A Classification Schema for Development Technologies

Davide Taibi, Christiane Plociennik
University of Kaiserslautern
Kaiserslautern, Germany
{taibi, christiane.plociennik}@cs.uni-kl.de

Laurent Dieudonné
Liebherr-Aerospace
Lindenberg, Germany
laurent.dieudonne@liebherr.com

***Abstract*— Software and hardware development organizations that consider the adoption of new methods, techniques, or tools often face several challenges, namely to: guarantee process quality, reproducibility, and standard compliance. They need to compare existing solutions on the market, and they need to select technologies that are most appropriate for each process phase, taking into account the specific context requirements. Unfortunately, this kind of information is usually not easily accessible; it is incomplete, scattered, and hard to compare. Our goal is to present a case study on a classification schema we applied on the avionic domain to help decision makers to easily find, compare and combine existing methods, techniques, and tools based on previous experience. The results show that the schema helps to transfer knowledge between projects, guaranteeing quality, reproducibility, and standard compliance.**

Keywords—process improvement; technology classification; technology selection; tool selection; method selection; process configuration.

I. INTRODUCTION

The software and hardware market is evolving continuously and companies that develop software or hardware need to keep improving their processes by introducing new technologies, in order to be able to keep pace with other competitors on the market.

Finding a product development process that guarantees quality and reproducibility often takes years. Moreover, in certain domains, such as avionics, the process must comply with a set of standards, such as DO-178 [13].

The introduction of a new technology may break the consistency and standards compliance of the process. To limit this risk, two major aspects must be considered. First, the objectives and prerequisites for each process step must be fully documented and structured. Second, the contribution of each method and tool intended to be used must be limited to the objectives set by each domain process activity and their role in each process step must be fully described.

A structured framework, enabling the classification of the technologies in the process activities would speed up the integration of new technologies and contribute to guaranteeing compliance with the company processes.

To facilitate the classification of technologies, the Reference Technology Platform (RTP) has been developed. RTP is a set and arrangement of methods, workflows, and tools that allow interaction and integration on various levels in order to enable efficient design and development of (complex) systems [20].

In the context of the ARAMiS project [16], a classification schema based on the RTP has been developed. It classifies technologies along two dimensions: abstraction levels and viewpoints.

In this paper, we present a use case on the application of this schema in the avionic domain. Moreover, we also introduce an implementation of the schema we developed: the Process Configuration Framework Tool (PCF) [8].

The results of this work suggest that the classification provides a useful framework for decision makers and allows them to base their decisions on previous experience instead of on personal opinions. Moreover, the classification allows them to guarantee process quality, reproducibility and standards compliance. Finally, it facilitates knowledge transfer from project to project or between employees.

The remainder of this paper is structured as follows: Section II describes related work; Section III introduces the classification schema, while Section IV describes the avionic use case. In Section V, we introduce the PCF tool and discuss the benefits of the schema in Section VI. Finally, we draw conclusions in Section VII and provide an outlook on future work.

II. RELATED WORK

In this section, we introduce the most common technology classification schemas.

An early work on technology classification is Firth et al. [19] from 1987. It classifies software development methods according to the *stages* of the development process (specification, design, and implementation) and the *view* (functional, structural, and behavioral). This schema is two-dimensional like our schema, and its *views* dimension is similar to our *viewpoints* dimension. However, the second dimension is rather different: Firth et al. focus on the process stages, while we map these onto the viewpoints dimension. Our second dimension is concerned with abstraction instead.

Another early work is the *Experience Factory*, published in the late 1980s [3] and updated in 1991 [4] and in 1994 [5]. Here, software development artifacts are described in so-called *experience packages* along with empirical evidence on how they have been used previously. The main goal of the Experience Factory is to provide a framework for software reuse to help software engineers make decisions based on company experience.

Compared to our work, the Experience Factory is a more general concept. In the Experience Factory, an object for reuse can be any software engineering artifact, including products, requirements documents etc. Furthermore, the Experience Factory does not provide a specific schema for

 ISBN: 978-1-61208-367-4

storing different technologies for reuse, and it does not include algorithms for searching or combining technologies. Another approach to technology classification developed in parallel to the later experience factory versions is the *C4 Software Technology Reference Guide* (C4 STR), a catalog containing more than 60 technologies.

