A Model Driven Approach for Requirements Engineering of Industrial Automation Systems

Hongchao Ji[1] Oliver Lenord[1] Dieter Schramm[2]

[1]Bosch Rexroth AG, Germany
{hongchao.ji, oliver.lenord}@boschrexroth.de
[2]Institute for Mechatronics and System Dynamics , University of Duisburg-Essen, Germany
dieter.schramm@uni-due.de

Abstract

Model driven requirements engineering (MDRE) is proposed to deal with the ever-increasing complexity of technical systems in the sense of providing requirement specifications as formal models that are correct, complete, consistent, unambiguous and easy to read and easy to maintain. A critical issue in this area is the lack of a universal and standardized modeling language which covers the whole requirements engineering process from requirement specification, allocation to verification. SysML is being proposed to meet these requirements. In this paper a model driven requirements engineering process for industrial applications in the field of automation systems is described in order to reveal shortcomings in recent modeling tools and modeling languages. Special focus is layed on the requirements definition and the automated verification of the design against the requirements using executable models. Based on the analysis a new profile of the Unified Modeling Language (UML) called Model Driven Requirements Engineering for Bosch Rexroth (MDRE4BR) is presented which aims to contribute to latest investigations in this field. An application example of a hydrostatic press system is given to illustrate the approach.

Keywords Model Driven Requirements Engineering, Industrial Automation Systems, SysML, Modelica

1. Introduction

Model driven engineering (MDE) has been proven to be capable to cope with complexity in the field of software engineering. A trend in modeling and simulation is to apply the MDE paradigms to technical systems consisting of components in hardware and software. The increasing complexity of technical systems has raised many challenges such as keeping the the design consistent and approving the correctness with respect to the customer requirements. Studies at the Bosch Group have shown that over 50% of field problems were due to insufficient requirements engineering (RE) [6]. The RE accompanies the whole product development process, in which various engineering disciplines are involved. Therefore, a universal and standardized modeling language is required which shares the understanding among engineers from different disciplines. This common language shall enable the building of requirements models, system design models, traceability models as well as verification models containing domain-specific details.

The Systems Modeling Language (SysML) [15] is being proposed by the Object Management Group (OMG) [8] to meet these requirements and has already been evaluated by several researchers [4, 5]. The main drawbacks coincide with the results of the analysis of the engineering process of automation systems in section 2 which can be concluded as follows: first the requirements constructs in SysML are not sufficient for real industrial applications, and second the SysML is not capable to describe continuous-time dynamic models.

In order to contribute to the solution of these shortcomings, a UML profile MDRE4BR is proposed in this paper. It reuses and extends the current requirements constructs in SysML targeting the first issue. Recent works on the integration of SysML and Modelica [7] like ModelicaML [12] have already addressed the second issue and proved its effectiveness [13]. Reusing these improvements the proposed extensions of the MDRE4BR are linked with the ModelicaML to transform the later introduced analytical models into executable Modelica models.

This paper is organized in 6 sections. Section 2 motivates the use of the model driven requirements engineering approach in the field of industrial automation systems. Requirements deriving from the application of this approach to the systems engineering of automation systems are discussed. Section 3 gives a short introduction to the related work on the modeling languages SysML, Modelica and ModelicaML. Section 4 describes the implementation of the MDRE4BR profile in detail. The capabilities of the proposed methodology and the MDRE4BR profile are demonstrated on behalf of an industrial application in section

4th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools. September, 2011, ETH Zürich, Switzerland.
 Proceedings available at:
http://www.ep.liu.se/ecp/056/
EOOLT 2011 website:
http://www.eoolt.org/2011/

5.The paper is closed with conclusions and an outlook to future work.

2. Requirements Engineering in the Field of Industrial Automation Systems

2.1 Scope

This work is seen from the systems engineer's perspective in the field of industrial automation. In terms of building a solid foundation of the later derived requirements on the modeling languages and the tool support, the engineering process is described at first.

Industrial automation systems are characterized by their ability to process a material or work piece according to a defined procedure to achieve the output of a desired product. The quality of the product shall remain stable even though the boundary conditions as climate, disturbances and material properties may differ in a given range. The process is managed in an automated way in order to meet the quality goals in a reproducible, efficient and reliable way.

The systems engineer is now challenged to design a machine that is capable to run the process in a deterministic way. This task is typically performed within a specific design domain that refers to a field of technical expertise such as: the treatment technology, the mechanical design, the drive system and the control. The treatment is usually considered as the core competence of the original equipment manufacturer (OEM). The same is in general true for the closely related mechanical design.

The drive and control technology is typically provided by suppliers. This is due to the fact that power supply, actuators and controls are available on the market as cost efficient standard components in a high quality. Nevertheless the drive and control system is also strongly coupled to other parts of the automation systems. The proper selection of the components and their integration into the overall structure strongly influence function, performance, robustness and reliability. Leading edge technology is determined by the competence of interdisciplinary system design.

