Towards a Common Classification of Changes for Information and Automated Production Systems as Precondition for Maintenance Effort Estimation

Birgit Vogel-Heuser, senior member IEEE, Thomas Simon, Jens Folmer
Institute of Automation and Information Systems
Technical University of Munich
Munich, Germany
{vogel-heuser, thomas.simon, folmer}@tum.de

Robert Heinrich, Kiana Rostami, Ralf Reussner
Institute for Program Structures and Data Organization
Karlsruhe Institute of Technologies
Karlsruhe, Germany
{heinrich, rostami, reussner}@kit.edu

Abstract—**Both information and automated production systems (aPS) evolve during their lifetime, e.g. due to changes in requirements and infrastructure. In order to estimate maintenance effort in information systems the KAMP method is applied. This paper discusses the necessary classification of changes as a prerequisite to apply such a method. Aggravating aPS consist not only of software but also include mechanics and electric/automation hardware. Therefore, the classification has to be enlarged to a multi-disciplinary one. The limitations of this approach for aPS are discussed in detail and demonstrated using three scenarios of a lab size pick and place unit. The paper closes delivering first ideas to cope with these.**

Keywords—software evolution; software maintenance; automated production systems; classification of evolution

I. Introduction

Both information and automated production systems (aPS) have a lifetime of several decades during which the systems evolve and thus are modified for the purpose of correction, improvement or adaptation [2]. For information systems (IS), an exemplary evolution life cycle (cp. Fig. 1) begins with the documentation of requirements and design decisions. This is followed by a static quality analysis possibly leading to a redesign before the system is implemented and deployed. A dynamic quality analysis is conducted for the running system to identify runtime issues [3] which may result in automated adaptation or trigger a new iteration for evolutionary changes conducted by human developers.

aPS on the other side not only consist of software but also of mechanics and electric/automation hardware, i.e. three disciplines are involved. These are typically unique systems which are designed and implemented on the basis of a contract between a customer and an aPS supplier at which the engineering of an aPS is carried out in the form of a project [4] [5]. Reusable (partial) solutions are developed during project independent activities and are used during project-related activities over the lifetime of an aPS (cp. Fig. 2), which usually encompasses a period of decades. During their operation time aPS are aging due to physical effects (e.g. tear and corrosion). This leads to the replacement of components of both mechanics and electric/automation hardware after a couple of years up to a few decades, but with different intervals (cp. Fig. 2), e.g. mechanics every 20-40 years, automation hardware including electrics every 10-15 years and software once a week until once a year [6]. Furthermore, the replacing components usually are not identical to the original ones, because the original spare parts are no longer available and/or a modernization is preferred [5].

Apart from the physical effects, changing requirements, e.g. market requirements or legal requirements, are another reason for aging. A lot of the changes caused by this can be implemented by software adaptions, but occur again in comparatively short periods of time and may even be executed during runtime. When change is required, the system may run through the process of evolution: all new and modified or affected requirements are gathered and/or checked for their validity and the system design is adapted on this basis [6].

Hence, both the domain of IS and the domain of aPS face the same challenge of their systems' evolution. However, in aPS a required change cannot always be assigned to one of the disciplines explicitly. The worst case is, that a change can be implemented in either mechanics, electric/automation hardware or software with different side effects on and needed adaptions in the respective other disciplines (cp. Table 1 in [6]). In order to be able to compare the effort of two or more similar ways to maintain or evolve a specific system, supporting the decision which solution is more efficient and less time and cost

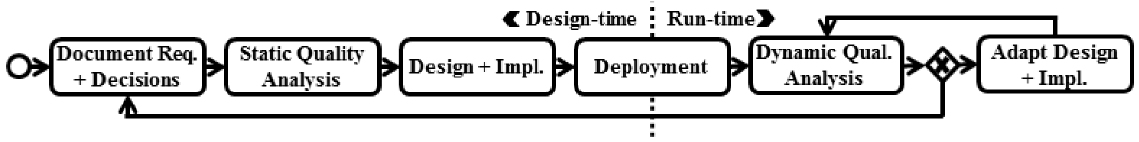

Fig. 1 Overview of the CoCoME Evolution Life-Cycle [1]