Compared to our work, the C4 STR provides a huge number of technologies in its schema. Nonetheless, compared to our schema, the attributes it uses are not as detailed and there is neither a reference to context nor to the impact.

The C4 STR was later merged with the Experience Factory approaches by Birk [5]. In the late 1990s, this evolved into a new concept of *experience management*. Based on this work, more publications evolving this schema and extending the Experience Factory idea [12] appeared.

Ploskonos [18] developed a classification schema for software design projects. The goal is to facilitate the adaptation of generic process descriptions and methods to individual processes. Design projects are classified into one of four groups: *Usability*, *Capability*, *Extension*, and *Innovation*. Each group is associated with certain process characteristics that help the user set up the actual process. This approach is narrower than ours: It focuses on classifying processes according to the project type, omitting other characteristics such as project size or domain.

III. THE CLASSIFICATION SCHEMA

In this section, we introduce the classification schema applied, as foundation for our case study. The schema is aimed at providing a complete engineering tool chain for collecting and integrating technologies to support the activities of a structured development process.

The paper addresses the development of big, complex projects in the industry, which are spread out over several years and occupy many employees.

Industries usually work with requirements-based process models planning the different baselines in order to ensure the accomplishment of these baselines on time via different phases of realization. Each phase and each step of the processes usually produces artefacts used as inputs for the next phase(s) or step(s). These models are derived from, or include, the V-Model [23], which is traditionally used inside the iterations made for the accomplishment of each baseline. Additionally to the iterations, other concepts like definition of phases, definition of objectives, periodical assessments, definition of roles, forward and backward traceability, etc. are traditionally used in these development processes, and have widely inspired current agile methodologies, like SCRUM [23].

The schema presented in this paper represents a generic development model covering the industry development processes. The instances of the generic development model are naturally dependent on the industry development standards and on the company itself.

The information provided in this schema, enables decision makers to find the most appropriate technology based on their interaction and integration on various levels. This contributes to the efficient design and development of complex systems. Furthermore, the schema can give an overview of methods and tools used in past projects. Via the different planning phases, assessment meetings and accomplishment summaries inherent to the industry processes and performed periodically during each project development, the decisions made, the quality and special uses of the tools, methods and technologies, can be collected during the whole development life cycle of each project. This contributes to building a knowledge database, addressing both best practices and pitfalls, adapted to the company development processes. Hence, new projects do not have to start from scratch, but can benefit from previous experience. The same applies for new employees: The schema can help them to familiarize themselves quickly with the methods and tools available for each phase of the development process. Thus, the schema facilitates knowledge transfer inside a company.

The schema can be represented as a matrix with viewpoints as columns and abstraction levels as rows. The viewpoints of the classification are defined as "*Requirements*", "*Functional*", "*Logical*", and "*Technical*". These viewpoints can be mapped to the three phases of the development process where the requirements viewpoint coincides with the requirements capture phase, the functional and logical viewpoints are related to the design phase, and the technical viewpoint is related to the construction or implementation phase (see Figure 1).

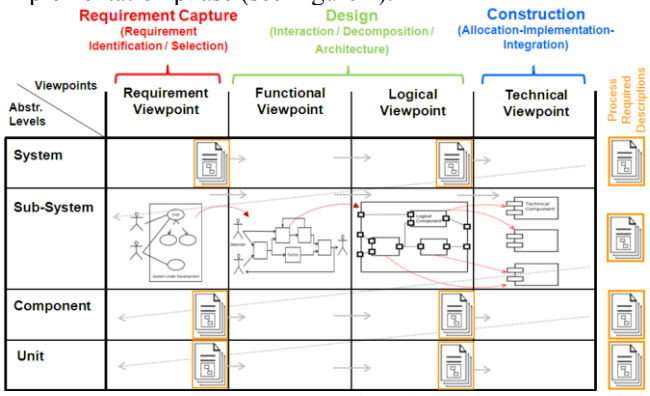

Figure 1. Generic representation of our classification schema.

In the generic version of the schema, the abstraction levels resemble the decomposition of the system into sub-systems, components, and sub-components or units (see Figure 1). For specific application domains (e.g., automotive, avionics and railways), a different, domain-specific set of abstraction levels can be defined. For example, in the avionics domain, abstraction levels are defined as *"Aircraft"*, *"System"*, *"Equipment"*, and *"Item"* (see Figure 2). Each cell of the schema represents a step of the product development process that must be performed starting from the topmost and leftmost cell to the rightmost, as shown by the arrows in Figure 1.

