of completions is hard to handle because mark models are not standardised in a manner that fosters integration into the editors. Also, completions which extend the core metamodel with new concepts need to modify it, since there is no method to extend the component model and the graphical editors non-invasively.

With a PCM profile mechanism in Palladio, we can define a standardised framework for completion integration, which helps us to solve the aforementioned shortcomings. We are gaining the following benefits: Better integration into the editors, the chance to non-invasively add new concepts to the core, as well as the automatic integration into Palladio's workflow.

A profile mechanism lets Palladio users annotate model elements directly within the editor, in a uniform manner. This can be realised with a stereotype which is capable of

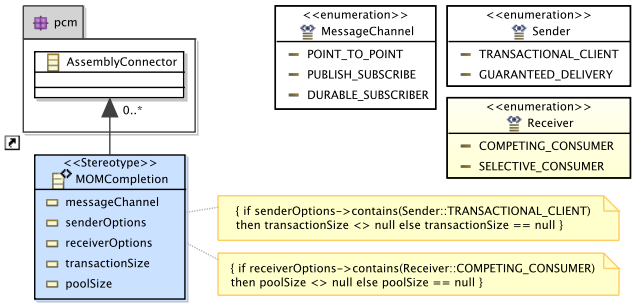

Fig. 10. Connector completion for message-oriented middleware

specialising a PCM element, displaying an icon as a distinct visual feature in the editors, and listing the stereotype's attributes context-sensitively in the properties view. Still, a lot of completions exist, whose parameter space is modelled as a feature tree. In the past, these completions had to be configured in an external feature editor. As an alternative, when registering a new stereotype for these completions, a custom properties view tab can provide an editable view for such feature configurations. With the help of OCL, however, even 3-layered feature trees, as used by the message-oriented middleware (MOM) completion, can be mapped onto a set of tagged values (depicted in Fig. 10).

As a second benefit, Palladio's core models can be extended with further concepts, without polluting the core models. Existing concepts, like, in the case of the event-based middleware completion, events, event groups, event connectors etc., but also future macro concepts, can be factored out of PCM's core model, keeping it clean and concise.

A third benefit is that each completion transformation can be obliged to specify at registration time a list of profiles it processes. This enables Palladio to automatically trigger a completion, depending on the profiles which are found to be applied to a given model. If execution is automatically triggered, manual approaches like that of Chilies would become obsolete.

D. *ProbeSpecification*

Another promising field of application is to attach measuring information to PCM models, which can then be used by the simulation to decide where to take measurements to satisfy the performance analyst's information need. When, for instance, the response time of a system call is of interest, the corresponding EntryLevelSystemCall could be annotated to indicate that the time span between call and return shall be recorded throughout a simulation run. Enabling the performance analyst to define measurement points manually avoids wasting resources due to unwanted measurements and renders highly specific measurement settings possible.

With the idea in mind of specifying measurement points manually, the *ProbeSpecification* metamodel (PS-MM) and a corresponding framework has been developed. While the metamodel allows for expressing measurement points, the framework provides the measuring infrastructure to Palladio simulators. The framework is already employed in SimuCom as

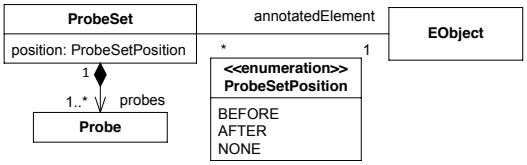

Fig. 11. Current ProbeSpecification metamodel (excerpt)

well as EventSim [12], whereas the metamodel has never been adopted. One of the major reasons is the lack of tool support to handle PS-MM instances. Namely, automatically generated EMF tree editors do not provide the ease of use we desire, which is mainly due to the separate PS-MM editor forcing the modeller to constantly switch between Palladio editors and the PS-MM editor when specifying measurement points. An integration with Palladio's graphical GMF editors does not exist at all and failed so far due to limited development capacities. As a result from the absent PS-MM support by Palladio simulators, measurement points are currently hard-coded in the simulation code, hence bringing along all drawbacks for such a predefined setting as discussed earlier. Adopting EMF profiles, we believe to benefit from a reduced development effort for integrating the PS-MM with Palladio's EMF and GMF editors.

The PS-MM was designed as a decorator model as illustrated in Fig. 11. ProbeSets subsume one or more Probes to form a unit which can then be mounted at a measurement location. While Probes can be seen as devices capable of measuring different quantities (e.g. the current time instant or the queue length of a processor), a ProbeSet can be seen as the application of measuring devices at a specific location (e.g. an EntryLevelSystemCall). Not shown in Fig. 11 is the concept of Calculators, which serve to calculate performance metrics out of measurements produced by Probes. Calculating response times, for example, involves two timestamps representing a service's invocation time and its return time. The reference annotatedElement reveals the decorator nature of the meta-model. Each ProbeSet decorates exactly one EObject, which can be an arbitrary PCM modelling-element, but also any other modelling-element from an arbitrary EMF meta-model.

A major drawback accompanying this design is the missing type safety for ProbeSet applications. That is, ProbeSets may be applied to any instance of an EMF metaclass even if the application makes no sense at all. OCL constraints could mitigate this problem, but introduce additional complexity.

To address these shortcomings, we are planning to refactor the PS-MM towards the presented profiles approach. Since the ProbeSpecification targets not only Palladio, but should also be applicable in similar contexts, *generic profiles* [5]—a part of EMF profiles—seem well-suited and is therefore planned to be supported by Palladio profiles. Generic profiles take into account that profiles are sometimes created with the intention to apply them to a broader range of metamodels. Thus, stereotypes in a generic profile may not refer to concrete metaclasses, but instead extend *generic types*. Each generic type represents a role which needs to be bound to a concrete metaclass once applying the profile to a specific metamodel. This is illustrated

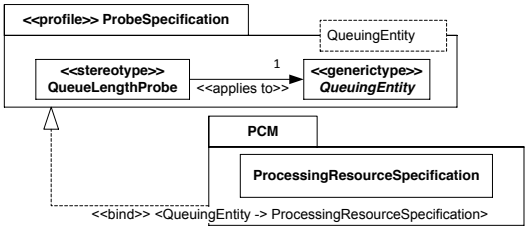

Fig. 12. Generic profiles example

by the example shown in Fig. 12. In the example, the PCM metamodel and the PS-MM are unaware of each other until the binding maps the abstract role of a QueuingEntity to the PCM metaclass ProcessingResourceSpecification.

The challenges of this approach are as follows. First, it has to be decided what PS-MM metaclasses will serve as stereotypes. Deciding for ProbeSets would closely resemble the current metamodel, but bears some disadvantages compared to using Probes, which also seems more natural. Second, although not explicitly foreseen in EMF profiles, the PS-MM will likely need to reference the corresponding profile since Calculators residing in the PS-MM refer to their ProbeSets or Probes. Third, SimuCom's Xpand transformations need to be extended and TraversalListeners are required for EventSim (cf. [12]).

VI. Related Work

The autosar metamodel [13], used in the automotive domain, supports annotations on multiple levels. For each Identifiable entity it is possible to add multi-language textual information as annotations. Each of these annotations is identified by an arbitrary string given the Origin of the information. Furthermore, each autosar PortPrototype can have multiple specialised and typed annotations to provide domain specific information. In the autosar metamodel, annotations are realised with dedicated containment relations.

Another annotation mechanism can be found at the heart of the Ecore metamodel [14]. It allows to add EAnnotations to every instance of EModelElement, i.e. basically any Ecore element can be annotated. An EAnnotation has an identifier and a key value map to store arbitrary values. This mechanism is especially used for storing metamodel documentation. The EAnnotations are stored as containment relations.

In the Service Architecture Meta-Model (SAMM) [15] developed in the Q-ImPrESS project, annotations are used to add QoS specifications, like resource demands, to model elements. Annotations in the SAMM are stored in a detached model using a decorator approach, i.e. each annotation has references to the annotated element. While initial tool support for specifying annotations exists, no tooling is available for adding new annotations. In addition, the implementation is strongly connected to SAMM concepts and thus not reusable.

The Ontology Annotation Metamodel (OAM) [16] was developed in order to derive ontologies from models based on annotated information. This way ontology concepts such as classes or object and data properties are assigned to an existing model. The implementation is based on the Atlas Model Weaver (AMW) and uses a dedicated model based on the AMW mapping model to store link elements that map between annotations and model elements.

VII. Conclusions

In this paper, we presented an approach for extending the PCM in a standardized way. The proposed extension mechanism uses profiles consisting of stereotypes to define extensions for the PCM. As a result, extensions for Palladio can be specified in a uniform format, which makes it possible to handle them generically. In addition to the extension format, we presented a mechanism to register extensions and we introduced a possibility to integrate them into Palladio's graphical editors. The envisioned approach strictly separates base models and extension information. As a result, the Palladio Bench can be unaware of future extensions. In order to provide a solid foundation for our approach, we also discussed practical challenges for an implementation based on EMF Profiles.

Using four existing PCM extensions, we illustrated that current extensions would benefit from a consistent extension mechanism using profiles. The example of a security extension showed that the used decorator approach could easily be replaced by a profile in order to benefit from generic tooling. With an extension for design patterns and related rationale, we demonstrated how extension metamodels can be simplified if the relation to the PCM is factored out into a profile. The third example of performance completions exemplified how our extension approach could render concepts of extension domains more explicit. Finally, an extension example for specifying measurements illustrated a use case for generic profiles, which are supposed to be applicable to a broader range of metamodels.

The approach that we presented in this paper meets the requirements that we identified upfront: its profiles are *type-safe* and *non-invasive* as they can be defined, registered and applied based on the type of existing PCM elements without changing the PCM. The Palladio Bench can be unaware of extensions, because extended model instances appear to the user and the Palladio Bench as ordinary models as long as they do not explicitly ask for extension information. Furthermore, the approach is *lightweight* because it provides specific concepts for stereotype applications so that it becomes unnecessary to explicitly model similar application mechanisms in extension metamodels. In addition, the approach is *flexible* because it supports profiles of different complexity: profiles that add only simple-typed information as well as profiles that refer to extension metamodels or involve application conditions formulated using the Object Constraint Language (OCL). We are convinced that the approach we presented is *intuitive* because profiles are graphically defined using a notation that is adopted from UML profiles. Profiles can be based on other profiles but they can also be applied independently of each other. Therefore, the approach supports *composable* extensions.

In future work, we are going to implement the proposed approach in the Palladio Bench and we plan to evaluate it using some of the example extensions presented in this paper.

ACKNOWLEDGEMENTS

We are grateful to Anne Koziolek and Erik Burger for co-developing this approach and thank Benjamin Klatt, Klaus Krogmann and Qais Noorshams for their comments on drafts. Many thanks go to the anonymous reviewers for their feedback.

REFERENCES

[1] S. Becker, H. Koziolek, and R. Reussner, "The palladio component model for model-driven performance prediction," *Journal of Systems and Software*, vol. 82, no. 1, pp. 3 – 22, 2009.

[2] R. Reussner, S. Becker, E. Burger, J. Happe, M. Hauck, A. Koziolek, H. Koziolek, K. Krogmann, and M. Kuperberg, "The Palladio Component Model," Karlsruhe, Tech. Rep., 2011.

[3] Object Management Group (OMG), "OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1," http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/, August 2011.

[4] P. Langer, K. Wieland, M. Wimmer, and J. Cabot, "From UML Profiles to EMF Profiles and Beyond," in *Objects, Models, Components, Patterns*, ser. Lecture Notes in Computer Science, J. Bishop and A. Vallecillo, Eds. Springer Berlin / Heidelberg, 2011, vol. 6705, pp. 52–67.

