Safety Patterns in Model-Driven Development

Timo Vepsäläinen, Seppo Kuikka
Tampere University of Technology
Dept. of Automation Science and Engineering
Tampere, Finland
{timo.vepsalainen, seppo.kuikka}@tut.fi

Abstract— **Design patterns capture named solutions to recurring challenges in development work. With an appropriate, non-restrictive tool support, design patterns could also improve the documentation value of models in model-driven development. This paper extends the design pattern modeling approach of UML Automation Profile with safety-related information and suggests the use of patterns in models to document safety aspects. The modeling concepts are tool supported. In the paper, the concepts are used for exporting safety-related documentation. The documentation can be used to guide the selection of development techniques as well as to perform consistency checks with respect to safety integrity levels that are required from modeled applications.**

Keywords-Model-Driven Development; Design Pattern; Safety.

I. INTRODUCTION

Design patterns document solutions to recurring challenges in design and development work. As a concept, design pattern was introduced in the work of Alexander [1][2] related to building architectures. In software development, design patterns began to gain popularity after the publication of the Gang of Four (GoF) patterns [3] that were targeted to object oriented software engineering. Support for the use of patterns was also developed to Unified Modeling Language (UML). Today, UML is the de-facto software modeling language. With domain specific profiles, UML is also the modeling basis of many Model-Driven Development (MDD) approaches. However, the support for design patterns in UML is still focused on describing contents of UML Classes.

The idea of MDD is to use models as the primary engineering artefacts during the development of software systems. Models describe the systems and applications from different points of view and on different abstraction levels. In MDD, the development often starts from high abstraction level models, e.g., Computation Independent Models (CIM) as in Model Driven Architecture (MDA) [4]. Model transformations are used between the models to ensure their consistency and to produce refined models based on the earlier ones. Models also document the developed systems. However, in specific application domains the required information content of documentation is governed by regulations and standards, in addition to development needs.

Safety-related systems and applications constitute such a domain. The development process of safety applications as well as solutions and techniques to be used during the process is governed by standards, e.g., IEC 61508 [5]. In addition to using standard-compliant techniques, a developer of such a system must be able to prove the compliance of it. This is where the relevant documentation is needed.

The use of MDD to safety system development has been suggested by few researchers and even less MDD has been taken to industrial practice. The reason is not that safety standards would not allow the use of MDD techniques. Instead, for example "automatic software generation" is recommended as an architecture design technique by IEC 61508 [5]. Possible explanations for the scarce use of MDD techniques in the application area are, however, the strict documentation requirements. It is possible that given the strict requirements, MDD has not been seen to offer possibilities to improve the efficiency of the development.

The purpose of this paper is to extend a design pattern modeling approach of UML Automation Profile (UML AP) [6] to safety patterns. Safety patterns are design patterns that are applicable for safety-related systems and include additional information related to safety. They can be used by exporting documentation from models of the developed systems in which the patterns are used. The documentation generation is intended to facilitate development work by: 1) supporting traceability between applicable safety solutions and their use in systems, 2) enabling verification of safety levels of patterns in comparison to required safety levels and 3) guiding the selections of techniques and solutions.

The rest of this paper is organized as follows. Section 2 reviews work related to design patterns and use of design patterns in models and model-driven development. Section 3 recapitulates the recent pattern-related work that is extended in the paper. Sections 4 and 5 present the safety-related extensions to the pattern concepts and the developed tool support, respectively. Before conclusions, Section 6 discusses the work and the relevance of safety aspects in control system development in general.

II. RELATED WORK

Support for using design patterns in UML models is in the language based on Collaboration and CollaborationUse [7] concepts that are suitable for presenting patterns inside

 ISBN: 978-1-61208-367-4

UML Classes. The concepts have been developed along the language itself from parameterized collaborations that were utilized in, e.g., [8]. In addition to the standard approach, however, many tool vendors have developed additional pattern support in a more ad hoc manner. For example, MagicDraw [9] enables the specification of model element templates and copying the templates to models to instantiate patterns. Without pointing out pattern instances, however, the information on the occurrences is endangered to vanish.

To enable precise but practical use of patterns in UML, France et al. [10] have developed a pattern modelling approach using UML. Precise specification of pattern solutions is seen to enable tool support for building solutions from pattern specifications and for verification of the presence of patterns in design. In the approach, an overall pattern specification consists of a structural pattern specification specifying the class diagram view of the solution, and a set of interaction pattern specifications that specify the interactions in the pattern solutions.