The output of each step leads to the realization of artifacts contributing directly to the fulfillment of the process objectives required by the domain or indirectly by focusing artifacts needed by other cells, which are inputs for later

 ISBN: 978-1-61208-367-4

steps. The objectives specified by the domain process depend on the development phase and the abstraction level.

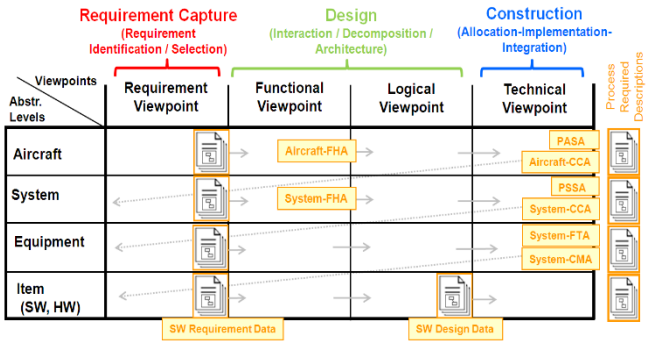

Figure 2. Example of classification schema for the avionics domain.

Starting at a given abstraction level, the requirements suitable for this abstraction level must first be captured in the requirement viewpoint. These filtered requirements are the outputs of this viewpoint and are necessary to start the design of the system. During the design phase, the function network determined in the functional viewpoint is needed first in order to perform decomposition and/or structuration of the identified functions, realized in the logical viewpoint. Once the objectives of the logical viewpoint have been achieved, the construction of the system can be started in the technical viewpoint. Iterations are possible, among others to introduce new requirements or to consider realization constraints appearing a posteriori that influence the system design.

At the end of an abstraction level, the requirements derived from the design and thus from the requirement viewpoint not being fulfilled at this abstraction level are used as a basis for the next abstraction level. They are captured in the requirement viewpoint at the new current abstraction level, where similar work as for the previous abstraction level is performed again.

To allow partial and iterative development, the transition from one cell to the next is controlled by a set of transition criteria. Transition criteria support the evaluation of the risks of starting the next development step although the objectives of the current step are only partially fulfilled. The current fulfillment of the objectives can then be controlled and will be realized after several iterations.

To fulfill the objectives of each matrix cell, the system and software engineers have to use methods that must mostly be supported by tools. Depending on the category of product to be developed, its requirements, the abstraction level, and the focus set in the current development iteration (e.g., which objectives are addressed), the methods and tools may differ, and the technology chain used can also be integrated differently. The transition criteria between the process steps must be supported by the methods as well.

IV. APPLYING THE CLASSIFICATION SCHEMA IN THE AVIONICS DOMAIN

In this section, we sketch an example of a use case of the classification schema in the avionics domain.

In the avionic industry, two main processes are defined and address two different aspects corresponding to the two branches of the V-Model: the *Development Process* and the *Integral Process* [14]. The combination of both main processes defines abstraction levels (Aircraft, System, Equipment/Item, Software, Hardware, etc.) and specific processes for each of them. Iterations can be done inside an abstraction level, or inclosing them. The overall resulting applicable development process can be summarized like the following suite of development phases, where the previous ones are required by the next ones: *Aircraft Requirements Identification*, *Aircraft Function Development*, *Allocation of Aircraft Function to Systems, System Requirements Identification*, *Development of System Architecture*, *Allocation of System Requirements to Items*, *Item Requirements Identification*, *Item Design* (corresponds to Software and Hardware Development, both having specific processes), *Item Verification*, *System Verification*, and *Aircraft Verification*.

These different phases can be well mapped onto the generic development model, among others by instancing the abstraction levels and by specifying the objectives of the viewpoints for each abstraction level, according to the company and project needs.

For example, at the system level, the *System Requirements Identification* corresponds to the *Requirement Capture Viewpoint*, the *Development of System Architecture* is realized via the *Functional* and *Logical Viewpoints*, the *Allocation of System Requirements to Items* belongs to the *Technical Viewpoint*, where the decision is taken on which technology will be involved to realized the *Items* (*Item Design* corresponds to Software and Hardware development). The *Verification* phases are realized in the *Technical Viewpoint* of corresponding abstraction levels, where the integration activity is performed. For each phase, objectives concerning safety assessments, validation, verification, etc. are defined via the *Integral Process* and should be met in order to move to the next phase, or must be accomplished during a next iteration. The same logic applies when moving to the next abstraction level.