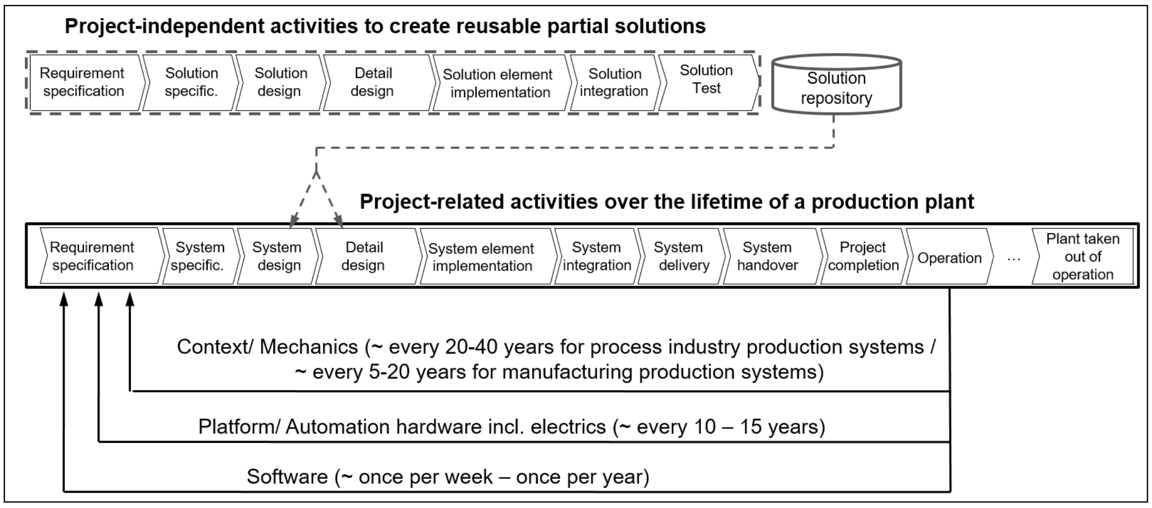
Project-independent activities to create reusable partial solutions
Requirement specification
Solution specific.
Solution design
Detail design
Solution element implementation
Solution integration
Solution Test
Solution repository
Project-related activities over the lifetime of a production plant
Requirement specification
System specific.
System design
Detail design
System element implementation
System integration
System delivery
System handover
Project completion
Operation
...
Plant taken out of operation
Context/ Mechanics (~ every 20-40 years for process industry production systems / ~ every 5-20 years for manufacturing production systems)
Platform/ Automation hardware incl. electrics (~ every 10 – 15 years)
Software (~ once per week – once per year)

Fig. 2 V-Modell XT integrated into life-cycle of different disciplines in aPS distinguishing between project-independent activities (top) and project-related activities (bottom) [6]

consuming would be beneficial not only for software in aPS. It would be utterly beneficial to be able to answer the even more challenging question whether a needed change should best be conducted in mechanics, electric/automation hardware or software or a combination of those.

In the domain of IS, methods for change impact analysis already exist. The approaches to change propagation can be divided into four categories [15]: (i) Task-based project planning approaches (e.g. [7], [8]) consider software architecture at a coarse-grained level and thus result in an accurate impact analysis. (ii) Approaches to architecture-based project planning (e.g. [9], [10]) and (iii) approaches to architecture-based software evolution (e.g. [11], [12]) do not support automated change impact analysis. And finally, (iv) works on scenario-based architecture analysis (e.g. [9], [13], [14]) consider only development activities and neglect the management tasks. Compared to these existing approaches, Karlsruhe Architectural Maintainability Prediction (KAMP) (cp. [15]) considers all kinds of software artifacts (e.g. source code, test cases or deployed instances). On the basis of a classified change request as well as the meta model of the architecture and the other artifacts, task lists for conducting the change are generated for the maintenance effort estimation.

However, a common classification of changes/evolution scenarios for both IS and aPS, which does not exist yet, is a precondition for the adaption of such methods for change impact analysis. In this paper, a first attempt for such a common classification is made and challenges are discussed on the basis of examples from both domains. For these examples demonstrators of each domain, i.e. CoCoME (cp. [16], [17]) for the domain of IS and a lab size pick and place unit (PPU) from the domain of aPS, are presented and evolution scenarios are described and classified.