[5] ——, "EMF Profiles: A Lightweight Extension Approach for EMF Models," *Journal of Object Technology*, vol. 11, pp. 1–29, 2012.

[6] EMERGENT Project, "Deliverable D.Q1.G1.3.2 / M24: Beschreibung von Anforderungen und Architektur für Qualitätssicherungssysteme in föderierten Cloud-Umgebungen," http://www.software-cluster.com/en/projects/joint-projects/emergent-en, 2012.

[7] M. Elaasar, L. C. Briand, and Y. Labiche, "A Metamodeling Approach to Pattern Specification," in *9th Int. Conf. on Model Driven Eng. Lang. and Syst.*, ser. MoDELS'06, 2006, pp. 484–498.

[8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, Amsterdam, 1995, no. 1.

[9] Z. Durdik and R. Reussner, "Position Paper: Approach for Architectural Design and Modelling with Documented Design Decisions (ADMD3)," in *Proc. of the 8th Int. ACM SIGSOFT Conf. on Quality of Software Architectures*, ser. QoSA '12. New York, NY, USA: ACM, 2012, pp. 49–54.

[10] M. Woodside, D. Petriu, and K. Siddiqui, "Performance-related Completions for Software Specifications," in *Int. Conf. on Software Engineering (ICSE)*, 2002.

[11] B. Klatt, C. Rathfelder, and S. Kounev, "Integration of event-based communication in the palladio software quality prediction framework," in *Proceedings of the joint ACM SIGSOFT conference – QoSA and ACM SIGSOFT symposium – ISARCS on Quality of software architectures – QoSA and architecting critical systems – ISARCS (QoSA-ISARCS 2011)*. New York, NY, USA: ACM, 2011, pp. 43–52.

[12] P. Merkle and J. Henss, "EventSim – an event-driven Palladio software architecture simulator," in *Palladio Days 2011 Proceedings*, ser. Karlsruhe Reports in Informatics ; 2011,32, S. Becker, J. Happe, and R. Reussner, Eds. Karlsruhe: KIT, Fakultät für Informatik, 2011, pp. 15–22.

[13] AUTomotive Open System ARchitecture (AUTOSAR), "AUTOSAR Software Component Template," October 2011.

[14] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework*, 2nd ed., ser. Eclipse series. Addison-Wesley Longman, Amsterdam, Dec. 2008.

[15] Q-ImPrESS Project Consortium, "Project Deliverable D3.1 Prediction Model Specification," http://www.q-impress.eu/wordpress/wp-content/uploads/2009/05/D3.1-Prediction_model_specification_v20.pdf, 2009.

[16] G. Hillairet, "AMW Use Case - Metamodel Annotation for Ontology Design," http://www.eclipse.org/gmt/amw/usecases/oamusecase/.

Extending Palladio by Business Process Simulation Concepts

Robert Heinrich, Barbara Paech
Institute of Computer Science
University of Heidelberg, Germany
{heinrich, paech}@informatik.uni-heidelberg.de

Jörg Henss
Institute for Programme Structures
and Data Organisation
Karlsruhe Institute of Technology, Germany
joerg.henss@kit.edu

***Abstract*—Business process design and enterprise information system (IT system) design are often not well aligned, which leads to problems at runtime caused by neglecting the mutual impact of business processes and IT systems. Simulation is a promising approach to support the alignment of business process design and IT system design by impact prediction. Currently, the Palladio approach does not include business process modeling and simulation functionality, which impairs the accuracy of simulation results. We propose an extension to the Palladio Component Model (PCM) and the simulation behavior in order to enable the modeling of business processes and the simulation of the mutual impact of business processes and IT systems using the existing Palladio tool chain.**

***Index Terms*—Business Process; Enterprise Information System; Simulation; Alignment; Performance.**

I. INTRODUCTION

Business processes and IT systems mutually impact their performance in non-trivial ways. Nonetheless, business process design and IT system design are often not well aligned. Missing alignment of business process design and IT system design results in problems at runtime such as large IT response times, large process execution times, overloaded IT systems or interrupted processes. Simulation is a powerful approach to predict the impact of a business process design on the performance of supporting IT systems and vice versa. Based on the predicted impact, business process design and IT system design can be adapted to enable alignment.

There are several business process simulation approaches and several IT system simulation approaches. In the business process domain, simulation is commonly used to predict business process performance and financial impact (e.g. [1]). In the IT domain, computer network simulation is widely spread for decades to estimate performance of network topologies. Moreover, there are approaches for software architecture simulation on component level (e.g. [2]) and service-oriented architecture simulation (e.g. [3]). Focused on one domain, these approaches are adequate to predict the performance of a business process or an IT system isolated of each other. However, in current simulation approaches, there is little integration between the business process domain and the IT domain.

Considering business processes along with IT systems in simulation can support several roles in the joint development of business processes and IT systems.

- Requirement engineers can check in design phase whether an IT requirement can be met by a proposed IT system design for a given business process design.
- System developers can compare design alternatives of IT systems invoked in a given process to each other without implementing prototypes.
- Hardware administrators can check the utilization of IT resources such as CPU or hard disk drive for a proposed IT system design and a given business process design.
- Business analysts can check in design phase whether a process requirement can be met by a proposed business process design and a given IT system design.
- Process designers can compare business process design alternatives to each other without executing a business process in practice. Thereby, the impact of the given IT system(s) on the process is considered.

The Palladio approach [2] currently does not provide business process modeling and simulation functionality. This impairs the accuracy of IT simulation results since mutual impact, e.g. on workload distribution, is not correctly considered. Betz et al. [4] sketch a framework to integrate the life-cycles of business processes and IT systems based on Palladio. This framework uses IT simulation and business process simulation isolated of each other. However, isolated simulations are not able to consider the mutual impact of business processes and IT systems addressed in this paper.

If we abstract from the different semantics of business process simulation and IT simulation, there are several analogies. Both kinds of simulations...

- ...can be built upon queuing networks (queuing theory [5])
- ...simulate the utilization of resources (human actor resources or IT resources). An actor resource is the representation of a human actor in the process. Actor resources as well as IT resources process jobs from their waiting queue in a certain processing rate. See [1] for an example of actor resources and their waiting queues in business process simulation.
- ...use a specification of a workflow of actions to be processed by the resources. In business process simulation besides system steps also actor steps are considered.
- ...use actions that can be composed hierarchically.

- ...use a specification of workload. In process simulation workload is often specified in the form of inter-arrival times which is comparable to the open workload in Palladio.
- ...acquire and release shared passive resources. In analogy to passive resources in Palladio, passive resources in a business process are devices or machines which are available in limited capacity and are required to perform the process but do not actively process it (cf. [1]).

Considering these analogies, Palladio seems to be an adequate foundation to be extended by business process simulation concepts to enable an integrated process and IT simulation for performance prediction purposes.

In this paper, we propose extensions of the PCM and the simulation behavior in order to enable the simulation of the mutual impact of business processes and IT systems. The paper is structured as follows: In Section II, we introduce definitions required for understanding the following sections. We present an example process and discuss the mutual impact of business processes and IT systems in Section III. The need for an integrated business process and IT simulation is presented in Section IV by pointing out open issues in Palladio. Requirements on an integrated simulation are listed in Section V. In Section VI, we show how to extend the PCM and the simulation behavior by business process concepts to enable an integrated simulation. An example of the behavior of the proposed extensions to the simulation is shown in Section VII. Section VIII concludes the paper and presents future work.

II. Definitions

A *business process* is a "set of one or more linked activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships" [6]. Each activity within the process consists of a set of one or more linked steps. Steps are either completely performed by a human actor – called *actor step*s – or completely performed by an IT system – called *system step*s. In the PCM, a system step is represented by an EntryLevelSystemCall [2]. Yet, to keep things generally intelligible, we use the term system step in this paper.

A human actor performs actor steps lined up in his/her *worklist*. A worklist is comparable to a waiting queue of an IT resource (cf. [2]) which lines up jobs to be processed by the IT resource. Business processes are typically specified on several levels of abstraction which are composed hierarchically. Processes consist of subprocesses, subprocesses consist of activities, and activities consist of actor steps and system steps. Hierarchical composition is required to keep track of large processes. In this paper, we focus on the level of steps as detailed actor steps and system steps are required to determine the mutual impact of business processes and IT systems in simulation. The mutual impact is discussed in detail in Section III. A *business process instance* is the "representation of a single enactment of a process" [6]. An *IT system instance* is an executable representation of the IT system design. A business process design P and an IT system design S are *aligned*, if

- System steps of S are invoked in P, and

for all the process instances P' of P and all the system instances S' of S holds:

- S' meets the requirements of S when used in P', and
- P' meets the requirements of P when uses S'.

Workload is "the amount of work to be done" [7]. Business process workload determines the amount of process instances that traverse the business process. Often workload is measured in process instances per time unit. Process instances traverse all the actor steps and system steps on a certain path of the process from the process start point to a process end point. *Response time* is the total time required by a process instance to traverse a system step. *Execution time* is the total time required by a process instance to traverse an actor step. The time required to traverse an activity within the process or an entire process is called execution time, too. *Distance* refers to the difference in time in which the process instances reach a certain point in the process. The distance between two process instances begin the execution of the process is called *inter-arrival time*. *Workload distribution* refers to the distance between process instances within the process. For example, three process instances can occur in a constant distance (30 seconds) to each other, or they can occur in *bursts*, e.g. all three process instances occur directly after the other or even at the same time. In all cases, the workload is three process instances in one minute. While traversing the process, the distance between process instances can vary. Bursts of process instances or gaps between the process instances can be created.

III. Mutual Impact of Business Processes and IT Systems

Business processes and involved IT systems mutually impact each other in several ways. In this paper, we focus on performance impact. In the following, we introduce an example and discuss the mutual impact of business processes and IT systems based on the example.

A. The Process of Order Picking

Suppose the process of order picking in the store of a manufacturer. Goods requested in an order are taken from the store and are packed to be transported by trucks. The process is illustrated in Figure 1. The example process is a simplified representation of a process from practice we are currently analyzing in a case study.

The process has a start event and an end event, represented by circles. Steps are represented by rectangles with rounded corners. "AS:" is used to mark actor steps. "IT:" is used to mark system steps. Arrows represent the control flow in the process. Lanes represent roles of human actors. For each role, several human actors are available, i.e. orders can be processed concurrently.

The shift leader first gathers the data from the order for picking. The IT system inserts the data into a database and

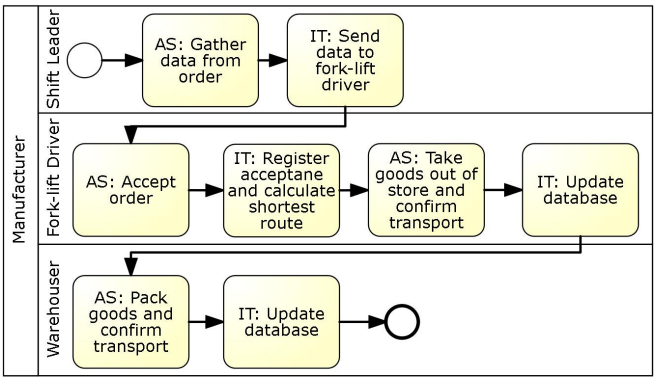

Fig. 1. Example Process

sends the order data from the database to a mobile client of the fork–lift driver. The fork–lift driver accepts the order, which is registered in the database by the IT system. The IT system then calculates the shortest route to the goods in the store of the manufacturer. After the route has been calculated, the fork–lift driver takes the goods out of the store and put them on a location where they are packed for transport. Then, s/he confirms the transport. The IT system updates the database and informs the warehouser. Then, the warehouser packs the goods for transport, put them on a location where they are picked up by a truck later, and confirms the transport. Finally, the IT system updates the database.