The same principles apply for all the other abstraction levels. This is also true for the Software and Hardware development, but with different steps inside the phases and different objectives, because they are defined by specific processes specified in the avionics standards DO-178C [13] and the DO-254 [21].

We consider the development of a safety-critical system – a Flight Control System (FCS). We give an example on how the regular avionic development process, according to the civilian aircraft and systems development process guidelines ARP4754A [14], can be mapped on the classification schema (see Figure 2).

Here, we briefly introduce how to use the classification schema efficiently by describing the most important development process steps and their artifacts.

 ISBN: 978-1-61208-367-4

Based on the high-level aircraft requirements and design decisions, the requirements on the FCS must first be captured, expressed, and validated precisely (*requirement viewpoint*). The artifacts for this step are the functional and non-functional requirements that contain the goals of the system (e.g., "control the three axes of the aircraft: pitch, yaw, and roll"), the operational requirements (e.g., operational modes), the safety requirements (e.g., which criticality for which surface/axis), the high-level performance requirements (e.g., aircraft response time following cockpit control requests), etc. The requirement capture can be facilitated with use-cases, such as SysML/UML, or with requirements tools using structured text.

Once captured, the requirements must be validated, which is a transition criterion for proceeding to the next step. Different activities and requirements types are analyzed using different technologies, according to the avionics standards.

Based on these requirements, the behavior of the system is then analyzed and a functional architecture in the form of a network of the essential functions covering the major system functionalities must be formulated (*functional viewpoint*). An example of a major functionality at the system abstraction level is the altitude control via the pitch axis, which is realized by the elevator surfaces. Essential functions are those realizing the functionality and having an external interface with other parts of the system, for example actuator control, acquiring of the surface position, synchronization with the other surfaces, etc. For example, block definition diagrams from the SysML and signal flow diagrams are well suitable to model the functions network.

Once the definition of these functions and their related requirements is completed, a Functional Hazard Assessment (FHA) must be performed [14]. The FHA produces safety requirements and design constraints for the next design step which are necessary to make decisions about the decomposition and structuration of the functions in order to realize a suitable system design. In this next step (*logical viewpoint*), these essential functions are structured, completed, and/or decomposed in order to shape the components to be realized on this abstraction level – here named "logical components". The logical architecture determination is also efficiently supported by the SysML/UML technologies, and the behavior can be well designed via control flow diagrams, state machines, etc. Simulation technologies can be used to validate the interactions and behavior between the logical components, once they are correctly formalized.

Based on these components and their inherited requirements (the logical components are derived from the functions of the *functional viewpoint*, which are themselves derived from the requirements of the *requirement viewpoint*), technical solutions suitable for this abstraction level are identified or existing technical solutions are chosen (*technical viewpoint*). These technical solutions are called "technical components" in this paper. The requirements expressed by the logical components drive the selection of the technical components.

Iterations inside an abstraction level are feasible for introducing new requirements, or for increasing the reusability rate by considering already existing technical components. As a consequence, the structuring (decomposition and composition) of the logical components may be performed in a different way. A configuration management system is mandatory for managing the different alternatives and versions.

At the end of the technical viewpoint, different validation activities (part of the transition criteria) must be accomplished, like a Preliminary System Safety Assessment (PSSA), a preliminary common cause analysis (CCA), etc. [14] in order to validate the decisions made in the functional, logical, and technical viewpoints.

If the already existent technical components fulfill exactly the requirements expressed by the logical components mapped onto them, the work is completed and the associated requirements are considered as fulfilled. This is an ideal case of reusability and will probably not arise very often at higher abstraction levels such the Aircraft and the System levels, but may arise at the Equipment or Item level.

The technical components that do not exist yet or that do not completely fulfill the requirements expressed by the logical components mapped onto them, and the logical components that are still too complex to be allocated to a particular technical solution are both inputs for the next abstraction level. They express requirements that have not been fulfilled at the current abstraction level and must be dealt with at the next one. Thus, the work on the next abstraction level can start.