Approaches to apply and evolve design patterns to UML models have also been developed with use of model transformations [11][12][13][14] using Query/View/Transformation (QVT) and Extensible Stylesheet Language Transformations (XSLT) techniques. Detection of design patterns in models, on the other hand, has been studied for example with use difference calculation [15], graph matching [16], graph similarity scoring [17], as well as graph decomposition and graph isomorphism [18].

In the approach of the authors, the novelty is neither in the approach to transform patterns into design nor in detecting pattern instances. Instead, a starting point in the work is that uses of patterns are design decisions that should be deliberately documented by marking the patterns. On the other hand, attention is paid to the questions how the pattern markings could be used to produce documentation in general and in safety-related application development in particular.

For safety-related systems, design patterns have been specified, for example, related to redundancy. In [19], Douglass presents 4 patterns to implement redundancy or redundancy-like behavior so that a task is performed in different channels or that another computing channel is used to observe the behavior of the main channel. Also IEC 61508 [5] in the 6th part of it presents several M out of N solutions in which the idea is to perform a calculation redundantly and to use voting to acquire a reliable result for it.

In the tables of recommended techniques and measures for software architecture design (annex A), IEC 61508 [5] also refers to a wide range of solutions that already have corresponding patterns in pattern literature. For example, the standards suggest the use of (different kinds of) redundancy [19], backward recovery (from faults) [20][21] and cyclic program execution [19]. Another example on use of patterns in the domain is related to documenting recurring arguments of safety cases in order to systematically collect and gain benefit from arguments of previous projects [22].

MDD of safety systems has been studied in the DECOS project [23] that is targeted to development of both critical and non-critical functions of embedded control systems. In the approach, the preferred means for specifying application functionality is Safety-Critical Application Development Environment (SCADE) which is based on formally defined data flow notation and enables simulation at model level and code generation.

UML based modelling and development of safety applications has also been facilitated with UML profiling techniques. In [24] the approach is based on extracting key concepts of a safety standard, RTCA DO-178B, to stereotypes with which it is possible for software developers to include safety-related concepts and properties in models. It can be assumed that such models suit well also for the purpose of producing documentation. However, we regard the work presented in this paper as an important complement to the approach. Whereas UML stereotypes are applied to single modelling elements, with patterns it is possible to link several elements in designs to patterns and roles of them. This is needed in order to characterize how a number of elements are used together to perform a task.

III. NEED FOR PATTERNS IN MDD

The key concept of MDD is to shift the development efforts from written documents to models that are used throughout the development process. For special purposes, e.g., safety system development, it could be possible to maintain separate documents. However, that would require additional work and could significantly reduce the potential to benefit from MDD. In a sense, it would also be against the central idea in MDD. A more appropriate approach would be to include the documentation in the models, in the first place.

A possible challenge in this objective is that models, in general, tend to be more applicable for representing solutions than rationale behind them. For example, many of the basic concepts of UML are similar to concepts of object oriented programming languages. UML models can be well used to answer the question how to implement, e.g., a class or a program. In the MDD context, it is even possible to generate code from models to avoid the manual programming work. However, information on why something has been designed in the way it has, is often missing. This information could be crucially important for, e.g., quality assurance and maintenance purposes.

Design patterns are a possible solution to improve the situation. Patterns document named, proven solutions that are well-known among developers and suited for solving recurring challenges and tasks. They are structured so that they consist of named parts that have responsibilities in the solutions. The solutions that patterns include may have crucial advantages. The use of design patterns and pattern instances in MDD and models could thus increase the value of models significantly. Patterns could 1) indicate the use of standard solutions in systems and specifications, 2) mark potential challenges (that are treated with the patterns), 3) make design more understandable (because of the use of the known solutions) and 4) clarify the roles of model elements in design, just to name a few benefits. In specific application areas, e.g., safety system development, the use of patters could even automate tasks and checks that are currently performed manually.

 ISBN: 978-1-61208-367-4

A. Design Patterns in UML

In UML, pattern definitions and pattern instances are defined with the Collaboration and CollaborationUse concepts of the language, respectively. Similarly to the Class concept, Collaboration extends the StructuredClassifier and BehavioredClassifier concepts. A pattern definition is in the language a set of cooperating participants that are Properties of a Collaboration. In a similar manner Properties can be owned by Classes. The features that are required from the participants are defined by the Classifiers that are used as types of the Properties. Graphically Collaborations can be presented in composite structure diagrams in which participants of a pattern are connected with Connectors.