There are strict time constraints in the process, since requested orders have to be available for transport at the time the trucks arrive. Since delays are very expensive in the logistics business, it is a time-critical process.

B. Process Impact on IT System Performance

IT system performance is determined by the business process design as well as the business process workload. The business process design determines which system steps are invoked in the process, when a specific system step is invoked and which system steps are invoked concurrently. For example, suppose a business process that includes two system steps. IT system performance may differ depending on whether the steps are invoked sequentially or concurrently. As defined in Section II, business process workload determines the amount of process instances that traverse the process. Process instances traverse all the actor steps and system steps on a certain path of the process from the process start point to a process end point. Thus, business process workload determines how often an IT system is invoked. IT system performance may differ depending on whether the system is invoked once per second or 100 times per second.

C. IT System Impact on Process Performance

IT system performance impacts the business process performance in two ways. Firstly, if the IT system is overloaded because too many actors invoke the system, it is no longer available for actors in the business process. Thus, the execution of the business process is impeded or even interrupted. For example, if the IT system cannot send the order data to the mobile client (e.g. as it is overloaded by too many actors), the goods may not be available for the truck in time. Secondly, the response time of system steps may impact the business process performance if its extent is comparable to the extent of execution time of actor steps within the process. Thus, IT system response time may significantly increase the execution time of the entire business process or single activities within the process. Response time and execution time is defined in Section II. For example, in our case study, the transmission of order data to the mobile client of the fork–lift driver can last up to 40 minutes and more which heavily impacts the process execution time as it increases accordingly.

D. Mutual Impact of Actor Steps and System Steps on Workload Distribution

IT system performance and business process performance are influenced by workload distribution within the process. See Section II for a definition of workload distribution. According to Mi et al. [8], workload distribution has “paramount importance for queueing prediction, both in terms of response time mean and tail” as it impacts performance significantly. Unequal distributions of workload (“burstiness” factor) often lead to increasing response times. As human actors process jobs on their worklist in a similar manner as IT resources, e.g. following the FIFO (First In First Out) principle, it is logically comprehensible that workload distribution impacts process performance the same way as it impacts IT performance.

Workload distribution in the process is influenced by actor steps as well as system steps assuming synchronous scheme of communication. In this paper, we discuss synchronous scheme of communication. Assuming asynchronous scheme of communication (i.e. actors do not wait for system steps to finish), workload distribution is only influenced by actor steps. If an actor is already busy when an actor step should be performed by this actor, the execution of the actor step has to wait until the actor is ready to perform the actor step. If an IT resource used in a system step is already busy when it is invoked by an actor request, the request has to wait until the resource is ready to process the request. Process instances may also have to wait for shared passive resources to be released. As mentioned above, there are shared passive resources in business processes as well as in IT systems. Waiting times hinder the flow of the process instances through the business process. For each step, waiting times may differ from process instance to process instance. Thus, the distances between the process instances in the process may vary during process execution. The example in Table I shows how the distance between process instances is decreased which may result in a burst.

Suppose, there are two actors A1 and A2 of the role fork–lift driver. The queue of A1 has a length of five time units at t_0. The queue of A2 has a length of six time units at t_0. This represents the processing time of the remaining work to be done by the actors. There are two process instances I1 and I2 which reach the actor step “AS: Accept order” at a distance

Time	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
Actor A1	5 (+3 from I1)	4 (+3)	3 (+3)	2 (+3)	1 (+3)	I1	I1	I1	–	–
Actor A2	6	5	4 (+3 from I2)	3 (+3)	2 (+3)	1 (+3)	I2	I2	I2	–

TABLE I
EXAMPLE: DECREASING THE DISTANCE

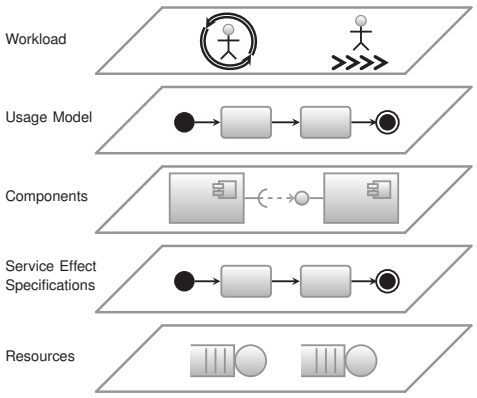

Fig. 2. Simulation Layers Used in the PCM

of two time units. Both instances put a demand of three time units on the actors. At the point in time t_0 the process instance I1 is allocated to the actor A1, because the queue of A1 is the shortest. At the point in time t_2 the process instance I2 is allocated to actor A2, because its waiting queue is the shortest, now (see Table). The processing of I1 is finished at the point in time t_7. The processing of I2 is finished at the point in time t_8. The distance between both process instances has decreased to one time unit. If A1 and A2 both had a waiting queue of length five time units at t_0, the processing of both process instances would have finished at the same time.

In this case, I1 and I2 reach the system step "IT: Calculate shortest route" at the same time which, according to Mi et al. [8], may results in a higher mean response time than if the process instances reach the system step at a distance of two time units.

IV. OPEN ISSUES IN PALLADIO

Considering the mutual impact of processes and IT systems described above, the following open issues currently appear when using Palladio and a business process simulation isolated from each other to simulate IT systems used in a business process.

O1: In the PCM usage model (cf. Figure 2) actor steps are considered as "black boxes" representing a delay. The execution time of actor steps (i.e. the extent of the delay) is not determined in simulation but has to be specified before simulation as an input. Thus, execution times represent assumptions made before simulation. The extent of the execution time of an actor step depends on the utilization of the corresponding actor which is unknown before simulation.

Palladio requires a specification of the usage of the IT system in the form of one or more usage scenarios. Using process simulation and IT simulation isolated of each other requires deriving an IT usage profile (i.e. a usage scenario) for IT simulation from the business process specification. The following open issues appear while deriving an IT usage profile:

O2: Variation of process arrival distribution during simulation cannot be easily mapped to a time-invariant IT workload specification as it is assumed in the PCM usage scenario. The execution of a business process typically lasts several hours or even days. Thus, there may be changes in the workload during simulation. For example, from 9:00 am to 11:30 am the process is triggered 100 times per minute, and from 12:30 pm to 5:00 pm, the process is triggered 50 times per minute. From 11:30 am to 12:30 pm, the process is not triggered at all, due to a lunch break. Figure 3 shows an example for time-variant workload on a resource and the resulting queue length assuming the resource has a processing rate of 53 requests per minute.

As shown in the example, inter-arrival time changes over time. The PCM assumes a time-invariant workload which can have a probabilistic distribution. Even so, changing arrival distributions cannot be mapped to a time-invariant workload without approximation. Approximation usually results in reduced accuracy of the predicted IT response times.

Moreover, deriving an IT usage profile includes the specification of actor steps and their execution time which is unknown at this stage (see O1).

O3: Although Palladio is already able to simulate system steps within the PCM usage model, systems steps cannot be included in the simulation of business process scenarios as the usage model currently does not provide business process model elements such as actor steps and existing simulators do not reflect business process simulation behavior such as simulating the utilization of actors.

O4: Workload distribution is only influenced by IT system steps. The impact of actor steps on workload distribution is neglected in current Palladio simulations. There may be some workarounds to manipulate workload distributions, e.g. by modeling delays as stochastic expressions. However, this is not an adequate way as it does not result from simulation but has to be specified before simulation. Thus, workload distributions are not correctly represented in current Palladio simulations. As a result, performance may not be predicted accurately (cf. [8]).

O5: Currently, the system steps contained in a PCM usage model are stochastically independent in terms of their parameters. Parametric dependencies are also possible for actor steps. Probabilistic parametric dependencies in the process impact on

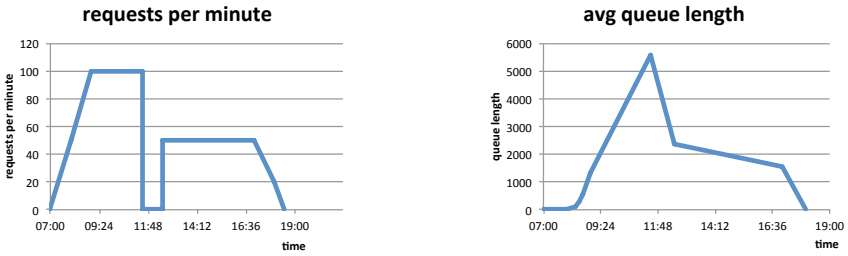

Fig. 3. Example: Time Dependent Workload on a Resource and Resulting Queue Length

workload as parameters such as the number of goods included in an order may increase the number of loop iterations required to handle the order. For example, when an order arrives at the IT system, it is first processed by the IT system optimizing the pickup sequence. Afterwards, the fork-lift driver has to pickup all the goods from the store. Therefore, not only the processing time of the order in the IT system is dependent on the number of goods included in the order, but also the actor step of picking up the goods. The more goods are ordered, the higher is the time required to process the steps.

V. Requirements on Palladio

In order to adequately represent the mutual impact of business processes and IT systems in simulation, Palladio has to fulfill the following requirements.

R1: In IT performance prediction the execution time of actor steps are determined by simulation instead of using assumptions. This requirement addresses O1.

R2: IT resources are demanded directly from the process simulation without deriving an IT usage profile. Thus, IT resource utilization directly results from the process model and the process workload without any approximation (see Section III-B). This requirement ensures that the process model and process workload specification correctly impacts on IT performance and thus addresses O2.

R3: In process performance prediction the response time of system steps are determined by simulation instead of using assumptions and system steps are considered as a factor of process performance. This requirement addresses O3 and is required to ensure that the impact of IT performance on process performance is adequately represented as described in Section III-C.

R4: In simulation the workload distribution within the process model is influenced by the utilization of actor waiting queues within actor steps and the utilization of IT resource waiting queues within system steps as described in Section III-D. This requirement addresses O4.

R5: Probabilistic parametric dependencies of actor steps and system steps within the process are considered in simulation. This requirement addresses O5.

VI. Extending Palladio by Business Process Simulation Concepts

In this section, we describe how to extend Palladio by process simulation concepts in order to meet the requirements introduced above. The proposed extensions include extensions to the PCM as well as extensions to the behavior of the simulation which are discussed in the following.

Figure 4 gives an overview of the proposed extensions on several layers. Extensions are colored blue in the figure. In the following sections we refer to certain layers of the figure to explain it in detail.

A. Extension of the Palladio Component Model

a) Process model: The PCM usage model is extended by the model element ActorStep in order to model steps performed by a human actor. Figure 5 shows the extension. ActorSteps are visualized as rectangles with rounded corners and a stickman icon in Figure 4. For each ActorStep the processing time has to be specified which is described in detail in the following. Moreover, delays commonly used in business process specifications (cf. [1]) can be documented and included in simulation. Idle time is the delay between the possible start and the actual start of the execution of the actor step. Resting time is the delay between the completion of an actor step and the start of the following step respectively the start of a transport. Transport time is the delay required to transport objects. Moreover, the model element Activity is added to enable the modeling of sub processes. The extension of the PCM usage model to a process model contribute to R1 as the modeling of actor steps is required to include them in simulation. It also contributes to R3 as now a process model is available for process performance prediction which also includes system steps. Moreover, the extension contributes to R2 as system steps are now triggered directly from the process model without deriving a usage profile.

b) Simulated resources: The PCM is extended by the organization environment model which is the counterpart of the hardware environment model. Figure 6 shows the organization environment model. It represents the organizational context of the process and contains basically three types of elements – ActorResources, Roles and DeviceResources. An ActorResource represents a human actor. ActorResources are visualized as circles around a stickman icon in Figure 4.