The rest of the paper is structured as follows. Section II gives an overview on related work, while the case studies are introduced afterwards in section III. In section IV selected evolution scenarios are described and classified by introduced criteria and challenges are discussed at the same time. Finally, section V closes with the results and an outlook.

II. Related Work Regarding Change Classification

In this section, an overview of categorizations of (evolutionary) changes from the domain of IS is given, followed by a respective one for the domain of aPS.

As introduced by Lientz and Swanson [18], three types of evolutionary change can be distinguished – corrective, perfective and adaptive evolution. Buckley et al. [19] proposed a taxonomy of software change consisting of 15 dimensions that characterize the mechanisms of change and the factors that influence these mechanisms. The dimensions can be subdivided into four logical themes: temporal properties (when), object of change (where), system properties (what) and change support (how). Williams and Carver [20] proposed a software architecture change characterization scheme based on a systematic literature review. This scheme differs from change classification schemes because it does not match change requests into a particular class, but characterizes the change's impact with respect to a number of characteristics. Jamshidi et al. [21] identified five classification categories: (i) type of evolution, (ii) type of specification, (iii) type of architectural reasoning, (iv) runtime issues, and (v) tool support. Chapin et al. [22] propose a change classification taking into account (i) the software, (ii) the documentation, (iii) the properties of the software, and (iv) the customer-experienced functionality.

For aPS, a categorization of evolution is introduced in the following which is based on Vogel-Heuser et al. [6] and allows to distinguish different causal orders of change by which the three disciplines involved in an aPS are affected. This is due to different reasons for change by which evolution is initiated (cause of evolution).

In order to fulfill both changed and unchanged functional and non-functional requirements, which describe the desired behavior of aPS and change over their lifetime, the systems are undergoing various types of changes, i.e. evolution. Since all disciplines of the aPS may be affected due to changed requirements, the control software and/or the mechanical parts and/or the electric/automation hardware parts may be modified by the customer's maintenance staff. Ideally, a model driven approach is chosen from requirements to design, implementation, test and start-up. Nevertheless, "a well-managed and documented engineering procedure is not always performed in practice when requirements change" [6]. This is especially the case when changed requirements can be implemented by minor software adaptions, which is when they are usually performed instantaneously or even during runtime to avoid standstills.

Furthermore, changes in industrial practice can occur either during operation or a maintenance phase. These changes are initiated on the shop floor by maintenance personnel and necessary adaptions only affect either the software or electrical parts or mechanical parts or both of the latter. Thus, even the step of changing the requirements is omitted and they remain unchanged. The time of these changes is, according to Buckley et. al. [19], online and unanticipated, i.e. the changes are not foreseen during development.

In many cases, evolution in aPS is characterized non-sequentially. According to the classification proposed by Buckley et. al. [19], parallel evolution is caused by divergent changes. In aPS, divergent changes can occur e.g. when a plant, a single machine or its components are used as a basis for two or more different variants. Accordingly, divergent changes result in variants of a machine or plant. Such divergent changes occur frequently in industry because on the one hand in most companies evolution is realized by modification of existing similar components, but on different levels, i.e. sub-component level (atomic), component level (basic) or machine/plant level (application/facility) [23]. On the other hand, mostly non-functional requirements, e.g., PLC and device suppliers required by customers, control voltage depending on the country of delivery or different operating philosophies lead to different types of software, i.e. parallel evolution [6].

III. Introduction of the case Studies

In the following, both case studies, i.e. the CoCoME demonstrator and the pick and place unit (PPU), are described.

A. Common Component Modelling Example (CoCoME)

CoCoME (cp. [16], [17]) is a case study demonstrating a trading system which implements processes of a supermarket chain handling sales. Processes of CoCoME include all activities regarding processing sales (e.g. scanning products and paying) and all enterprise-wide administrative tasks (e.g. reporting and inventory management). It has a layered software architecture to support distribution of the software system. Several variants of CoCoME including various artifacts (e.g., test cases, requirement specifications, or architecture model) are available, such as plain Java code, service-oriented or hybrid cloud-based architectures [17].

B. Application example lab-size pick and place unit

A simple lab size model, the pick and place unit (PPU), is used as a demonstrator to research methods and technologies on evolving aPS. The PPU performs a (discrete) manufacturing process and handles, stamps and sorts different kinds of workpieces (cp. Fig. 3) [24]. The PPU consists of software, electric/automation hardware and mechanical parts.