A CollaborationUse represents an application of a pattern to another Classifier (Class). The CollaborationUse must be owned by the Class to the contents of which it (the pattern) is applied. Properties of the applying Class can be bound to the roles of the Collaboration with Dependencies. The entities playing the roles must be owned by the same Class instance that owns the CollaborationUse. In short, with the UML pattern concepts, patterns are seen to describe contents of Classifiers.

Pattern literature of today, however, is not restricted to contents of UML Classifiers only. For example, many well-known patterns such as the Layers pattern [25] (and many other architectural patterns) are intended to clarify the division of systems to, e.g., Components or Packages. However, marking the occurrence of such patterns may not be possible with the UML concepts. This is because Packages are not Properties or necessarily owned by Classes. With application domain specific extensions, the support for patterns in UML becomes even more constraining. In order to benefit from the use of patterns in MDD, a new approach to define and mark patterns in models is required. The approach should restrict neither the types of elements that play roles in patterns nor the types of elements to contents of which patterns can be applied.

B. The New Pattern Approach

The developed pattern modelling approach [6] uses a set of concepts that have been developed for defining patterns and marking pattern instances in models. In the approach, pattern instances are not owned by Classes but Packages that are used in models in any case. The elements playing pattern specific roles in pattern instances can be any direct or indirect contents of the Packages and instances of any metaclass, instead of Properties only. Pattern definitions include textual properties that are essential information content in patterns. Lastly, the element roles in pattern definitions are separated from the template elements that are used in automating the application of patterns.

The approach is tool-supported including functions for instantiating patterns, exporting statistics and traceability information related to the use of patterns as well as for visualizing patterns in diagrams [6]. Patterns are instantiated to models with the use of a wizard that performs pattern specific modifications to the models, according to user selections. Markings of pattern instances are also created automatically by the wizard.

Statistics and traceability information on patterns can be exported to MS Excel files. Statistics include lists of design patterns that are used in a model including the number of instances for each pattern. Patterns are traced to Packages with traceability matrices to indicate the patterns that are used in each Package and vice versa. Visualizing patterns in diagrams utilizes the Collaboration notation of UML and presents pattern instances with dotted ellipses. Model elements that play pattern specific roles in the instances are connected to the ellipses with dotted lines. The tool support for the use of patterns can be used in any UML, Systems Modeling Language (SysML) or UML Automation Profile (AP) models and diagrams in UML AP research tool [26].

IV. SAFETY PATTERN METAMODEL

With extensions to safety aspects, the purpose has been to experiment how design patterns could specifically support documentation of safety applications. Most importantly, the extensions to the pattern modeling concepts, see Figure 1, include a specific SafetyPattern. SafetyPatterns are design patterns that have been identified to be related to safety. To distinguish the concepts that are used for defining patterns from those used to mark pattern instances, the Figure has been divided to two parts. The new (in comparison to [6]) concepts are in the Figure high-lighted with grey color.

A SafetyPattern is, thus, a design pattern that has been identified to be related to safety and that may have recommendations for applications of different *safety levels*. With safety systems, we refer to systems that perform *safety functions* the correct operation of which is required to ensure the safety of a controlled process. The safety levels in the metamodel correspond to the 4 Safety Integrity Levels (SILs) in IEC 61508 [5]. In general, a SIL determines the probability of correct functioning of a safety function, SIL1 being the lowest and SIL4 being the highest level. For traditional, e.g., electrical safety systems it is possible to determine SILs statistically. However, due to the systematic (vs. random) nature of software faults, the statistics approach cannot be applied to software. For new software components there would not even be statistics available. In IEC 61508, this problem is solved by focusing on development techniques and solutions the use of which are documented. For each SIL and for each development phase, the standard specifies a set of techniques that can be highly recommended (HR), recommended (R) or non-recommended (NR) or with non-specified recommendation (NS). The alternatives in the Recommendation (enumeration) in the metamodel correspond to these alternatives.

The purpose of the SafetyCatalogue concept is to collect together (from various pattern sources) related SafetyPatterns. Catalogues contain patterns that should be used together and to which sets of patterns that are used in models can be compared. Patterns in a catalogue can be related to, e.g., a phase in development or a specific purpose. For example, IEC 61508 [5] includes lists of techniques to be used during specific software development phases. For software architecture design, for instance, the standard mentions 27 techniques and/or measures, some of which are non-recommended or alternatives to each other.