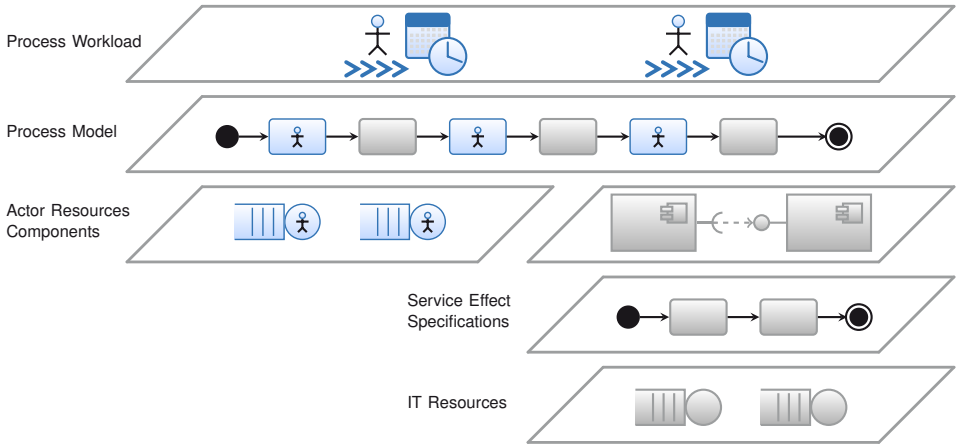

Fig. 4. Extended Simulation Layers

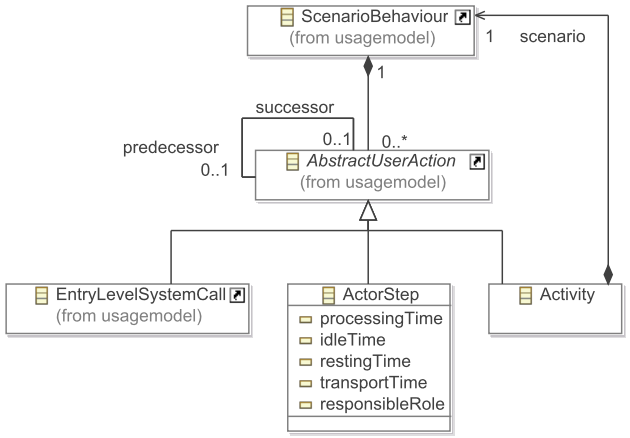

Fig. 5. Extension of the PCM Usage Model

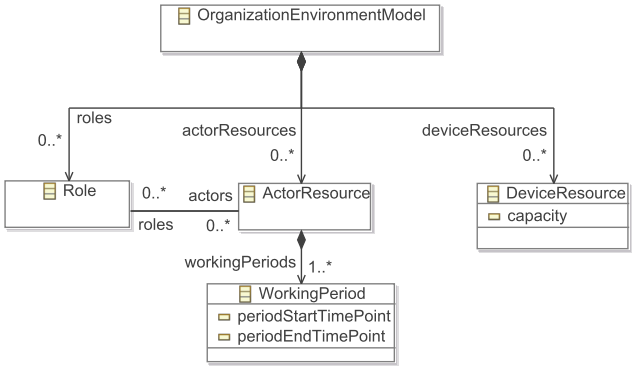

Fig. 6. Organization Environment Model

In analogy to the ProcessingResource, which represent IT resources in Palladio [2], each ActorResource has a waiting queue in which the actor steps (i.e. the jobs) to be done by the ActorResource are lined up. In contrast to ProcessingResources, ActorResources are not continuously available, e.g. human actors have to take a lunch break or to sleep at night. Thus, the availability of each ActorResource has to be specified in one or more WorkingPeriods. Each WorkingPeriod is specified by a start time and an end time. Each ActorResource can have one or more Role(s) which group several ActorResources that have the same properties.

A DeviceResource is a device or machine which is required to perform an actor step of the process but does not actively process the step. Thus, it is called a passive resource. DeviceResources are shared by process instances and are available in a limited capacity. A process instance can acquire a DeviceResource required to perform one or more actor step(s) and release again when the actor step(s) are finished.

The extension of the simulated resources contributes to R1.

c) Modeling of demand on ActorResource: Actor steps specify resource demands in terms of their processing time as shown in Figure 5. Processing time is the time a human actor spends actively processing an actor step. The processing time is specified as a number of abstract time units. In contrast to resource demands on IT resources, work of human actors does not have a measuring unit such as CPU cycles or byte. So we decided to use the time required to perform a step as measuring unit. In contrast to system steps there is only one resource demand per actor step possible. Moreover, there is only one type of resource demand – demand on ActorResources. In contrast to ProcessingResources in Palladio, it is not directly specified which ActorResource performs a certain actor step but in each actor step it is specified which role is required to perform the step. The ActorResource that performs the step is selected dynamically in simulation. From the ActorResources that are available at the current point in simulation time, the ActorResource that has the shortest waiting queue in terms of the sum of processing time of the actor steps in the queue will be selected to perform the actor step. Dynamic selection of actors in simulation is common in process simulation (cf. [1]).

This extension contributes to R1.

d) Modeling of process workload: Besides the simulated resources and resource demands also the workload to be processed by the resources has to be specified for simulation. In Figure 4, workload is represented on the upper layer. In PCM, to each usage scenario within the usage model a workload driver is associated. Currently, there are two types of workload drivers in the PCM – closed workload and open workload. In a closed workload a population of *n* users execute the scenario concurrently [2]. In an open workload users enter the scenario at a specific arrival distribution [2]. Both workload drivers do not support changing process arrival distributions as described in Section IV (O2). We derived a new ProcessWorkload driver from the OpenWorkload driver as visualized in Figure 7. The workload of the process results from the arrival distributions of the ProcessTriggerPeriods associated to the ProcessWorkload driver. A ProcessTriggerPeriod specifies an interval of simulation time in which process instances start the execution of the process in simulation. Outside of the interval there is no start of process instances possible. Each ProcessTriggerPeriod consists of a start time point, an end time point and an inter-arrival time specification. Time designation in the ProcessTriggerPeriod follows the common date and time format which is more readable for human modelers than number of abstract time units usually used in Palladio. However, as Palladio only supports abstract time units, time designations are converted into abstract time units before simulation.

The workload on the IT system is a consequence of the process workload as system steps are invoked by the process instances. The introduction of the ProcessWorkload driver contributes to R2.

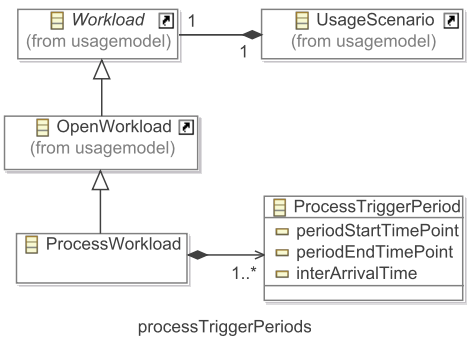

Fig. 7. Process Workload Driver

B. Extension of the Behavior of the Simulation

e) Simulation of process instance arrival: The simulation continuously generates process instances that traverse the process model. The start point of the first instance is the start time of the first ProcessTriggerPeriod of the process. Then, the distance to the start point of the next instance is generated randomly, based on the arrival distribution of the current ProcessTriggerPeriod allocated to the process, and added to the last start point. The next instance starts traversing the process model at that point in time. Instance start points will be generated and instances start traversing the process model until the generated start point of a process instance exceeds the end time of the last ProcessTriggerPeriod. This extension contributes to R2 as the simulation is now able to consider time-variant process workload.

f) Simulation of execution time: If a process instance reaches an actor step within the usage model, the actor step is put as a job into the waiting queue of an ActorResource allocated via his/her role to the actor step. The specific ActorResource is selected based on the length of its waiting queue and whether it is currently available (i.e. the current simulation time lies within a WorkingPeriod). Each ActorResource processes the actor steps in its waiting queue, e.g. following the FIFO principle. For actor steps, the processing time is already specified as resource demand. The waiting time is determined in simulation by waiting in the queue of the ActorResource. The resulting execution time of an actor step is the sum of its processing time and its waiting time at the corresponding ActorResource. This extension determines the execution time of actor steps in simulation and thus contributes to R1. The behavior of DeviceResource in simulation is comparable to the behavior of passive IT resources already contained in Palladio. They can be acquired and released again. Thus, the simulation of DeviceResource is not described in detail.

g) Simulation of workload distribution: In order to fulfill R4, simulation has to consider the mutual impact on workload distribution as described Section III-D. This is represented by the waiting queues of ActorResources and ProcessingResources in simulation. If a job is put in a waiting queue as the resource is busy at the time the job arrives at the resource, the flow of the process instance is hindered. For both kinds of steps, waiting times may vary from instance to instance. As a result, the distance between instances in the process model may change. Thus, workload distribution is manipulated by actor steps as well as system steps in the simulation.

h) Simulation of parametric dependencies: Currently, probabilistic parametric dependencies are limited to basic definitions in the usage model layer, i.e. it is only possible to define variable characterizations for input parameters. This is a good staring point to fully include parametric dependencies of actor steps as well as system steps to fulfill requirement R5. We can built upon the parametric dependency implementation already contained in EventSim [9] for simulating dependencies of Service Effect Specifications (SEFFs).

R3 is addressed as a consequence of the realization of the other requirements as system steps are already contained in the PCM usage model and are included in the simulation. As the PCM and the simulation behavior are extended by process simulation concepts, system steps are included in the process simulation.

C. Implementation-related Considerations

Extending the simulation behavior requires to extent the implementation of an existing simulator as well. Currently, there are two specialized software architecture simulators available for the PCM, SimuCom [2] and EventSim [9]. Furthermore, there exist transformations for translating PCM instances to the Layered Queueing Networks (LQNs) [10] and the Queueing Petri Nets (QPNs) [11] formalisms. The transformations and corresponding formalisms were successfully used in [12] and [13] to simulate complex Palladio models.

We decided to build upon the new event-driven simulator EventSim as the traversal strategy concept implemented in EventSim allows for a simpler extension of the simulation behavior as we expect it for SimuCom. Some of the existing traversal strategies can be reused or easily adapted for the new metamodel elements. Furthermore, using EventSim we can built upon the parametric dependency implementation.

Moreover, especially for high degrees of parallelism and concurrency, EventSim can perform simulations faster than SimuCom (cf. [9]). This is important, as process models usually have many process instances working concurrently. In addition, simulation speed is especially important in the business process context where simulated time is typically much longer than in the IT context. Simulated time in the business process context often spans months or even an entire year.

We decided against the LQN and QPN simulation approaches as these would require the development of additional model transformations to support the newly introduced elements and behavior.

D. Simulation-related Considerations

The different granularities of events in terms of their duration may limit the feasibility of the simulation. In cases where many events happen during a short time frame (e.g. a second) simulating a year may take a long time. Nevertheless, in order to get statistically significant results with workloads that vary over the day, many days have to be simulated. Also, if the actor steps may last several minutes, one needs to simulate longer. Thus, the combination of short running demands (milliseconds for IT events) on the one hand and long running demands (minutes for actor steps) as well as different time intervals (e.g. working time or breaks) on the other hand may cause large simulation times, as fine-grained simulation (which takes long per simulated minute) is required but also a long simulated time frame (e.g. a year) is needed. Especially the time and number of replications required for reaching a given confidence, when using a transient analysis, are a limiting factor. Thus, smart simulation strategies will be useful to circumvent these problems.