The initial scenario is the evolution scenario Sc0 where only the stack, the crane and a slide (cp. Fig. 3, left bottom) exist. The stack pushes a single black plastic workpiece out of the stack into the crane's pick-up position. At the pick-up position, the crane picks up single workpieces by moving downwards and using a vacuum gripper to hold the separated workpiece. Upon rotation of 90 degrees, the crane reaches the slide's position, where the workpiece has to be placed. After moving down, the vacuum gripper releases the workpiece, which then glides down the slide.

An overview of all evolutionary changes of the PPU is available in [6] Table II.

A set of typical parallel variations in aPS based on the PPU's scenario Sc15 are discussed in the following. Due to the demand for a higher throughput of workpieces (WPs), scenario Sc15a with a faster sorting of WPs is developed. A drive with increased dynamics is installed to realize faster WP movement, which entails that faster pushers are required for extruding WPs. In parallel, a customer demands, as a non-functional requirement, an adjusted variant of PPU's scenario Sc15 (scenario Sc15b) which is able to handle larger and heavier WPs. Depending on the country, a machine or plant shall be located in, different supply and control voltage must be supported by field devices., e.g. whereas the existing PPU is engineered to be located in Germany, a customer requests a PPU which can be operated with different supply and control voltage (as used e.g. in the United States). Accordingly, all field bus components, which are not capable to handle the desired

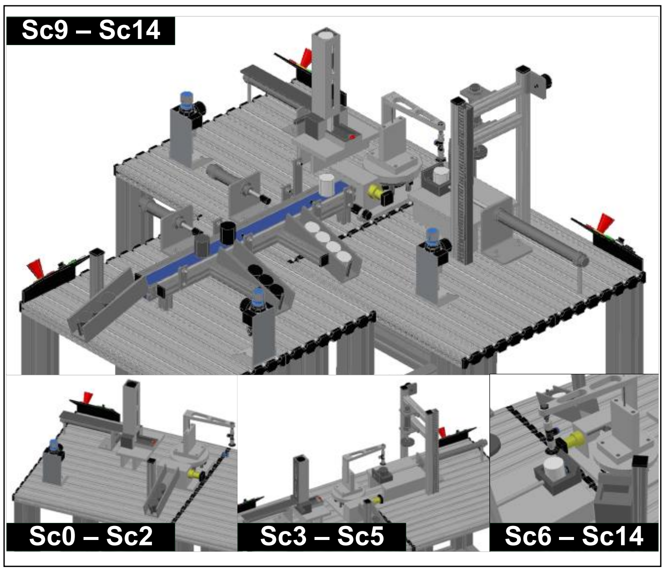

Fig. 3 PPU and related mechatronic configurations based on evolution scenarios from [24]

TABLE I
SELECTED EVOLUTION SCENARIOS OF THE PICK&PLACE UNIT

			Scope																		
	Scenario	Cause of evolution	Stack			Crane			Stamp			Slide			Conveyor Belt			Conveyor System			Realization
			Me	*AH*	*S*	*Me*	*AH*	*S*	*Me*	*AH*	*S*	*Me*	*AH*	*S*	*Me*	*AH*	*S*	*Me*	*AH*	*S*	
sequential evolution	…	…																			…
	2	Additional processing of metallic workpieces	A	A	A	o	o	o	-	-	-	o	o	o	-	-	-	-	-	-	Inductive sensor for metal
	…	…																			…
	4a	Reduction in the error rate caused by sensor contamination	o	o	o	o	M	o	o	o	o	o	o	o	-	-	-	-	-	-	Replacement of crane sensors
parallel evolution	…	…																			…
	15c	Different control voltage	M	M	o	M	M	o	M	M	o	M	M	o	M	M	o	-	-	-	Different I/O modules required
	…	…																			…

Me – Mechanics, AH – Automation Hardware, S – Software, A – Added, M – Modified, o – no changes

control voltage, have to be changed (scenario Sc15c, cp. Table I).

IV. DESCRIPTION AND CLASSIFICATION OF SELECTED EVOLUTION SCENARIOS

In the first part of this section the evolution scenarios of the CoCoME case study are presented together with a selection of change/evolution criteria. This is followed by the application of these criteria on the evolution scenarios of the PPU case study in the second part.