 ISBN: 978-1-61208-367-4

Relations between Patterns can be modeled with the PatternRelation concept that has been extended with a Specialization relation. The background of the new (specialization) relation is an observation that many solutions (such as redundancy) that are recommended by safety standards actually have families of related, specialized pattern versions in pattern literature. With the Specialization relation, the purpose is to enable the use of general SafetyPatterns in SafetyCatalogues but in such a way that patterns specializing the general patterns can be considered as their alternatives.

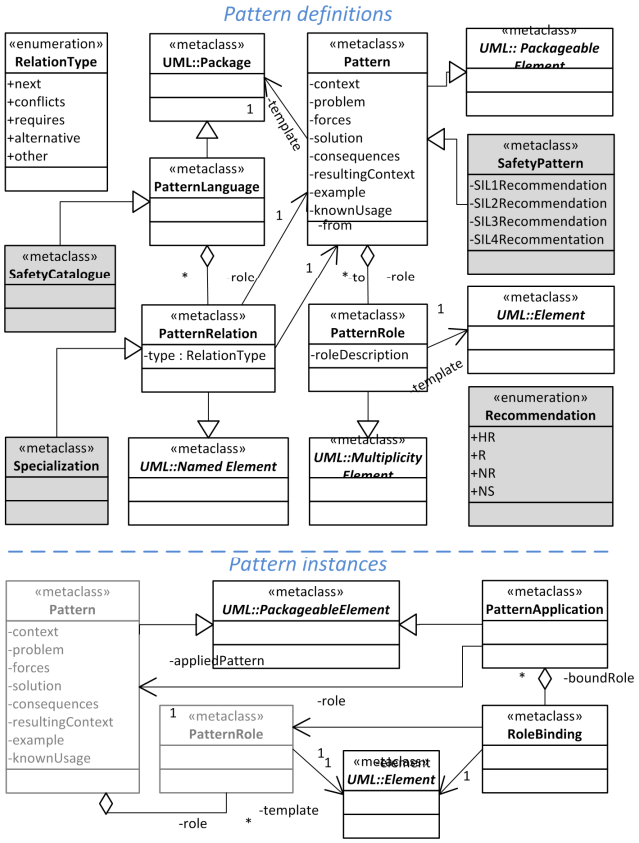

Figure 1. The new concepts for defining and using safety patterns.

The modeling concepts have been implemented to UML AP Tool [26]. With the implementation, the purpose has been to demonstrate how the concepts can be used to generate safety-related documentation. The implementation of the concepts uses Eclipse Modeling Framework (EMF) as a meta-modeling framework, with which the new concepts have been defined by extending the existing UML AP modeling concepts. The developed documentation generation extends the work presented in [6] and [27] that already addresses, e.g., traceability of requirements.

V. FOR GUIDANCE AND DOCUMENTATION

In this Section, we present three example documentation sheets. The generation of the sheets has been automated with use of the concepts. In addition to discussing how the sheets can be used, the following sub-Sections will briefly describe how the sheets are compiled from models.

The first of the sheets to be presented was created based on a SafetyCatalogue that had been defined to correspond to recommendations of IEC 61508 to software architecture design. The latter two example sheets compare a set of SafetyPatterns that is used in an example model to another SafetyCatalogue. The generation of the sheets relies on patterns that have been identified to be related to safety and that include recommendations for the different levels of safety.

A. *Safety Catalogue Sheet*

The purpose of the Safety Catalogue sheet is to enable illustrating SafetyCatalogues in a tabular form that is similar to the form of recommendation tables of IEC 61508 [5] (annex A of part 3 of the standard). On one hand, the sheet has been developed to facilitate the development of SafetyCatalogues, including checks of their conformance to standards. The tabular presentation can be used also during development to look for possible patterns or solutions that should be applied during specific design phases.

In addition to recommendations of safety standards, the sheet enables illustrating custom catalogues of SafetyPatterns for which there may not be standard recommendations. Nevertheless, such patterns may provide solutions to similar problems and be alternatives to each other. On the other hand, it may be meaningful to represent in which order such patterns should be applied so that composing pattern catalogues with next and alternative relations can be useful.