The traceability, required by avionics processes at the different abstraction levels, is performed 1) between the viewpoints of the same abstraction level and 2) between the abstraction levels. For this second case, the traceability is performed between the technical and logical viewpoints of a given abstraction level and the requirement viewpoint of the next abstraction level.

For example: For 1), the technical components (*technical viewpoint*) are assigned to the logical components (l*ogical viewpoint*) that drove their selection. For 2), on abstraction level AL, each technical component not already realized and each logical component that cannot be mapped to a technical component must be addressed on abstraction level AL-1. They express requirements to be captured in the *requirement viewpoint* of AL-1. The requirements expressed at the *Requirement viewpoint* of AL-1 are then linked to the requirements expressed by the corresponding technical and logical components from the abstraction level AL.

The other abstraction levels follow the same logic for each step with methodology objectives, process objectives and artifacts, and similar activities that need to be carried out. All of them can be well mapped in the classification schema.

For example, at the Aircraft abstraction level, similar process activities as for the system level are realized, like an FHA, a Preliminary Aircraft Safety Assessment (PASA), and a CCA. For the equipment abstractions level, a Fault Tree Analysis (FTA) is required as well as a Common Mode Analysis (CMA), etc. For the software abstraction level, the

 ISBN: 978-1-61208-367-4

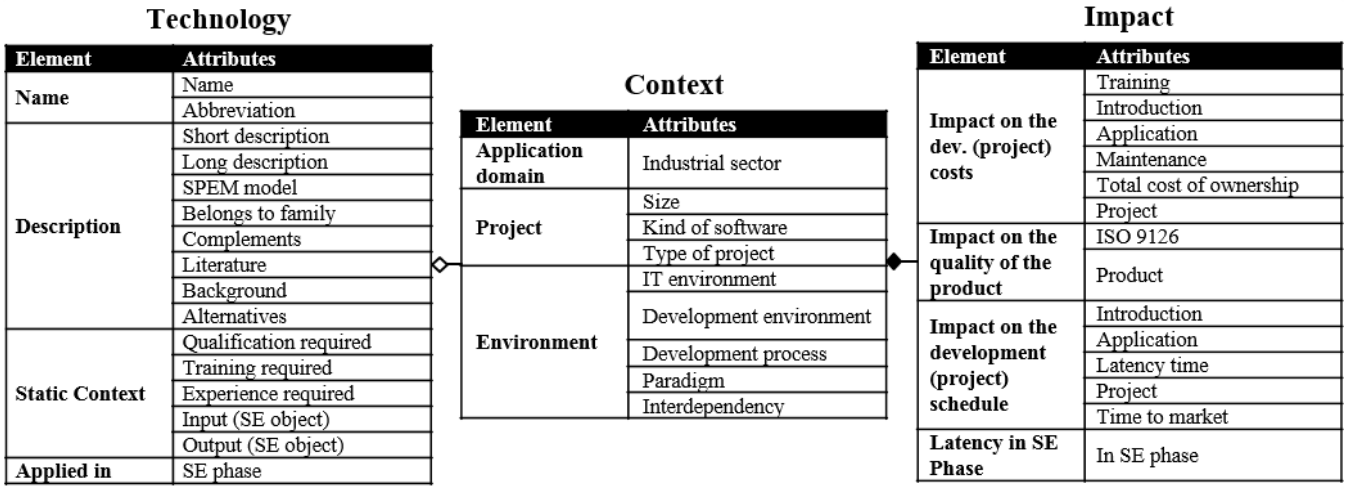

Figure 3. PCF Data Schema.

avionics standard DO-178 [13] defines different phases (called "processes", such as the Software Requirements Process and the Software Design Process) with several objectives requiring numerous artifacts, such as requirements and detailed design descriptions, validation and verification artifacts, etc., which can be performed by using different methods and tools (e.g., for verification: Classification Tree, Equivalence Partitioning, Cause-and-Effect Analysis), with each containing pros and cons, depending on the context of the current development.

Another issue that belongs to the top-down process explained here is that the reusability of existing solutions potentially fulfilling parts of the system also requires suitable and standardized methods and tools. Existing technical solutions may also consist of components developed outside the company, such as a microcontrollers, software libraries, etc. with other degrees of quality and using different processes. In any case, these existing solutions need to be completely and suitably characterized and must be integrated efficiently into the development process.