A. CoCoME evolution scenarios and selected change/evolution criteria

In the following, we describe how CoCoME is changed during a perfective (S1), an adaptive (S2), and a self-adaptive (S3) evolution scenario [25] (cp. Table II).

In evolution scenario S1, a new web shop is added, which allows customers of CoCoME to order the goods online and pick them up in an existing store. To this end, new use cases have to be considered and the existing design decisions have to be changed. In order to reduce the operating cost, the enterprise server and the corresponding database are migrated to the cloud in evolution scenario S2. In the case of increasing load, the limited capacities of the cloud provider may lead to performance issues of the database. In order to solve these, the database is migrated to another cloud provider in evolution scenario S3.

In order to classify the evolution scenarios, a selection of change/evolution criteria, which is based on the taxonomy of change described in the literature, is proposed. This selection of change criteria is then applied to scenario S1-S3 (cp. Table II) of which S1 is exemplarily described in more detail.

The cause of a change (cp. Williams and Carver [20]) is the motivation or trigger of an evolution scenario, such as emerging user requirements or changes in the technology stack. The type of a change (cp. Lientz and Swanson [18]) can be adaptive, corrective, perfective or self-adaptive whereat a corrective change fixes bugs and design flaws. Changes in the software environment can lead to adaptations in the software. Changes due to new or changed requirements are called perfective changes.

A further criterion describes whether the change is conducted at the systems design-time (offline) or at its run-time (online) (cp. Buckley et al. [19], Jamshidi et al. [21]). The scope describes, what part of the IS has been modified, i.e. the context, the platform or the software. This is further divided by the affected entities (cp. Buckley et al. [19], Williams and Carver [20]). Affected entities of the software can be e.g. components, interfaces or subsystems.

Furthermore, the granularity (cp. Buckley et al. [19], Williams and Carver [20]) describes the scale of the artifacts that have to be changed, which encompasses coarse (e.g. system, subsystem, composite component), medium (e.g. basic component, class, interface) and fine granularity (e.g. variables, method, statement). Following the definition in [26], a basic component specifies a behavior by operations of its provided interfaces, hence it is treated as a black box. In contrast, a composite component is composed of basic components and therefore is a white box. So we consider changes that refer to composite components as coarse grained and changes that refer to basic components as fine grained.

The change history includes all parallel or sequential changes, that moreover can be “anticipated” or “unanticipated” (cp. Buckley et al. [19]). The degree of automation refers to the degree of change support and it is to be distinguished between automated, partially automated and manual change support (cp. Buckley et al. [19]). Finally, the degree of formality describes, whether a change is deposited in an informal or a mathematical formal way, e.g. in a model (cp. Buckley et al. [19]).

TABLE II
CLASSIFICATION OF SELECTED EVOLUTION SCENARIOS OF COCOME AND THE PPU

	Criteria \ Evolution Scenario	Cause of Evolution	Type of Evolution	Design-/Run-time	Means of Evolution	Scope	Affected Entities	Granularity	Evolution History	Anticipation	Degree of Automation	Degree of Formality
CoCoME	***Adding a Web Shop (S1)***	Emerging User Requirement	PF	DT	Architectural Decision	ApplicationSoftware	Web Shop, Communication Interfaces	Coarse	S	A	Manual	Adhoc
	Platform Migration (S2)	Lower Costs, Need for Scalability and Flexibilty	AD	DT	Architectural Decision	Application Software, Platform Software	Data Layer, Interface to Business Layer	Coarse	S	UA	Manual	Adhoc
	Database Migration (S3)	Increased Usage Intesity, Upcoming Performance Issue	SA	RT	Recon-figuration	Deployment	Database (Component)	Fine (Component Deployment)	S	UA	Auto-mated	Formal Meta-model, Adaption Routines
PPU	***Scenario 2***	Additional processing of metallic workpieces	PF	DT	Adaptation	see Tab. 1	Mechanical fixture sensor; wiring for I/O and power supply; FB for sensor	Medium Cross-disciplinary	S	A	Manual besides Self-Healing	Adhoc
	Scenario 4a	Reduction in the error rate caused by sensor con-tamination	CR	DT	Adaptation	see Tab. 1	sensor changed	Medium, component	S	A	Manual besides Self-Healing	Adhoc
	Scenario 15c	Different control voltage	PF	DT	Architectural Decision	see Tab. 1	Interface to Sensor/Actuators changed, control voltage changed, mechanical change in local terminal	Medium, Cross-disciplinary	P	A	Manual besides Self-Healing	Adhoc