The Safety Catalogue sheet is compiled as follows. PatternApplications of an exported model are iterated through to find all SafetyPatterns that are used in the model. The SafetyPatterns are iterated through to find the SafetyCatalogues in which they appear. The list of the catalogues is provided to the user of the tool. The selected catalogues are printed to separate tables starting from their first patterns that are assigned number 1 in the tables. Next and alternative SafetyPatterns can be found with use of the PatternRelations. Alternatives are in the tables assigned same numbers but different letters, to indicate them being alternatives to each other. Recommendations of the SafetyPatterns to SILs are printed to the tables.

Safety Catalogue: "IEC 61508 Architecture Design"

#	Pattern	SIL 1	SIL 2	SIL 3	SIL 4
1	Fault detection	NS	R	HR	HR
2	Error detecting codes	R	R	R	HR
3a	Failure assertion programming	R	R	R	HR
3b	Diverse monitor techniques (with independence)	NS	R	R	NS
3c	Diverse monitor techniques (with separation)	NS	R	R	HR
3d	Diverse redundancy	NS	NS	NS	R
3e	Functionally diverse redundancy	NS	NS	R	HR
3f	Backward recovery	R	R	NS	NR
3g	Stateless software design	NS	NS	R	HR
4a	Re-try fault recovery mechanism	R	R	NS	NS
4b	Graceful degradation	R	R	HR	HR
5	Artificial intelligence - fault correction	NS	NR	NR	NR
6	Dynamic reconfiguration	NS	NR	NR	NR
7	Modular approach	HR	HR	HR	HR
8	Use of trusted/verified software elements (if available)	R	HR	HR	HR
	...				

Figure 2. An example generated Safety Catalogue sheet.

 ISBN: 978-1-61208-367-4

An example Safety Catalogue sheet can be seen in Figure 2 that presents a part of a printout of a catalogue of techniques or measures that IEC 61508 recommends for software architecture design. In the table, patterns can be highly recommended (HR), recommended (R) or non-recommended (NR) or with non-specified recommendation (NS). To avoid repeating a table of the standard, the table includes only 15 techniques that have been modeled as patterns. By looking at the table, however, it also becomes clear that pattern literature already includes specialized versions of many of the techniques, for example to implement redundancy [19].

B. *Safety Catalogue Conformability Sheet*

Whereas the purpose of the Safety Catalogue sheet is to enable presenting catalogues of SafetyPatterns, the purpose of Safety Catalogue Conformability sheets is to present how a set of SafetyPatterns (that are used in a model) conforms to a SafetyCatalogue. Similarly to the previous sheet, the conformability sheet serves both the guidance and documentation purposes. In addition, the table presents to which SILs the set of SafetyPatterns would be applicable.

The sheet is compiled as follows. In a similar manner than in the case of the previous sheet, the SafetyCatalogues related to the model are collected to a list from which the user may select the desired ones. General structure of the sheet is similar to the previous sheet. However, the SafetyPatterns of the catalogue that are used in the model are indicated with light grey color. In addition, the table presents whether the set of (used) patterns is compatible with each SIL. Compatibility of the used patterns is illustrated with green color and incompatibility with red color. Incompatibility can result from both using a non-recommended pattern or not-using a recommended (or highly recommended) technique or any of its alternatives.

The last two rows of the table also present the numbers of patterns (excluding alternatives) that would be recommended for each SIL and how many of them have been actually applied. As such, the table also answers the question how many techniques (more) should be applied in order to conform to the catalogue for each SIL.

Conformability to "IEC 61508 Software safety requirements specification"

#	Pattern	SIL 1	SIL 2	SIL 3	SIL 4
1a	Semi-formal methods	R	R	HR	HR
1b	Formal methods	NS	R	R	HR
2	Forward traceability between the system safety requirements and the software safety requirements	R	R	HR	HR
3	Backward traceability between the safety requirements and the perceived safety needs	R	R	HR	HR
4	Computer-aided specification tools to support appropriate techniques/measures above	R	R	HR	HR
	Pattern usage:	4/4	4/4	4/4	4/4
	Pattern usage (%):	100.0	100.0	100.0	100.0

Figure 3. An example generated Safety Catalogue Conformability sheet.

An example Safety Catalogue Conformability sheet can be found in Figure 3. It presents the conformability of SafetyPatterns used in an example model to the software safety requirement specification techniques of IEC 61508 [5] that have been modeled as a SafetyCatalogue. According to the table (grey highlighting), it can be seen that a semi-formal modeling technique has been used, the software safety requirements specification supports both backward and forward traceability and that computer-aided specification tools have been used. The table also illustrates (with green color) that these choices are applicable to all SILs. In addition to the techniques used, it is not necessary to use any other technique (for requirements specification).