However, reusability is not a separate activity that can be transposed directly as a technology that can be integrated into the schema. In fact, it influences different activities, such as the decomposition in the design phase at the *logical viewpoint*, the accurate characterization of the existing solutions and the deployment activity at the *technical viewpoint*, etc. All these aspects related to reusability must also be taken into account in these activities. For example, it should be possible to integrate a systematic deployment process and its related techniques as explained by Hilbrich and Dieudonné [15] into the schema via these activities. As an example for this case, the software applications that are to be mapped optimally onto electronic execution units (ECU) need to be decomposed and structured in a way that makes them well compatible with the capabilities of the ECUs in order to allow the use of a minimum number of ECUs. However, on the other hand, the ECUs must be formalized completely and their description must be easily accessible by the system and software architects in order to influence the system design and to be correctly selected during deployment. In ARAMiS, we also provide a template for formalizing multicore processor capabilities in a form and on an abstraction level that can be used by system and equipment engineers. The formalization must be performed by the software and hardware engineers who design the ECUs. A noticeable advantage is to be able to validate per analysis or per simulation more aspects of the system, like the timing reactions, or the resource consumption.

These activities related to reusability are scattered across different cells of the matrix. At present, they need to be taken care of by the system designer. It would be helpful if they could be better integrated into the chain of methods and tools in the future.

V. IMPLEMENTING THE CLASSIFICATION SCHEMA IN PCF

The proposed schema has been implemented as a web application in the PCF tool [8]. PCF allows users to search for technologies based on abstraction levels and viewpoints as defined in the schema. Furthermore, PCF adds two more aspects to provide information about previous experience using a specific technology: *Context* and *Impact*. Hence, the data schema in PCF is based on three models as defined in [9] (as shown in Figure 3):

- *Technology*: includes a set of attributes for describing a technology in as much detail as possible.
- *Context*: includes information on the context, such as application domain, project characteristics, and environment in which the respective technology has been applied.
- *Impact:* includes previous experience on applying a specific technology in a specific context.

The PCF tool contains a search feature that allows users to search for technologies based on the attributes defined in the models in Figure 3. This enables the user to search for technologies used in projects with specific characteristics, e.g. projects fulfilling a certain industrial standard.

Basic use cases for PCF, as shown in Figure 4, are:

- Search for a technology based on context requirements (not mandatory)

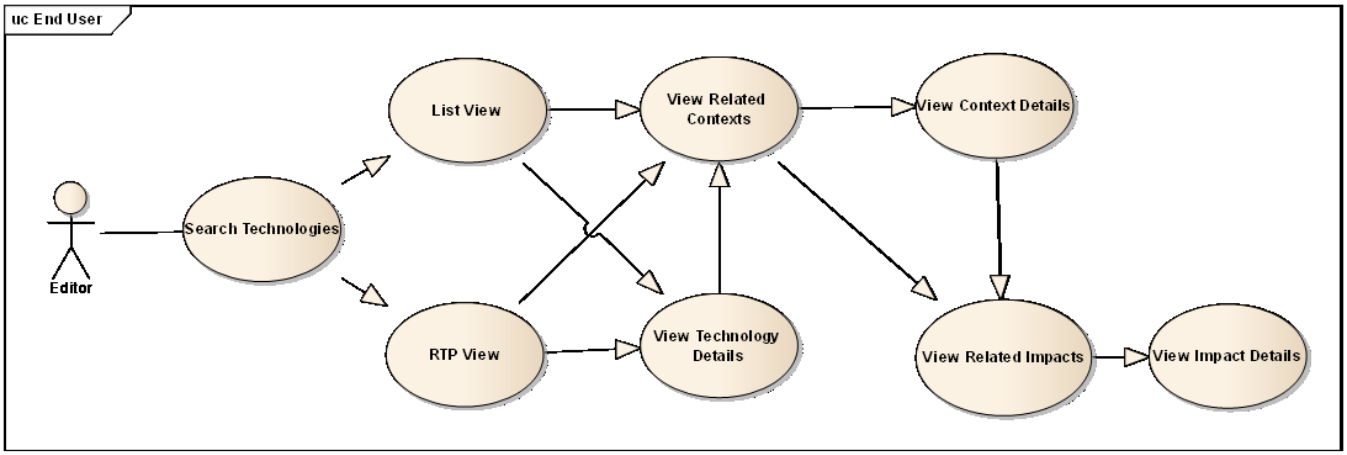

Figure 4. PCF Use Case.