In our research, we focus on IT response times that may impact the business process performance as the extent of response time is comparable to the execution time of actor steps. Thus, we do not necessarily need to consider cases where the IT simulation has a large number of fine-grained events per second. Rough estimates of these events seem to be sufficient in the business process context. In the future, we want to analyze the feasibility of the integrated simulation based on several examples. If necessary, we want to investigate strategies to consider fine-grained IT events in a feasible way while simulating a long time frame. One possible strategy is to perform isolated fine-grained IT simulations prior to the integrated simulation for a set of representative classes of workload and workload distribution. During simulation the response time is then determined by looking up results from an equivalent class. Furthermore, an iterative refinement approach can be used to simulate critical time spans in more detail or add missing classes.

VII. Example of the simulation behavior

In this section, we continue the example introduced in Section III-A and sketch how the integrated simulation works.

Suppose there are two ProcessTriggerPeriods in the order picking process per working day. It is common that ProcessTriggerPeriods repeat for example per day or per week. From 8:00 am to 1:00 pm orders arrive and process instances start the execution of the process in a certain distribution. From 2:00 pm to 6:00 pm orders arrive and process instances start the execution in another distribution.

The simulation starts at 0:00 am of the first simulation day. The waiting queues of the ActorResources and the waiting queues of ProcessingResources are empty until the first process instance starts the execution of the process at 8:00 am of the first simulation day. At 8:00 am the first actor step “AS: gather data from order” is put in the waiting queue of an ActorResource that own the role shift leader. As at that point in simulation the waiting queues of all the ActorResources have the same length – they are all empty – the first ActorResource on a list of ActorResources that own the role shift leader is selected. For each ActorResource several working periods are defined. For example, from 8:00 am to 12:30 pm and from 1:00 pm to 6:00 pm. Between both periods there is a lunch break. In the course of simulation, actor steps line up in the waiting queues of ActorResources and internal actions of system steps line up in the waiting queues of ProcessingResources. Each ActorResource processes the actor steps in its waiting queue following the FIFO principle as long as the current simulation time is located within one of the ActorResource’s WorkingPeriods. If the current simulation time exceeds a WorkingPeriod of the ActorResource, e.g. the current simulation time exceeds 12:30 pm, the ActorResource interrupts the processing of the actor steps until the current simulation time is again located in a WorkingPeriod. In the time between WorkingPeriods the process instances stuck in the waiting queue of the ActorResource and cannot reach another waiting queue, e.g. of a ProcessingResource. Being stuck in a waiting queue increases the waiting time for the corresponding instances. As ProcessingResources keep on processing jobs from their waiting queue during lunch break, waiting queues of ProcessingResources empty during lunch break and refill again in the afternoon as then another WorkingPeriod starts.

In the example process, the warehouser requires a fork-lift to put the goods on a location where they are picked up by a truck. Fork–lifts are shared DeviceResources which are available in a limited capacity. If all the fork–lifts are acquired at the moment, the warehouser has to wait until a fork-lift is released. Thus, in simulation the flow of process instances stuck. Waiting time is caused. The next system step "IT: Update database" is not reached by the process instance before a DeviceResource is released and acquired by the process instance. Waiting queues, e.g. of ProcessingResources demanded by the system step "IT: Update database", empty in meantime.

If a process instance is waiting for a passive IT resource, the flow of process instances stuck, too. Waiting time is caused which can impact the process performance, if its is long enough. Waiting queues, e.g. of ActorResources, empty in meantime.

As shown in the example, the integrated simulation of business processes and IT systems as proposed in this paper considers the process impact on waiting queues of ProcessingResources as well as the IT impact on waiting queues ActorResources. The other impact discussed in the paper is considered, too. Thus, we expect the integrated simulation to adequately represent the mutual performance impact of business processes and IT systems that occur in reality which results in increased performance prediction accuracy.

VIII. Conclusion and Future Work

In this paper, we discussed the mutual impact of business processes and involved IT systems in terms of performance. We pointed out the need for an integrated simulation of business process and IT and argued that Palladio is an adequate foundation to realize an integrated simulation. We also showed some open issues in Palladio to support business processes. We presented extensions of the PCM and the simulation behavior in order to enable the simulation of the mutual impact of business processes and IT systems.

Currently, we are implementing the extensions proposed in this paper in the new Palladio software architecture simulator EventSim. In a case study, we are currently investigating the mutual impact of business processes and IT systems in practice. We plan to use the extended tool support in the case study to perform what-if analysis on a process and IT system from practice.

In the future, we want to evaluate the prediction accuracy of the integrated simulation. We also plan to investigate the feasibility of a combined simulation of fine-grained IT events and long simulated time frames. The usability of the new features has to be improved and further evaluated. For example, a graphical representation of a calendar is useful for human modelers to specify ProcessTriggerPeriods quickly in the ProcessWorkload driver. In addition to the FIFO principle, further scheduling policies for actor resources are possible and will be explored in the future. Furthermore, a translation to the LQN and QPN formalisms could be developed to allow for more lightweight simulation.

IX. Acknowledgement

The authors want to thank Philipp Merkle for valuable comments and support related to the implementation in EventSim.

References

[1] S. Junginger, H. Kühn, F. Bartl, and J. Herbst, "Evaluation of financial service organizations with adonis simulation agents," in *Proceedings of the 10th European Simulation Symposium (ESS 98)*, 1998, pp. 582–588.

[2] S. Becker, H. Koziolek, and R. Reussner, "The Palladio component model for model-driven performance prediction," *Journal of Systems and Software*, vol. 82, pp. 3–22, 2009.

[3] "Jboss community, Savara." [Online]. Available: http://www.jboss.org/savara

[4] S. Betz, E. Burger, A. Eckert, A. Oberweis, R. Reussner, and R. Trunko, *An approach for integrated lifecycle management for business processes and business software*. IGI Global, 2012.

[5] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. New York, NY, USA: Wiley-Interscience, 1998.

[6] W. M. C. Specification, *Workflow Management Coalition, Terminology & Glossary (Document No. WFMC-TC-1011)*. Workflow Management Coalition Specification, Feb. 1999.

[7] "Oxford dictionaries online." [Online]. Available: http://oxforddictionaries.com/

[8] N. Mi, G. Casale, L. Cherkasova, and E. Smirni, "Burstiness in multi-tier applications: symptoms, causes, and new models," in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, ser. Middleware '08. Springer-Verlag New York, Inc., 2008, pp. 265–286.

[9] P. Merkle and J. Henss, "EventSim – an event-driven Palladio software architecture simulator," in *Palladio Days 2011 Proceedings*, ser. Karlsruhe Reports in Informatics ; 2011,32, S. Becker, J. Happe, and R. Reussner, Eds., Karlsruhe, 2011, pp. 15–22.

[10] J. Rolia and K. Sevcik, "The method of layers," *Software Engineering, IEEE Transactions on*, vol. 21, no. 8, pp. 689–700, Aug. 1995.

[11] F. Bause, "Queueing Petri Nets˜- A formalism for the combined qualitative and quantitative analysis of systems," in *Proceedings of the 5th International Workshop on Petri Nets and Performance Models, Toulouse, France*, Oct. 1993, pp. 14–23.

[12] H. Koziolek and R. Reussner, "A model transformation from the palladio component model to layered queueing networks," in *Performance Evaluation: Metrics, Models and Benchmarks*, ser. Lecture Notes in Computer Science, S. Kounev, I. Gorton, and K. Sachs, Eds. Springer Berlin / Heidelberg, 2008, vol. 5119, pp. 58–78.

[13] P. Meier, S. Kounev, and H. Koziolek, "Automated transformation of component-based software architecture models to queueing petri nets," in *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*, Jul. 2011, pp. 339–348.

Case Study: Palladio-based Modular System for Simulating PLC Performance

Jens Frieben
Fraunhofer IPT
Project Group Mechatronic Systems Design
Zukunftsmeile 1
33102 Paderborn, Germany
Email: jens.frieben@ipt.fraunhofer.de

Dipl.-Ing. Henning Heutger
PHOENIX CONTACT Electronics GmbH
Business Unit Control Systems
Dringenauer Strasse 30
31812 Bad Pyrmont, Germany
Email: hheutger@phoenixcontact.com

***Abstract*—Modern facilities controlled by programmable logic controllers (PLC) have to fulfill a broader range of tasks and meet higher performance requirements than a few years ago. They have to cope with their basic control tasks as well as communicating with each other or external systems. The selection of an appropriate performance class during the early development phases is based on the experience of a developer or expensive and time consuming tests. In this paper, we present an approach to simulate the PLC and its environment to predict the CPU utilization. For this purpose, the PLC and its internal and external influences on the CPU utilization have been modeled with Palladio. Based on the simulation results of different usage scenarios, the developer is now able to choose a fitting performance class without conducting expensive tests.**

I. Introduction

The vision of increased flexibility in production by smart factories is becoming reality. Modern facilities and production plants are developed towards smaller production sizes and better reuse of machines. For this goal, the communication between different stations and even different plants, as well as more self-aware systems, have to be realized. This for example allows ad-hoc reconfiguration resulting in shorter setup times. Therefore, the Programmable Logic Controller (PLC) controlling and coordinating the different machines in a factory, need not only to handle more actuators and sensors (I/Os), but also react faster to more events and even provide additional services to external systems. To cope with these requirements, the PLC must fulfill higher and higher performance requirements.

Figure 1 shows a standard PLC communicating with various systems over one or more fieldbusses. It is mandatory that the provided input by different sensors is processed by the PLC and the correct commands are send out to the actuators in time. Any violation of these hard real-time constraints could lead to failures in the plant or, in case of safety-critical systems, cost lives. Their controlling functionality is performed by several programs running in tasks. These tasks are regularly executed in intervals (e.g. every four milliseconds) and called *CyclicTasks*. Also, other execution strategies like *EventTasks* exists, which are triggered by events. If the execution of the programs set inside a cyclic task takes longer than their specified time intervals, a run-time exception is thrown and the PLC stops all programs for safety reasons. Therefore it is most important to assure that the PLC provides enough performance for the in-time execution of programs. On top of the basic controlling functionality the PLC has to provide additional services. Figure 1 shows also external systems connected via fieldbus that communicate with the PLC. These services might belong to the upper management layer that coordinates the whole production plant (e. g. SCADA[1]) or machine to machine communication to negotiate new setups or configurations for different products. These service functions influence the performance of the PLC as well.

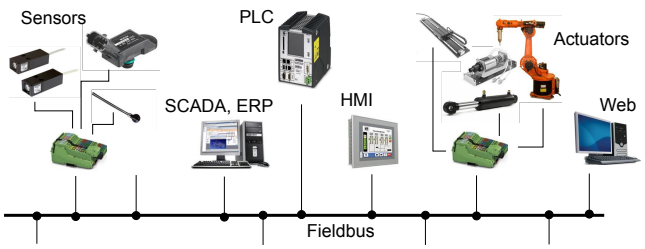

Figure 1. The PLC and its environment

During the development of automated systems, the specific details of the plant, the control programs and the services are not fully worked out. The most important factor, the control programs, are often finished after the plant has been set up or at least close to its end. This complicates the early selection of a suitable PLC performance class, considering the performance of the CPU.

In practice, an appropriate performance class is estimated based on the experience of the developer or with the help of time-consuming and expensive tests. Despite the fact that these tests run in parallel to the late development phases, they can often not consider or replicate

[1]Supervisory Control And Data Acquisition (SCADA) systems monitor and control industrial processes

all the factors and scenarios that should be taken into account.