C. *Safety Pattern Traceability Sheet*

While patterns can have recommendations for different levels of safety, it is also possible to check their conformance to safety levels required from the safety functions. The purpose of the safety pattern traceability sheet is to trace safety requirements (of UML AP) to Packages that contain implementing design elements for the requirements and to SafetyPatterns that are used in the Packages. In addition to traceability, the table presents the safety levels (SIL) related to the requirements, Packages as well as recommendations of the Patterns for each level. Similarly to the previous sheet, the use of recommended or highly recommended patterns is indicated with green color whereas the use of non-recommended patterns is warned with red color.

The sheet is compiled as follows. Safety-related (UML AP) requirements and their respective safety integrity levels are collected to a list. The Packages that contain implementing design elements for the requirements are identified based on TraceRelations (of UML AP). The SafetyPatterns, instances of which can be found from the Packages, are identified based on PatternApplications. The traceability table is printed. In the table, traceability between a requirement and a Package is presented with an arrow (↘). SILs for the Packages are determined by finding the highest SILs from the requirements that are traced to the Packages. Traceability between a Package and a SafetyPattern used in the Package is, again, presented with the arrow symbol.

Traceability: Requirements --> Packages --> (Safety) Patterns

REQUIREMENT:							
P100 Protection (SIL1)	↘						
P100IR (SIL1)		↘					
PACKAGE:	Software Safety Requirements (SIL1)	ControlStructures (SIL1)	PATTERN:	SIL 1	SIL 2	SIL 3	SIL 4
		↘	Automatic software generation	R	R	R	R
	↘		Semi-formal methods	R	R	HR	HR
	↘		Backward traceability between the safety requirements and the perceived safety needs	R	R	HR	HR
	↘		Computer-aided specification tools to support appropriate techniques/measures above	R	R	HR	HR
	↘		Forward traceability between the system safety requirements and the software safety requirements	R	R	HR	HR

Figure 4. An example generated Safety Pattern Traceability sheet.

An example Safety Pattern Traceability sheet can be found in Figure 4. According to the table, it can be seen that the example model contains 2 requirements of safety level SIL1: P100 protection and P100IR. The former one (a general safety function requirement) is traced to "Software Safety Requirements" Package and the latter one to "ControlStructures" Package. SILs required from the Packages (their contents) come from the requirements, both

 ISBN: 978-1-61208-367-4

being SIL1. Moreover, the sheet presents that the use of "Automatic software generation" has been marked in ControlStructures Package and Semi-formal methods, backward traceability, forward traceability as well as computer aided specification tool in the Software Safety Requirements Package. According to the table (color coding), the techniques are recommended for the safety integrity level (SIL1) required from the Packages.

VI. DISCUSSION

This paper has presented an approach to extend the information content of design pattern concepts of UML AP with safety aspects. The new concepts enable specifying the applicability of SafetyPatterns, i.e., design patterns of safety systems, to applications of different safety integrity levels. In addition, SafetyPatterns can be collected to SafetyCatalogues with which it is possible to model both recommendations of safety standards and custom catalogues of SafetyPatterns.

To illustrate the use of the concepts, the paper has presented 3 example documentation sheets. The sheets were generated automatically based on a library model containing two SafetyCatalogues and a model utilizing the patterns of the catalogues. The first of the sheets presented one of the catalogues. The other two sheets presented compliance of a model (of a developed systems) to the other catalogue. The new information content of SafetyPatterns was in the sheets used for automating identification of safety-related patterns and consistency checks with respect to safety levels. The sheets, thus, documented rather the developed systems than SafetyPatterns themselves. In the developed metamodel, SafetyPatterns share most of their information content with the design pattern modeling concepts that are used in [6].

The authors believe that the possibility to export documentation from models is a future research topic within MDD research. Moreover, it could improve the applicability of the MDD techniques to safety system development. This is because safety applications cannot be used in practice without appropriate documentation. Without automated support for producing documentation, it would have to be produced manually. On the other hand, by automating even part of the work, it would be possible to obtain additional, MDD specific benefits in the application area.