- List view
- Matrix view

- View details for a technology
- View related context
- View details for a context
- View related impacts
- View details for a related impact

Moreover, PCF implements the schema for different domains (avionics, automotive, and railways).

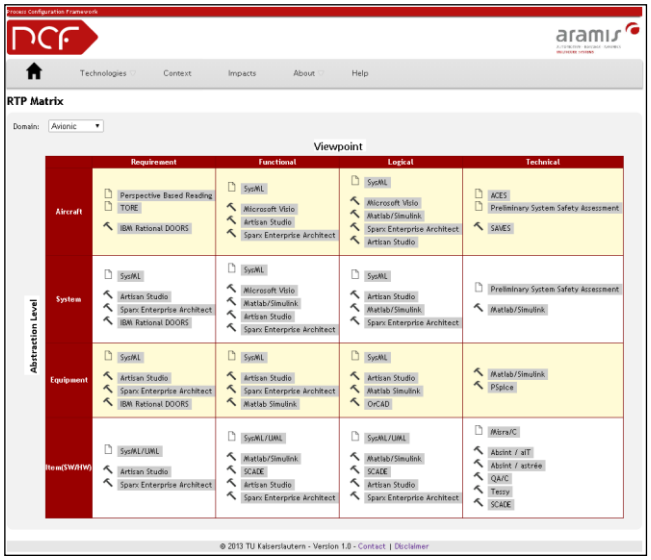

Figure 5. An example of the schema in the avionic domain implemented in PCF.

Figure 5 shows an example of the schema represented in PCF for the avionics domain. This figure includes the methods mentioned in the use case or directly the tools realizing them, as well as several other technologies for the avionics domain in addition to those mentioned above. In this version of the tool, we do not consider interoperability issues. The next version of the tool will address the challenge of interoperable tool chains.

VI. BENEFITS

The classification schema provides benefits for different people working in software projects, especially for project managers, software engineers, and technology providers (software and hardware vendors).

The use case indicates that, from the point of view of software engineers and decision makers, the classification schema provides an effective platform for searching for existing technologies. For industry domains strongly based on process based development, it also provides a toolbox for accurately specifying the use of each technology for rigorous process steps.

The main benefit for the ARAMiS project was that creating the classification schema for the avionics domain helped us to improve the schema. Several changes to the schema have been suggested based on issues raised during the application of the schema concept in practice. Another major benefit for the ARAMiS project was the identification and specification of methods and tools for improving the integration of multicore processors for safety-critical domains.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a use case reporting on the usage of a classification schema in the avionics domain and its implementation in the PCF tool.

The schema is aimed at collecting and integrating methods and technologies to support the activities of a structured development process. It allows decision makers to find the most appropriate technology based on their interaction and integration on various levels to enable efficient design and development of complex systems.

The schema provides a matrix representation of the development activities classified into viewpoints and abstraction levels that enables users to easily search for the most appropriate technologies throughout the whole development lifecycle.

The use case shows that the schema helps process managers to keep track of the technologies used in previous projects and to maintain traceability throughout the whole

 ISBN: 978-1-61208-367-4

process. Moreover, the schema can be useful to enable knowledge transfer inside the company.

Supported by the ARAMiS project and its partners, future work will include the collection of existing technologies to create a baseline for the platform. Moreover, we are planning to run an empirical study to validate the effectiveness of the schema.

ACKNOWLEDGMENT

This paper is based on research carried out in the ARAMiS project, funded by the German Ministry of Education and Research (BMBF O1IS11035Ü).

REFERENCES

[1] A. Rajan and T. Wahl, “CESAR - Cost-efficient methods and processes for safety-relevant embedded systems”, Springer, 2013, ISBN: 978-3-7091-1386-8.

[2] K. Pohl, H. Hönninger, R. Achatz, and M. Broy, “Model-based engineering of embedded systems - The SPES 2020 Methodology”, Springer, 2012, ISBN: 978-3-642-34614-9.

[3] V. Basili and D. Rombach, “Towards a comprehensive framework for reuse: A reuse-enabling software evolution environment”, Technical Report, University of Maryland, 1988.