In cooperation with Phoenix Contact, we developed an approach for simulating their PLC to predict the future utilization of the CPU. This allows the developer to pick a fitting PLC performance class for the specific task at hand. These simulations, which can be easily set up and executed, provide a fast feedback to the developer and can be refined along the development process of the automation system. The key data used as parameters for the simulation is updated based on the current or future development states. This approach helps to cover more usage scenarios than it would be feasible with tests and to save costly setups.

The focus of this work lies on predicting the CPU utilization of the PLC and not its hard real-time behavior like it is done for other embedded devices[1]. However, the simulation can provide approximated insights to the response times of the running programs and anticipate possible run-time exceptions.

Figure 2 shows the process from specifying the automation system to its simulation. First, the developer creates an abstract *automation model* containing the key data used for the simulation. In the next step, this information is transformed into Palladio Component Models (PCM). These models belong to the Palladio [2; 3] performance prediction framework which carries out the simulation. The result of this simulation is a collection of sensor values which are used to calculate the overall CPU utilization.

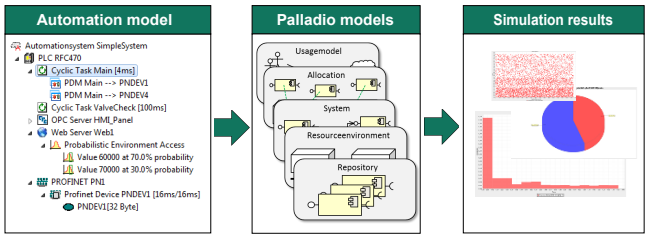

Figure 2. From automation model to simulation results

The paper is structured as follows. We introduce the different influence factors of a PLC in Section II. In the following Section III the automation model used to specify these influence factors is presented. In addition, the Palladio models used for the simulation and the transformations to generate them from the automation model are described. Section IV gives a short overview of the simulation results. We conclude this paper in Section V and give a short outlook of future work.

II. Influence factors

In this section we explain the influence factors that are covered in our approach and shown in figure 3. We introduce each factor but will detail only a few to give an idea of the different levels of detail used for the final simulation model.

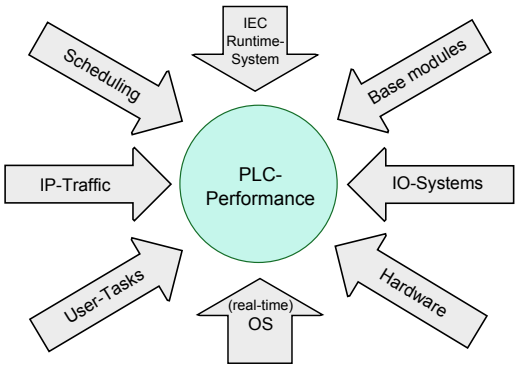

Figure 3. Performance influence factors of a PLC

- **Hardware** is one of the most obvious influence factors on PLC performance. Primary factor is the clock rate of the PLC, which is specified as instructions per simulation time unit. For our first attempts to simulate the Phoenix Contact PLC we focused on a specific product, the RFC 470. This PLC has a clock rate of 1 GHz and therefore is specified with 1.000.000 instructions per time unit, setting the simulation time unit to 1 ms.
- **User-Tasks** are the most important influence factors on the PLC performance. Each task carries out one or more programs that calculate the commands for the actuators and therefore controls the behavior of the machine. There are different kinds of tasks: event based, system, default, and cyclic. For this use case, only the cyclic task - which is the most commonly used task type - has been regarded. For the cyclic task, three values can be specified. The interval between executions (*cycletime*), the priority of the task (*priority*), and an estimated worst-case-execution-time (WCET) of the programs (*execution-time*). The execution time is specified in milliseconds and currently is based on the developers estimation or measurements, in case the programs already exists.
- **(Realtime) OS** and **IEC-Runtime system** belong to the PLC performance base load. The IEC-Runtime system (called ProConOS in Phoenix Contact PLC) executes the programs defined in the tasks, written in the languages specified in the IEC 61131-3[4] standard. The IEC-Runtime system is executed on top of an operating system. Both systems, IEC and OS, put a certain amount of load onto the PLC.
- **Scheduling** takes place in the IEC-Runtime system to schedule the different tasks according to their priorities and on the level below regarding the operating system. Caused by preemption, low priority tasks might have significantly increased response times. Therefore the scheduling must be incorporated into the simulation to get precise results.

- **IO-Systems** are a crucial part of automation systems. Sensors provide input for a PLC and actuators influence their environment. The data send from and to the PLC is transmitted via fieldbusses. A fieldbus is an industrial network system for real-time distributed control. A PLC might be connected to one or more different fieldbus systems like PROFINET[5] or INTERBUS[6]. Depending on the PLC, the communication over fieldbusses is either realized in hardware (e.g. DPM[2], FPGA[3]) or in software. A communication over hardware also uses a certain amount of CPU time, but is significantly faster than a software solution. Due to the real-time constraints (fieldbusses must adhere short communication cycles down to 250μs), they consume a significant part of the CPU performance.
  For the case study conducted with Phoenix Contact, the first fieldbus system under investigation is PROFINET. We focused only on the CPU utilization, not remote communication effects or response times. It supports different operating modes ranging from non-real-time (e.g. HMI access) to real time use in motion control. Sensors and actuators are called PROFINET-Devices (short pndevice). Data between PLC and pndevices are send in predefined time slots in a cyclic manner. Each pndevice consists of one or many modules, on which sensors and actuators can be connected to. The module size defines the amount of data that can be send to or from a pndevice. At the PLC, the data is provided to the running programs as variables, also called process data. Therefore, each task has a connection to a specific module in the PROFINET. A message send from the PLC to a pndevice has a fixed length based on the sum of all modules. The payload it transports is data that will update the programs variables. Figure 4 shows a short summary over the important properties needed to specify the influence factor IO-System on the PLC. The pndevice that *sends* and *receives* messages in specified intervals, the modules (*modulsize*) attached to it, and the data (*datasize*) transferred to the PLC.
- **Base modules** (services) are used for secondary functions of the PLC. Most of these services provide means for communication with other devices or systems. The three services handled for this study are:
  - **OPC**: OPC stands for for Object Linking and Embedding (OLE) for Process Control and is used to exchange data between control devices. The OPC-Server provides the variables used by the programs for either visualization (user interfaces) or for SCADA-systems for controlling purposes. The OPC-client requests these variables in predefined *intervals*. In addition, the *variable type* and the number of variables have an impact on the PLC performance.
  - **FTP-Server**: A FTP-Server is used to download log-files from the PLC or to upload new projects (specifying tasks and programs) to it. The *filesize* and the access *frequency* is important for simulating the CPU utilization.
  - **Web-Server**: The Web-Server allows remote users to view the state and variables of the PLC. Of course this generates a certain amount of CPU usage depending on the *access time* and the *file size*.
- **IP-traffic** is an influence factor depending on the fieldbus implementation. Traffic, whether from the control network or other IP-based devices, can cause additional CPU load for analyzing and forwarding the messages either to the windows- or the PROFINET communication stack.

[2]Digital Processing Module (DPM)
[3]Field-Programmable Gate Array (FPGA)

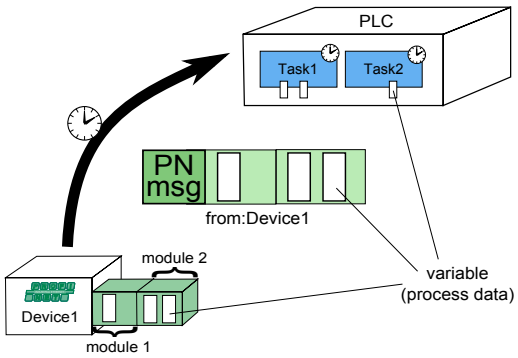

Figure 4. Sending and receiving of PROFINET messages

III. Modeling

In this section the different models and their relations to each other are described. One important goal is the easy use and handling of a simulation. The target audience for carrying out performance simulations are electrical engineers and, for future plans, non-technical personnel. Due to the complexity of modeling (even a medium sized PLC) with Palladio, a domain specific language for abstracting the details was necessary. In Section II the most important influence factors have been identified. These factors can now be specified with the automation model. The following subsections highlight some details about the automation model, the Palladio models that are used for the final simulation, and the transformation rules for making the transition between them.

A. Automation model

The automation model is used to specify the influence factors of the PLC in a simple way. Its structure is similar

to the tool PC Worx[4] used to program and configure the fieldbusses of PLC. First we created an Eclipse Modeling Framework (EMF) based meta model to be able to specify the automation system in a precise and automatically analyzable form. This allows the developer to create an automation system with a PLC and its specific settings and environment.

Figure 5 shows an excerpt from the automation model meta-model without properties. Root element is the *AutomationSystem* element, which can contain multiple PLCs. Currently the simulation only supports one, but for future use, a complete system of PLCs communicating with each other, should be realized. Several *Services* may be added to the PLC, which can be used to set up Web-Server or OPC-Server loads. Also, one or more *Tasks* can be added to the model. The properties that can be specified for a task, like calculation time or priority, are identified in Section II. The *ProcessDataMappings* are used to connect variables of the task with specific *IOModules* which are added to *IODevices*. They are part of a fieldbus like PROFINET.

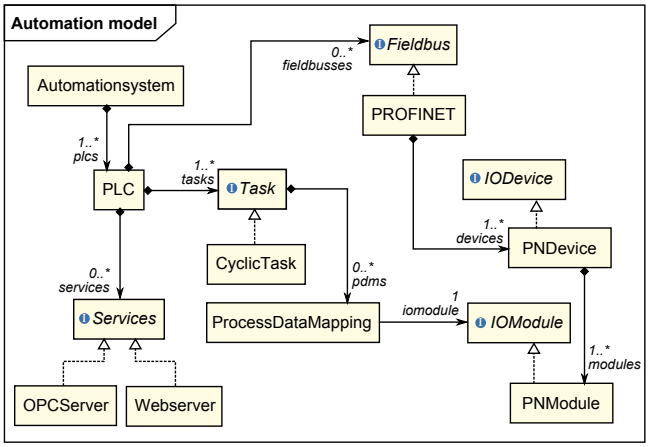

Figure 5. Automation model meta-model (excerpt)

A possible instance of this meta-model is shown in figure 6. The screenshot of the generated EMF-Editor shows two *CyclicTasks* (*Main* and *ValveCheck*) which are executed every 4 ms, resp. every 100 ms. Each task needs exactly 1.5 ms execution time. In this example, different variables are connected to the pndevices via *ProcessDataMappings* (PDM), which specify the size of the variable send over the fieldbus in bytes. The pndevices haven been created with an send/receive interval of 16 ms. Two additional services are the OPC-Server, providing several variables at a fixed refresh interval and a Web-Server with varying access times.

B. Palladio models

In this subsection, we explain how we measured the different influence factors and captured them in a simulation model. For our approach, we used the Palladio Component model (PCM) and SimuCom[7] for the simulation. PCM consists of five models (repository, usage, system, allocation, and resource environment) each specifying a specific aspect of the automation system. The repository is used to model the influence factors and internal processes of the PLC. Most important element to model the CPU utilization is the *ResourceDemand*. It is specified in *Service-Effect-Specifications* (SEFF) which can be compared to methods or functions of programming languages. They might call internal and external actions which refer to other SEFFs. Inside a SEFF, a *ResourceDemand* for a given resource (e.g. CPU or HDD) can be defined. When calling for example the SEFF named *receive_PROFINET_Message*, the simulation will stress the CPU with 1000 units of CPU time. This corresponds to the number of instructions used to read from the network hardware and write the data into the memory. A SEFF also supports a conditional control flow, which allows the modeling of alternative resource demands depending on given parameters.