When developing safety applications with MDD techniques, the development process should be supported. A tool should assist developers by pointing out the issues that need to be addressed, by presenting the alternatives (when appropriate) and by documenting the decisions for later use. For example, the supported process could start from modeled requirements that determine the required integrity levels. A developer could select a SafetyCatalogue to be used to guide, e.g., architecture design. Based on the selection and required integrity levels, the tool could suggest patterns to be used. In practice, this scenario could be supported with only a small modification to the Safety Catalogue sheet, by hiding inappropriate patterns based on required integrity levels.

Work that aims for guiding development work in MDD has been previously carried out by the authors also based on use of an Architecture Knowledge Management (AKM) platform [28]. Use of an external tool, however, may lead to redundant information. On the other hand, it is believed that documentation and guidance support should be available for both architectural and detailed design levels. Thus, it is feasible to integrate the required support in one tool, which is used throughout the MDD process.

A challenge in developing guidance for MDD is that development processes, techniques and solutions vary between companies and between controlled processes. The approach presented in this paper could improve the situation. Documentation sheets can be developed to support various purposes and processes, not only the ones presented in this article. In addition, by using, e.g., the SafetyCatalogue concept, the generated sheets and their contents are also dependent of the catalogues to the contents of which the models are compared. Thus, to support another kind of a development process or other techniques, one could specify other catalogues to which the models would be compared.

The authors regard safety aspects important for also basic control systems that are not critical. Safety is an issue that should be taken into account in development of any control system. Safety standards state their recommendations on techniques, measures and solutions based on evidence on their usefulness. It is likely that adopting selected techniques and measures from safety system development, e.g., traceability could also improve the quality of basic control systems. This could in turn improve the productivity of the controlled processes at least in application domains in which the development processes are not strictly governed.

On the other hand, considering selected aspects of safety standards in development of basic control systems could shorten the gap between the systems. Safety systems and basic control systems are currently not only separated from each other but also developed with different development processes and tools and often by different teams. It is possible that professionals are not even aware of the practices in the other teams. Because the development of safety systems is regulated by authorities, the only possibility to shorten the gap would be to adopt suitable practices of safety system development to basic control system development.

VII. CONCLUSIONS

Design patterns document solutions and capture expert knowledge to recurring challenges in design and development work. On one hand, design patterns support the re-use of design by preserving named, proven solutions to recurring challenges. However, they can also increase the documentation value of models that usually tend to present design solutions rather than rationale behind the solutions. With use of patterns, designs become easier to understand and the roles of design elements clear for possible third parties that use the documentation. Especially the use of patterns could benefit MDD in which the idea is to use models for both development and documentation purposes.

In this paper, a set of pattern modeling concepts was presented that enable increasing the information content of design patterns with applicability to safety integrity levels. The new concepts enable constructing catalogues of safety-related patterns with which it is possible to model

 ISBN: 978-1-61208-367-4

recommendations of safety standards. Automated functions for generating documentation sheets enable the use of the concepts for producing documentation. In addition to presenting which patterns are used in a model, the sheets present whether the models comply with the catalogues, e.g., recommendations of safety standards. The sheets can be used also during development as guidance to present the standard-compliant selections that still have to be addressed.

Ability to use models as documentation or to produce documentation from models to a suitable form is a possible key for industrial acceptance of MDD techniques in safety system development. Without automated support, the documentation would have to be produced manually. This could significantly reduce the potential to benefit from MDD. However, with documentation support, MDD would provide another means to benefit from the use of models.

When developing safety applications with MDD techniques, the development process should be supported and guided in a flexible manner. Instead of only predefined forms and checks, the presented documentation tables are compiled with use of modelled SafetyCatalogues to which models are compared. As such, the suggestions that the tool can be considered to provide are also dependent on the modelled catalogues. Tailoring the approach for different application domains or development practices could thus be possible to achieve with changes to the catalogues. While acknowledging that the development concepts still require further development, the authors regard this kind of flexibility as an important feature in MDD tool support.

REFERENCES

[1] C. Alexander, S. Ishikawa, and M. Silverstein, "A pattern language: towns, buildings, construction", Oxford University Press, 1977.

[2] C. Alexander, "The timeless way of building", Oxford University Press, 1979.

[3] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design patterns: Elements of reusable object-oriented software", Addison-Wesley, 1994.

[4] OMG, "Model Driven Architecture (MDA) Guide", Object Management Group, 2003.

[5] IEC, "61508 functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements", International Electrotechnical Commission, 2010.