[4] V. Basili, D. Rombach, “Support for comprehensive reuse”, Software Engineering Journal, vol. 6, Sep. 1991, pp. 303-316, ISSN: 0268-6961.

[5] V. Basili, G. Caldiera, and D. Rombach, “Experience factory”, In: Encyclopedia of Software Engineering, Marciniak, John J., Ed., New York: Wiley, pp. 469-476, 1994.

[6] A. Birk, “A knowledge management infrastructure for systematic improvement in software engineering”, PhD dissertation, Stuttgart, Fraunhofer IRB Verlag, 2000.

[7] K. Schneider, J.P. Hunnius, and V. Basili, “Experience in implementing a learning software organization”, IEEE Softw., vol. 19, May 2002, pp. 46-49.

[8] P. Diebold, L. Dieudonné, and D. Taibi, “Process configuration framework tool”, Euromicro Conference on Software Engineering and Advanced Applications 2014, in press.

[9] P. Diebold, “How to configure SE development processes context-specifically?”, 14th International Conference on Product-Focused Software Process Improvement (PROFES 2013), Springer, Jun. 2013, pp. 355-358, ISSN: 0302-9743.

[10] P. Diebold, C .Lampasona, and D. Taibi, “Moonlighting Scrum: An agile method for distributed teams with part-time developers working during non-overlapping hours”, Eighth International Conference on Software Engineering and Advances, IARIA, Oct. 2013, pp. 318-323, ISBN: 978-1-61208-304-9.

[11] A. Jedlitschka, N. Juristo, and D. Rombach, "Reporting experiments to satisfy professionals' information needs", Empirical Software Engineering, 2013, doi: 10.1007/s10664-013-9268-6. [Online]. Available from: http://publica.fraunhofer.de/documents/N-266529.html. Last access 2014.07.21.

[12] A. Jedlitschka, D. Hamann, T. Göhlert, and A. Schröder, “Adapting PROFES for use in an agile process: An industry experience report”, Sixth International Conference on Product-Focused Software Process Improvement (PROFES 2005), Springer, Jun. 2005, pp. 502-516, ISSN: 0302-9743, ISBN: 3-540-26200-8.

[13] RTCA DO-178C, “Software considerations in airborne systems and equipment certification”, Dec. 2011.

[14] SAE ARP4754 Rev. A, “Guidelines for development of civil aircraft and systems”, Dec. 2010. Available from: http://standards.sae.org/arp4754a. Last access 2014.07.21.

[15] R. Hilbrich and L. Dieudonné, “Deploying safety-critical applications on complex avionics hardware architectures”, Journal of Software Engineering and Applications (JSEA), vol. 6, May 2013, pp. 229-235, ISSN: 1945-3124.

[16] ARAMiS project, “Automotive, railway and avionics multicore systems”. [Online]. Available from: http://www.projekt-aramis.de/. Last access 2014.07.18.

[17] SPES_XT project, “Software platform embedded systems”. [Online]. Available from: http://spes2020.informatik.tu-muenchen.de/spes_xt-home.html. Last access 2014.07.18.

[18] A. Ploskonos and M. Uflacker, “A classification schema for process and method adaptation in software design projects”, Tenth International Design Conference (DESIGN 2008), May 2008, pp. 219-228.

[19] R. Firth, W. G. Wood, R. D. Pethia, L. Roberts and V. Mosley., "A classification scheme for software development methods", Technical Report CMU/SEI-87-TR-041, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1987.

[20] P. Reinkemeier, H. Hille, and S. Henkler, “Towards creating flexible tool chains for the design and analysis of multi-core systems”, Vierter Workshop zur Zukunft der Entwicklung softwareintensiver, eingebetteter Systeme (ENVISION 2020), colocated with Software Engineering 2014 conference, Feb. 2014. [Online]. Available from: http://ceur-ws.org/Vol-1129/paper37.pdf. Last access: 2014.07.21.

[21] RTCA DO-254, “Design Assurance Guidance for Airbone Electronic Hardware”, Apr. 2000.

[22] K. Forsberg and H. Mooz, “The Relationship of System Engineering to the Project Cycle”, First Annual Symposium of National Council on System Engineering, Oct. 1991, pp. 57-65.

[23] K. Schwaber and M. Beedle, “Agile software development with Scrum”, Prentice Hall, 2002, ISBN: 0-13-067634-9.

 ISBN: 978-1-61208-367-4