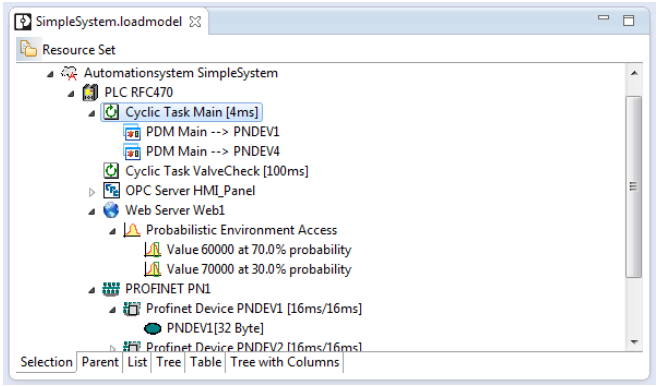

Figure 6. Screenshot of the eclipse based automation model editor

For our approach, we provided only the resource "CPU", since it is the primary goal to predict the CPU utilization. The HDD, a compact flash card, was neglected in our models due to rare accesses and its fast response time. Including the network interface into the model, as a possible bottleneck for some operations, is up for future work. The other PCM models are used to specify how a system is composed from elements specified in the repository (system model) or how the system is deployed (allocation model) onto the hardware (resource environment model). The usage model is used to specify how often, in which intervals, and with what parameters the SEFFs are called.

In the following we will give an insight how the PLC and its environment have been modeled with Palladio and how we measured the necessary resource demands for the different parts of the model. For this case study, we focus only on four influence factors: the operating system including its scheduler, cyclic tasks, the PROFINET

[4]Phoenix Contact development and configuration tool

load, and the OPC-Server.

Operating system & scheduler: To model the operating system in PALLADIO, two points have to be considered. First the scheduler and second the base load generated by the OS internal processes. Scheduling influences the response times for the different processes, not the CPU utilization in general, which is focused in this work. Still, a realistic preemption of processes was requested for future work. High priority tasks for the communication (like PROFINET) will always be executed before tasks, which can lead to run-time exceptions. Palladio supports some abstract out-of-the-box schedulers, but don't provide the means to set up the process priorities in the PCM. Therefore the scheduler and means to model the process priorities had to be implemented. The operating system for the RFC 470 is a modified version of WindowsCE 5.0, for which a new scheduler specific *resourcetype* has been created. This resourcetype is referenced in the resource environment, which is used to specify the hardware and its properties. Accompanying the resourcetype is a scheduler-file that configures the properties of the OS scheduler. These properties influence how the OS handles the different running processes. The most important settings are:

- Prioritylevel, which specify the range of integer values that can be used to put processes in an order. This order is used to process higher priorities (0) before lower priorities (255).
- Process-Handling defines in which order the processes are run or continued. The strategy used by WindowsCE 5.0 is that lower priority processes are preempted by higher ones. Processes with the same priorities are handled with a Round-Robin strategy.
- Timeslice/Quantumsize defines a unit for available running time. After a process completes its quantum, WindowsCE 5.0 may choose to run another process based on priority or state. The default quantum time set in the resourcetype is 25 ms.
- Starvationboosts are used to push threads which have been preempted for too long. The setting for the simulation is 0.

The simulation uses the configuration to set up the inbuilt scheduler, which manages the different incoming resource demands. For further information about the scheduler, its configurations, and functionality see [8].

In addition to the scheduling, the basic load of the PLC has to be considered. With a special version of the RFC 470 is it possible to see the CPU utilization calculated by the operating system. Unfortunately this includes not only the basic OS functions, but also the IEC-Runtime system executed on start-up. Since both influence factors could not be separately measured, the IEC-Runtime load has been combined with the OS basic load to a module named *WinCE5* load. To determine the ResourceDemand, the PLC has been set into a running state with no projects, services and connected devices. The CPU utilization has been read off the display, showing 16%, spiking between 15 % to 17 %. With a 1 GHz CPU, resulting in 1,000,000 instructions processable per millisecond in the simulation model, the ResourceDemand for WinCE5 is set to 160,000 CPU units. To reflect the spikes, a range of possible values has been set up in the SEFF's ResourceDemand with a DoublePDF-formula:

$$DoublePDF[(155200.0;\ 0.0)(157600.0;\ 0.05)\ (162399.99;\ 0.90)(164800.0;\ 0.05)]$$

The deviation of the exact value can also be specified in a transformation rule (see section III-C) allowing a fine tuning without changing the repository model. To trigger the SEFF containing the resource demand for WinCE5, a usage scenario with an closed workload has been created. The workload, containing just one worker, starts every millisecond. The result of a simulation with just the WinCE5 load is shown in figure 7. The red dots show the simulated load of the windows operating system. The load varies exactly in the ranges specified by the DoublePDF function.

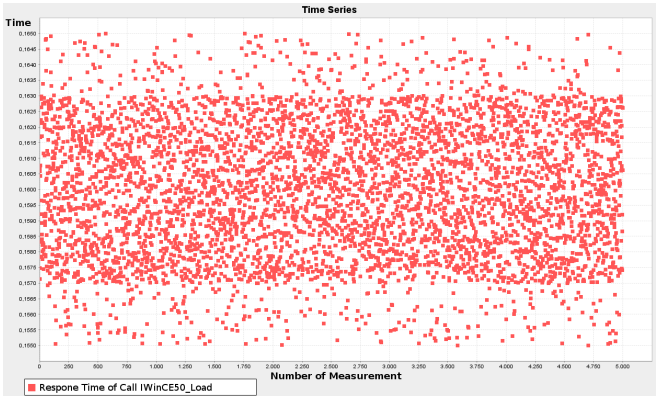

Figure 7. TimeSeries for the WindowsCE and ProConOS baseload

CyclicTasks: The creation of the PALLADIO models for cyclic tasks is pretty straight forward. Each CyclicTask has a property *executiontime* which specifies how many milliseconds the task will put a load on the CPU. This time can be converted to a ResourceDemand by checking how many instructions the hardware can process per millisecond. More complex is the issue of cyclic executions with fixed intervals. PALLADIO supports two types of workloads - open workload and closed workload. Both types wait a predefined time and call the specified SEFFs. After the SEFF has been executed, they start waiting again. This behavior can be applied if no preemption takes place and the tasks are executed always with the same response times. In this case, the execution time can be subtracted from the cycletime and the difference used for the workload specification. Due to preemption it is possible that the task response times vary. Therefore we use

a workaround to start the SEFFs in exactly the same intervals via closed workloads. To do so, we use a TASK-component with a SEFF including a fork. This fork calls in one execution path the SEFF with the actual ResourceDemand inside a LOAD-component. The other path contains no actions and therefore returns immediately to the workload, restarting the waiting time. Figure 8 shows the two components and their interfaces for the task *Main*.

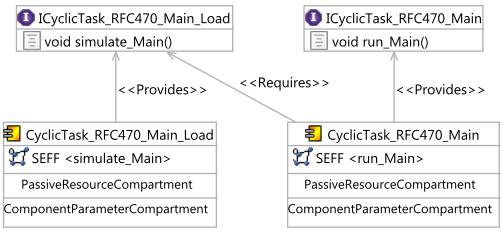

Figure 8. Interfaces and components for CyclicTasks

IO: PROFINET: A more sophisticated example is the modeling and simulation of PROFINET, which supports different performance modes. One of this modes, IRT (Isochronous Real-Time), has been investigated in more detail and its CPU utilization depending on the previously identified parameters (see Section 3) modeled in Palladio. For determining the resource demands, a series of measurements conducted by Phoenix Contact has been used. They instrumented the communication stack and important parts of the data flow from the network interface up to the tasks. Based on these information, four phases have been identified, which are shown in figure 9. In the first phase (receive), the pndevices send their

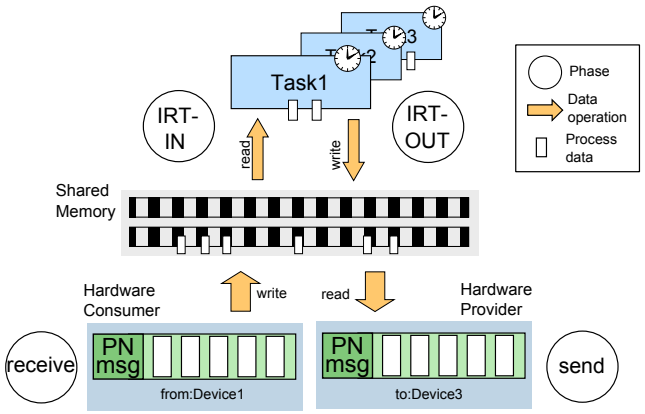

Figure 9. The four phases for the processing of PROFINET data.

messages via the network to the PLC. There, a function takes the message from the hardware consumer (interface buffer), decodes and analyses it, and puts the data into a predefined position in a shared memory. This function uses a certain amount of processing power, depending on module size (=message size) and the size of the data that will be copied. In the next phase (IRT-IN), each tasks refreshes its variables with data from the shared memory. The cycletime of the tasks and the size of the variables influence the consumption of CPU resources. After the task has finished, the variables are written back into the shared memory (IRT-OUT), packed into a message and send out to the pndevices (send). Based on the provided measurements, a function for each phase could be determined that is used to calculate the resource demand. At the current level of detail, we use the functions to calculate a mean value for each phase. This simplifies the calculation and we are able to combine all phases in one usage scenario. This might result in a slightly different load profile, but due the nature of the IRT-modus, in which the tasks usually have cycle times of 1 to 2 ms this is a reasonable deviation. For other PROFINET modes like *RT* new models and functions are needed.

An exemplary function to calculate the CPU resource demand for the receiving of messages and storing their values in the shared memory is shown in equation 1. The function *conDev* returns all pndevices for the PLC, *modules* returns a set of modules for a pndevice, *modulsize* retrieves the size of the given module, and *sendInterval* returns the millisecond between each message send to the PLC.

$$f_{PN_{receive}}(plc) = 10 + \sum_{conDev(plc)}^{d} \frac{(0.023 * \left(\sum_{modules(d)}^{m} modulsize(m) \right)) + 2{,}380}{sendInterval(d)} \quad (1)$$

OPC-Server: The last influence factor we like to present is the OPC-Server. During the measurements, we identified the previously mentioned parameters (see Section II) *variabletype*, *amount*, and the OPC-Clients request *intervals*. Table I shows the CPU utilization for varying amounts of String variables and different refresh times.

Table I
CPU utilization for string variables with different refresh intervals (excerpt).

#vars	0	1	10	50	100	200	1000
1 ms	16%	19,5%	19,5%	21%	24%	26%	30%
10 ms	16%	19,5%	19,5%	21%	24,5%	25%	30%
100 ms	16%	16,5%	16,5%	17%	17%	18%	18,5%

For the measurements we created an IEC program running in a low priority task, performing no operations. With the tool PC Worx the amount of initialized variables could easily be adjusted. The OPC-Client has been simulated with the tool OPC Test Client, which allows to configure different refresh rates. The variable type has a

non-negligible impact on the CPU utilization. When testing only a subset of all possible variable types (including self-defined STRUCTS), the type string used up more CPU resources than others.