PF – Perfective, SA – Self-Adaptive, AD – Adaptive, CR – Corrective, DT – Design Time, RT – Run Time, S – Sequential, P – Parallel, A – Anticipated, UA – Unanticipated

Other criteria such as availability, openness and safety from Buckley et al. [19] are rather properties of the system itself than properties of the evolution scenario. Support activity, type of architectural reasoning and runtime issues are introduced in Jamshidi et al. [21]. However, support activity does not refer to evolution scenarios. Type of architectural reasoning applies to architectural descriptions and runtime issues refer to runtime properties of a system. Criticality, importance, and quality attributes from Williams and Carver [20] cannot be determined in advance.

A detailed application of a subset of the evolution criteria (cp. Buckley et al. [19]) to CoCoME is described in Vogel-Heuser et al. [2]. In the following, we describe all relevant evolution criteria based on the example of evolution scenario S1. In S1, a web shop is added to CoCoME to fulfill emerging user requirements. For this purpose, the software architecture needs to be changed to include the new web shop components and the corresponding communication interfaces at design-time. As this change includes adding new components and interfaces and adapting the existing ones, it is at a coarse granularity level. Since this change can be foreseen during the design-time, it is an anticipated change. S1 is a perfective evolution due to changing requirements. In contrast, S2 is an adaptive evolution, as the enterprise server and the database of CoCoME are migrated to the cloud in order to reduce costs and improve the flexibility and scalability. S3 is a self-adaptive evolution, since CoCoME is automatically changed during operations.

B. Classification of selected PPU evolution scenarios

In the following, three evolution scenarios (scenario 2, 4a and 15c) of the PPU are described and characterized according to the criteria of the information system's domain (cp. Table II). For all PPU scenarios changes are conducted manually (*degree of automation*) and as there are no formal models for the change scenarios they are all to be categorized as adhoc (*degree of formality*).

In scenario 2 the PPU is to be retrofitted with an inductive sensor at the stack in order to be able to distinguish and therefore process metallic workpieces (*cause of evolution)*. The *type of evolution* is perfective as it is an enhancement of the system required by the customer, the *means of evolution* is adaptive and the *evolution history* is sequential. The addition of an inductive sensor affects all three disciplines involved and the following *entities*: the mechanics of the stack have to be modified to mount the fixture of the sensor. The sensor outputs need to be connected to the automation hardware, resulting in additional wiring and additional power supply. In case the input terminal of the fieldbus system is already fully occupied a new one has to be added. For being able to actually use the sensor, an additional input in the software has to be configured, and the logic of the function block controlling the stack has to be enlarged to provide the output if the workpiece is metal or not. The *scope* of this change scenario refers to mechanics, automation hardware and software of the stack. The *granularity* of the change will be discussed in the following to highlight the difference of the granularity levels compared to IS. Regarding its scale, a sensor by itself is a fine grained entity in an aPS. However, as the interface of a sensor matters in the context of aPS, it is to be seen as a basic component (black box) and thus to be classified as a medium grained change regarding the automation hardware. For the mechanics and the software, the same applies as the mechanical fixture and the new function block for the sensor both are basic components.

In Scenario 4a the sensors detecting the position of the crane are replaced. The *cause of evolution* in this case is a desired reduction in the error rate caused by sensor contamination and is to be classified as a corrective *type of evolution*. The *affected entities* of this sequential evolution scenario are the sensors and the *scope* of the evolution is the automation hardware of the crane as the sensors are replaced by identical ones. As just described in scenario 2, a sensor is classified as a basic component and therefore it is a matter of a change of medium *granularity*, although the change effort in this scenario is much lower compared to the one of scenario 2. The change is only realized in one discipline and the sensors merely have to be exchanged, but according to the classification from the domain of IS both scenarios are to be categorized identically regarding their granularity. Therefore, future categories need to be adapted to include the linkage of changes across disciplines. One idea is to remain with three different categories of granularity, but in case there are changes in two or more disciplines these cross-disciplinary links increase the level of granularity by one.