[6] T. Vepsäläinen and S. Kuikka, "Design pattern support for model-driven development", in 9th International Conference on Software Engineering and Applications, 2014. (in press)

[7] OMG, "Unified Modeling Language Specification 2.4.1: SuperStructure", Object Management Group, 2011.

[8] G. Sunyé, A. Le Guennec, and J. Jézéquel, "Design patterns application in UML", in Proc. of 14th European Conference on Object-Oriented Programming, 2000, pp. 44-62.

[9] No Magic, Inc. MagicDraw, 2014. Available: http://www.nomagic.com/products/magicdraw.html [retrieved: 07, 2014]

[10] R. B. France, D. Kim, S. Ghosh, and E. Song, "A UML-based pattern specification technique", IEEE Transactions On Software Engineering, vol. 30, pp. 193-206, 2004.

[11] J. Dong, Y. Sheng, and K. Zhang, "A model transformation approach for design pattern evolutions", in Proc. of 13th Annual IEEE International Symposium and Workshop On Engineering of Computer Based Systems, March 2006, pp. 80-92.

[12] P. Kajsa and L. Majtás, "Design patterns instantiation based on semantics and model transformations", in SOFSEM 2010: Theory and Practice of Computer Science, Springer, 2010, pp. 540-551.

[13] W. Xue-Bin, W. Quan-Yuan, W. Huai-Min, and S. Dian-Xi, "Research and implementation of design pattern-oriented model transformation", in 2nd International Multi-Conference on Computing in the Global Information Technology, 2007.

[14] J. Dong and S. Yang, "QVT based model transformation for design pattern evolutions", in Proc. of 10th IASTED International Conference on Internet and Multimedia Systems and Applications, 2006, pp 16-22.

[15] S. Wenzel and U. Kelter, "Model-driven design pattern detection using difference calculation", in Proc. of 1st International Workshop on Pattern Detection for Reverse Engineering, October 2006.

[16] M. L. Bernardi, M. Cimitile, and G. A. Di Lucca, "A model-driven graph-matching approach for design pattern detection", in Proc. of 20th IEEE Working Conference on Reverse Engineering, 2013, pp. 172-181.

[17] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, and S. T. Halkidis, "Design pattern detection using similarity scoring", IEEE Transactions On Software Engineering, vol. 32, pp. 896-909, 2006.

[18] A. Pande, M. Gupta, and A. K. Tripathi, "A new approach for detecting design patterns by graph decomposition and graph isomorphism", in Proc. of 3rd International Conference on Contemporary Computing, Springer, 2010, pp. 108-119.

[19] B. P. Douglass, Real-Time UML: Developing Efficient Objects for Embedded Systems. Addison-Wesley, 1998.

[20] R. Hanmer, Patterns for Fault Tolerant Software. John Wiley & Sons, 2013.

[21] T. Saridakis, "Design patterns for checkpoint-based rollback recovery," in Proc. of 10th Conference on Pattern Languages of Programs (PLoP), Spetember 2003.

[22] T. P. Kelly and J. A. McDermid, "Safety case construction and reuse using patterns. in Proc. of 16th International Conference on Computer Safety and Reliability, Springer, 1997, pp. 55-69.

[23] W. Herzner et al., "Model-based development of distributed embedded real-time systems with the decos tool-chain," in Proc. of 2007 SAE AeroTech Congress & Exhibition, 2007.

[24] G. Zoughbi, L. Briand, and Y. Labiche, "Modeling safety and airworthiness (RTCA DO-178B) information: conceptual model and UML profile", Software & Systems Modeling, vol. 10, pp. 337-367, 2011.

[25] F. Buschmann, R. Meunier, H. Rohnert, P Sommerlad, and M. Stal, "Pattern Oriented Software Architecture: A System of Patterns". John Wiley & Sons, 1996.

[26] T. Vepsäläinen, D. Hästbacka, and S. Kuikka, "Tool support for the UML automation profile - for domain-specific software development in manufacturing", in Proc. of 3rd International Conference on Software Engineering Advances, October 2008, pp. 43-50.

[27] T. Vepsäläinen and S. Kuikka, "Towards model-based development of safety-related control applications", in the 16th IEEE International Conceference on Emerging Technologies & Factory Automation, September 2011.

[28] T. Vepsäläinen, S. Kuikka, and V. Eloranta, "Software architecture knowledge management for safety systems", in the 17th IEEE International Conceference on Emerging Technologies & Factory Automation, September 2012.

 ISBN: 978-1-61208-367-4