The result of the measurements for a refresh interval of 1 ms is also visualized in figure 10. Starting at a base load of 16 % CPU utilization, an almost linear increase up to 2000 variables can be identified. Afterwards a saddle point is observable which could be an indicator for a bottleneck. Due to the limited sensors available, we can only assume that this bottleneck is caused by the network interface, pushing out messages to the OPC-Client. Therefore, only the linear section, marked with the dotted rectangle, is used to determine the CPU resource demand, allowing us to create the following function which is parameterized by the number of variables n:

$$f_{String}(n) = 0.0105 * n + 34895$$

Such a function must be created for each variable type. Due to the similarities of most types, we combined them to groups and thereby reducing the number of needed calculations. Using String and other variables in one OPC-Server request had the effect of using the lower CPU utilization curve instead of the higher. The reason for this behavior still needs to be investigated. Finally a usage scenario can be created which calls the OPC-Server SEFF that generate CPU resource demand. A single worker in an open workload is run in intervals specified by the OPC-Clients refresh interval.

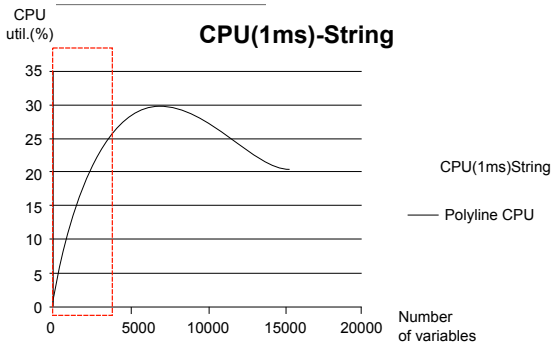

Figure 10. CPU utilization for the OPC-Server (String, 1 ms)

C. Transformation

Creating performance models with Palladio is a complex and time consuming task. For this reason, the automation model has been created, which hides the Palladio modeling from the user. This section will give some insights how the automation model is transformed to Palladio models and what advantages this approach offers.

The automation model and the five Palladio models (repository, usage, allocation, system, and resourceenvironment) are based on the EMF technology. This allows us to choose from a range of Model-To-Model approaches provided for Eclipse[5]. To select a fitting transformation technology we used the study conducted by Lehrig[9]. He provides a first guidance in form of a decision-tree. Based on the requirement to allow an easy modification of the transformation rules and a target audience of C/C++ developers, we choose QVT-Operational (QVTO)[10]. QVT-O is an imperative language designed for writing unidirectional transformations. Its imperative language is similar to programming languages like C++ and its text based specification allows an easy customization of its transformation rules. Additionally, QVTO supports QVT-BlackBox operations for invoking external code. This allows us to specify complex calculations or queries through the automation model in JAVA. Alternatives to QVT-O like ATL[11], TGG[12], or XTend[13] were not covered in Lehrig's work.

For the design of the transformation rules we followed the requirement to be as modular as possible. This lead us to the decision to create one transformation for each influence factor. A transformation contains mapping rules that specify how an object from the automation model is transformed to an object in the Palladio models. The top-level object in the automation model is of type *Automationsystem*. The root-transformation rule creates the five Palladio models based on this Automationsystem and sets up the hardware. Afterwards one or more follow-up transformations can be executed. This approach gives us the flexibility to build up the Palladio models incrementally, creating first model elements for cyclic tasks, OPC-Server, WindowsCE and so on. The functions used to calculate the different resource demands are implemented in the QVTO-Blackbox operations for each transformation. This modular approach allows us to combine different transformation rules to a product configuration. Such an exemplary configuration is shown in figure 11. The configuration named *RFC_470_Rev_2.34* is used for a specific firmware revision of the RFC 470. The firmware consists of several transformations, each creating a specific part of the Palladio models. These sub-transformations are called by the root-transformation. In case a FTP-Server firmware update enhances the performance of the service, a new sub-transformation is created and the old one is replaced. This allows a simpler creating and customizing of existing firmware configurations.

Another advantage of using QVTO-transformations to generate the Palladio models is the possibility to easy modify attributes and configurations. The listing 1 shows an excerpt from the transformation *WinCE5.0_Rev01*, which contains several global variables. Changing the priority of WindowsCE related processes can be done by just setting a new integer value. The same applies to the *osI-*

[5]Eclipse IDE - www.eclipse.org

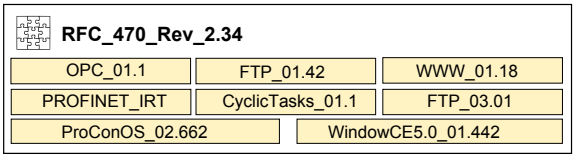

Figure 11. Product configuration for a specific RFC470 firmware version.

dlePercentage and *osDeviation*, which influences the creation of the ResourceDemand as mentioned in III-B.

Listing 1. Excerpt from the WinCE5.0_Rev01 transformation rule

```
// ------ Operatingsystem (Windows CE 5.0)
property osName : String = 'WinCE50';
property osPriority : Integer = 250;
property osIdlePercentage : Real = 0.16;
property osDeviation : Real = 0.03;
```

The validity of the resulting performance model holds for this level of detail and as long the influence factors only use the resource CPU and the scheduler manages all demands. As more fined grained actions are added to the model, interactions between the different modules (e.g. via passive resources like shared memory or FCFS queues at the network interface) should be considered, too. For this, the root-transformation, which creates all necessary models before executing the sub-transformations, could create the needed common resources as well.

IV. Evaluation

Performance tests with actual hardware in a networked environment are a time consuming and costly task. Therefore an sophisticated test covering all influence factors could not be conducted. Instead several smaller tests could be made which focus on one or two influence factors at once. These tests are based on measurements carried out in Phoenix Contact's dedicated testing facilities or in small, controlled environments.

Due to the nature of this contract research, most of the timing results and actual readouts are not approved for the public. Therefore only a few or just small excerpts from the simulation results can be shown here.

In the first test, different cyclic tasks have been simulated and compared with real executions on the PLC. For this, an IEC program has been created that allows the generation of load. The program performs stressing calculations such as cosine operations and watched its own execution time. This time is not the response time, which might be influenced by preemption, but the aggregated execution times. In Table II different tests are listed. Each test contains its simulated and real CPU utilization or run-time exception (RTX). In addition to this, the test name and a brief description of the setups are given. For example, stands *T(15 ms,5 ms),T(150 ms,5 ms)* for two tasks, in which the first produces a load of 5 ms and cycles with 15 ms, the second task with 5 ms load every 150 ms. These tests include the WinCE5.0 module and its base load. The average deviation between simulated and real values are just about 2 percent. This fault could be caused by internal caching effects that are not yet considered in our performance models.

Table II
CPU utilization of cyclic tasks with various configurations

Testname	Real(%)	Sim(%)	Description
1T	40-42	38	T(4ms,1ms)
3T	52	53	3x T(15ms,2ms)
3TSEMI6	60	63	3x T(6ms,1ms)
5T	61	63	5x T(10ms,1ms)
1TFAST	63	63	T(2ms,1ms)
T2-MIX1	48-50	50	T(15ms,5ms),T(150ms,5ms)
T3-MIX2	54	53	T(4ms,1ms), T(10ms,1ms),T(15ms,1ms)
T4-MIX3	RTX	61	T(4ms,2ms),T(15ms,2ms), T(30ms,4ms),T(150ms,6ms)
T4-MIX4	63-65	68	T(4ms,1ms),T(15ms,2ms), T(15ms,4ms),T(150ms,6ms)

Second, we present the comparison of measured and simulated CPU utilization of PROFINET running in IRT mode. Table III shows the number of pndevices, modulesize, size of the data sent (none or all), and the simulated and measured CPU utilization. Despite a constant offset of about two percentage points the values are almost identical. The measurements were collected with instrumented code that logged the execution times. This could also be a possible reason for this deviation, due to fact that logging also consumes CPU processing power.

Table III
CPU utilization comparison PROFINET real measurements with simulation (excerpt).

#Devices	3		12		16	
ModSize	1x32		1x128		1x64	
DataSize	0	32	0	128	0	64
Real(%)	27,5	32,5	33,2	50	35,5	57
Sim(%)	29,8	33,9	35,6	51,6	38,6	59,87

V. Conclusion and Outlook

Future production plants and automation systems need to cope with more complex tasks, including communications with each other and external systems. To cope with these requirements, their PLC have to handle their normal controlling tasks as well as new services. This complicates the early selection of a suitable PLC performance class by the developer. With our approach, we support this decision by providing a performance prediction of the PLC and its future environment and tasks. To simplify the simulation, we created an abstract automation model, containing all viable information about the automation system in a form that is familiar to the developer. This model is automatically transformed to the

Palladio models needed as input for the simulation. The transformations used for this step are created in a modular way, allowing the combination of different influence factors to a specific product configuration. Despite a full evaluation was not available at the time this paper was written, the results of smaller evaluation tests showed a promising start.

During the modeling of the internal and external influence factors on the PLC we also identified some Palladio features that were missing and could be used for detailing the model:

- Usage scenarios might use state based information to call different functions or use different parameters. This would allow us to model different amount of communication accesses for a system tick instead of calculating the mean value, leading to a more precise model.
- A (graphical) tool for easy definition of DoublePDM and DoublePDF functions would have helped to add various resource demands based on given load-curves.
- More sensors for easier analysis of for example minimum and maximum response times to detect run-time exceptions.
- Use of interpolated multidimensional look-up tables which are able to return a resource demand based on given parameters, like MatLab[14].

In the future, further influence factors should be incorporated into the simulation as well as refining existing ones. Using a modular system, some parts of the model can easily be exchanged by more precise ones. Also, new hardware modules should be added to the simulation like FPGA or DPM. Finally a full scale evaluation, which is already being prepared, based on an existing application will show the accuracy of the performance prediction.

References

[1] J. Happe, "Performance Prediction for Embedded Systems," pp. 1–16, 2005.

[2] R. Reussner, S. Becker, E. Burger, J. Happe, M. Hauck, A. Koziolek, H. Koziolek, K. Krogmann, and M. Kuperberg, "The Palladio Component Model," Karlsruhe, Tech. Rep., 2011.

[3] S. Becker, "Coupled model transformations for qos enabled component-based software design," Ph.D. dissertation, UniversitÃd't Oldenburg, Uhlhornsweg 49-55, 26129 Oldenburg, 2008.

[4] INTERNATIONAL ELECTROTECHNICAL COMMISSION, "IEC 61131-3: Programmable controllers - Part 3: Programming languages," 2003.

[5] Profibus and Profinet International (PI). [Online]. Available: http://www.profibus.com

[6] Profibus Nutzerorganisation e.V. (PNO). [Online]. Available: http://www.interbusclub.com/

[7] S. Becker, H. Koziolek, and R. Reussner, "The Palladio component model for model-driven performance prediction," *Journal of Systems and Software*, vol. 82, no. 1, pp. 3–22, 2009.

[8] J. Happe, "Predicting Software Performance in Symmetric Multi-core and Multiprocessor Environments," Dissertation, University of Oldenburg, Germany, August 2008.

[9] S. Lehrig, "Assessing the quality of model-to-model transformations based on scenarios," Master's thesis, University of Paderborn, Zukunftsmeile 1, October 2012.

[10] Object Management Group. "Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT)". [Online]. Available: http://www.omg.org/spec/QVT/

[11] E. M. Project, "Atl (version 3.2.1)." [Online]. Available: http://www.eclipse.org/atl/

[12] A. SchÃijrr, "Specification of graph translators with triple graph grammars," in *in Proc. of the 20th Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG '94), Herrsching (D.* Springer, 1995.

[13] Eclipse.org, "Xtend (version 2.3.1)." [Online]. Available: http://www.eclipse.org/

[14] MathWorks, "Matlab (matrix laboratory) numerical computing environment." [Online]. Available: http://www.mathworks.de/products/matlab/