An alteration of the control voltage, which is to be categorized as a customer requirement (i.e. country specific due to national regulations), is the *cause of evolution* in scenario 15c and it is again a perfective *type of evolution*. In contrast to the previous scenarios it is a matter of a parallel *evolution history*. Due to an alteration of the control voltage all bus components may be replaced and sometimes also sensors and actuators, which are incompatible to the required voltage affecting only the automation hardware at a first glance. But in case a sensor or actuator supplier needs to be changed the fixture and the software may be affected, too. According to these *affected entities*, the *scope* of this scenario encompasses the mechanics and the automation hardware as well as the software of all subsystems. Regarding the mechanical fixtures, which have to be modified and are to be classified as components, this change is of medium *granularity*. The same applies for the changes in automation hardware, as sensors and actuators are components as well, and the resulting change in the software, i.e. a function block. Again, this change would be classified into the same level of granularity as the other scenarios, but results in a tremendously bigger maintenance effort. Therefore, we can summarize that the classification of granularity from information systems does not fit the automation domain, because it does not allow to reflect the maintenance effort; in other words, the resulting information is too coarse grained to be beneficial.

Up to now only sequential evolution and offline change, i.e. change during design time, have been focused on. As in aPS five levels of software granularity have been identified [27] from several industrial case studies, the level of software granularity shall be enlarged as well. There are partial proposals for an architectural system model for aPS, but the entire model is also missing.

V. Results and Outlook

The paper introduced an already established and beneficial classification of change effort in case of software maintenance from the domain of IS and applied the fundamental characteristics to the domain of aPS. Hereby, the approach was enlarged by two more system aspects or disciplines, i.e. mechanics and electric/automation hardware. The weaknesses are explained in detail for three scenarios of a simple open source lab size example of a pick and place unit. The classification of granularity needs to be elaborated to allow a more fine grained classification on the one hand and to include cross-disciplinary aspects, e.g. by increasing the level of granularity in case more than one discipline is included, on the other hand. This would help to compare in our future work two or more alternative ways to maintain a system or to implement changes in software and facilitate the decision, whether a change should best be conducted in software or mechanics. This way, the alternative that is most time and cost efficient or long term beneficial according to architectural decisions can be chosen.

Acknowledgment

This work was partially supported by the DFG (German Research Foundation) under the Priority Programme SPP 1593: Design For Future – Managed Software Evolution (VO 937/29-1 and RE 1674/12-1).

References

[1] R. Heinrich, S. Gärtner, T.-M. Hesse, T. Ruhroth, R. Reussner, K. Schneider, B. Paech, and J. Jürjens, "The CoCoME platform: A research note on empirical studies in information system evolution," *International Journal of Software Engineering and Knowledge Engineering*, vol. 25, no. 9, World Scientific, 2015.

[2] B. Vogel-Heuser, S. Feldmann, J. Folmer, S. Rösch, R. Heinrich, K. Rostami, and R. Reussner, "Architecture-Based Assessment and Planning of Software Changes in Information and Automated Production Systems State of the Art and Open Issues," in *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 687–694, 2015.

[3] R. Heinrich, "Architectural run-time models for performance and privacy analysis in dynamic cloud applications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 4, pp. 13-22, 2016.

[4] R. Birkhofer, G. Feldmeier, J. Kalhoff, C. Kleedörfer, M. Leidner, R. Mildenberger, M. Mühlhause, J. Niemann, R. Schrieber, J. Wickinger, M. Winzenick, and M. Wollschläger, *Life-Cycle-Management für Produkte und Systeme der Automation Ein Leitfaden des Arbeitskreises Systemaspekte im ZVEI Fachverband Automation*. Frankfurt, M.: Zentralverb. Elektrotechnik- und Elektronikindustrie Fachverb. Automation, 2010.

[5] *Engineering of industrial plants - Evaluation and optimization - Subject processes*, VDI/VDE 3695, 2010.

[6] B. Vogel-Heuser, A. Fay, I. Schaefer, and M. Tichy, "Evolution of software in automated production systems: Challenges and research directions," *Journal of Systems and Software*, vol. 110, pp. 54–84, 2015.

[7] B. Kirwan and L. K. Ainsworth, *A Guide To Task Analysis The Task Analysis Working Group*. London, Philadelphia: Taylor & Francis, 2003.

[8] B. W. Boehm, *Software cost estimation with COCOMO II*. Upper Saddle River, NJ: Prentice Hall PTR, 2000.

[9] D. J. Paulish, *Architecture-centric software project management A practical guide*. Boston: Addison-Wesley, 2002.

[10] R. Carbon, K. Fraunhofer, D. Rombach, P. Liggesmeyer, and F. Bomarius, *Architecture-centric software producibility analysis*. Stuttgart: Fraunhofer IRB Verlag, 2012.

[11] M. Naab, *Enhancing architecture design methods for improved flexibility in long-living information systems*. Berlin, Heidelberg: Springer, 2011.

[12] D. Garlan, J. M. Barnes, B. Schmerl, and O. Celiku, "Evolution styles: Foundations and tool support for software architecture evolution," in *European Conference on Software Architecture*, pp. 131–140, 2009.

[13] P. Clements, R. Kazman, and M. Klein, *Evaluating software architectures Methods and case studies*. Boston: Addison-Wesley, 2002.

[14] P. O. Bengtsson, N. Lassing, J. Bosch, and H. van Vliet, "Architecture-level modifiability analysis (ALMA)," *Journal of Systems and Software*, vol. 69, no. 1, pp. 129–147, 2004.

[15] K. Rostami, J. Stammel, R. Heinrich, and R. Reussner, "Architecture-based Assessment and Planning of Change Requests," in *International ACM SIGSOFT Conference on Quality of Software Architectures*, pp. 21–30, 2015.

[16] S. Herold et al., "CoCoME - The Common Component Modeling Example," in *LNCS*, vol. 5153, pp. 16-53, 2008.

[17] R. Heinrich, K. Rostami, and R. Reussner, "The CoCoME platform for collaborative empirical research on information system evolution," *Karlsruhe Reports in Informatics*, 2016.

[18] B. P. Lientz and E. B. Swanson, *Software maintenance management A study of the maintenance of computer application software in 487 data processing organizations*. Addison-Wesley, 1980.

[19] J. Buckley, T. Mens, M. Zenger, A. Rashid, and G. Kniesel, "Towards a taxonomy of software change," *Journal of Software Maintenance and Evolution Research and Practice*, vol. 17, no. 5, pp. 309–332, 2005.

[20] B. J. Williams and J. C. Carver, "Characterizing software architecture changes: A systematic review," *Information and Software Technology*, vol. 52, no. 1, pp. 31–51, 2010.

[21] P. Jamshidi, M. Ghafari, A. Ahmad, and C. Pahl, "A Framework for Classifying and Comparing Architecture-centric Software Evolution Research," in *European Conference on Software Maintenance and Reengineering*, pp. 305–314, 2013.

[22] N. Chapin, J. E. Hale, K. M. Khan, J. F. Ramil, and W.-G. Tan, "Types of software evolution and software maintenance," *Journal of Software Maintenance and Evolution Research and Practice*, vol. 13, no. 1, pp. 3–30, 2001.

[23] U. Katzke, B. Vogel-Heuser, and K. Fischer, " Software Engineering/Modeling - Analysis and state of the art of modules in industrial automation," *Automatisierungstechnische Praxis*, vol. 46, no. 4, pp. 23–31, 2004.

[24] B. Vogel-Heuser, C. Legat, J. Folmer, and S. Feldmann, *Researching Evolution in Industrial Plant Automation Scenarios and Documentation of the Pick and Plance Unit*. München: mediaTUM, 2014.

[25] R. Heinrich, S. Gärtner, T.-M. Hesse, T. Ruhroth, R. Reussner, K. Schneider, B. Paech, and J. Jürjens, "A Platform for Empirical Research on Information System Evolution," in *International Conference on Software Engineering and Knowledge Engineering KSI Research Inc.*, pp. 415–420, 2015.

[26] R. Reussner, *Modeling and Simulating Software Architectures The Palladio Approach*. Cambridge, London: MIT Press, 2016.

[27] B. Vogel-Heuser, J. Fischer, S. Rösch, S. Feldmann, and S. Ulewicz, "Challenges for maintenance of PLC-software and its related hardware for automated production systems: Selected industrial Case Studies," in *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 362–371, 2015.