Due to the fact that the requirements specification is the subject matter of contract between customer and contractor the above described context implies that the requirements engineering has to be seen not only from the technical perspective but also needs to consider the contractual situation along the supplier chain. With respect to that it is self-evident that the requirements shall be defined and structured not only according to technical aspects but also according to the contractual situation. The definition of levels of abstraction is an appropriate way to meet these needs. The depicted levels of abstraction in Figure 1 reflect the described supplier chain and major technologies involved and therefore are a reasonable choice on behalf of the automated press system considered in section 5.

In order to deal with the complexity of large systems the design objects are clustered in a system break down structure. The requirements derived on the different levels of abstraction can be referenced in requirement specifications in order to provide the contractual views on subsets of requirements.

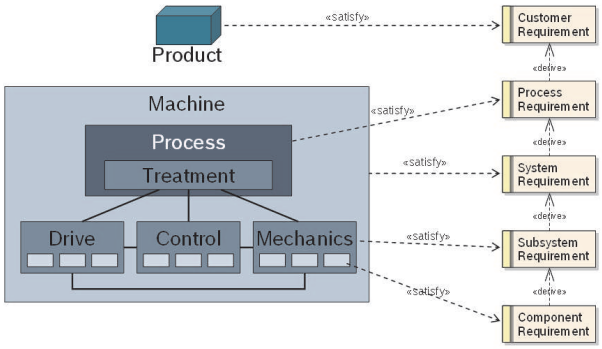

Figure 1. Levels of Abstraction in Requirements and System Design

2.2 The Model Driven Requirements Engineering Approach

Dealing with the above mentioned challenges of managing requirements the MDRE is a very promising and well supported approach. In addition to the UML the SysML defines requirements and several relations between requirements and between requirements and design objects such as: derive, refine and copy. Together with an appropriate package structure the proposed levels of abstraction can be reflected in a SysML requirements model. A classification of requirements can be easily established through stereotypes.

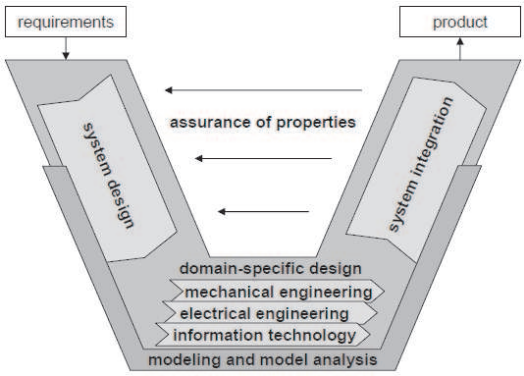

Figure 2. V Model According to VDI 2206 [16]

Furthermore the SysML supports the system design through structural and behavioral diagrams. Relations like trace and satisfy support the traceability between design objects and requirements.

All of this covers a wide range of what is needed in order to refine abstract customer requirements towards a detailed design. The descriptive character of the models are a very appropriate representation of the system design since it allows a certain fuzziness that is unavoidable in the early stage of the system design. Referring to the well known V Model according to the VDI 2206 standard [16] as depicted in Figure 2, it is concluded that the system design phase is well supported by the SysML.

In the domain-specific design, the vertical lower part of the V Model, real world examples, i.e. the hydraulic schematic of a press system, show that the expressiveness of the SysML and its descriptive models are too weak to precisely describe a design on a physical level. Standardized domain-specific schematics, like i.e. the ISO1219, have a much stronger semantics. In practice they have been proven to precisely describe a design such that there is no doubt on how to actually manufacture, assemble and commission the specified pieces. The challenge at this point is to close the gap between the descriptive design model and the physical design model. The SysML concept of blocks and ports is reflected by object-oriented drawing tools for hydraulic and electric circuits like D&C Scheme Editor [1] and ePLAN [9]. The SysML supports domain-specific graphical representations of blocks. The block attributes can be used to define specific technical attributes. Still it is very inconvenient to use common UML/SysML tools for the purpose of drawing domain-specific schematics. Mainly this is due to the fact that no libraries of standard components are available including a large number of symbols and related attributes for all kinds of components and variants. Furthermore common drawing features like title blocks or generating parts lists are not supported. A future engineering tool aiming to support the whole engineering process will have to cope with these requirements.

In the iterative process of refining the descriptive model it is desirable to frequently check the design. According to the V Model (Figure 2) this refers to the depicted iterative step of ”assurance of properties”. Subject matter of this task is to verify the integrated system model against the requirements. Up to now this kind of system verification and validation is a manual procedure that relies on the expertise of the systems engineer. In the sense of a model driven development process it would be beneficial to run this procedure in an automated fashion. Not for all types of requirements this is applicable. In case of analytically verifiable requirements (introduced in 4.2) this could be performed based on a system simulation. To achieve this goal, modeling languages and modeling tools are challenged to answer the following questions:

1. How to achieve an executable analytical model that is capable to approve functional requirements?
2. How to link the analytical model with the design objects of the descriptive model in order to keep it consistent with the evolving system design?
3. How to establish a link to the requirements such that the execution of the analytical model reveals directly which requirements are violated or satisfied?

Because of the interdisciplinary character of automation systems, domain-specific simulation tools like SIMULINK, ADAMS or PSPICE are applicable only in case of requirements that can be evaluated in the related domains separately. In general an integrated interdisciplinary simulation model is required. In the field of multi-physics simulations several modeling methods and tools are available to provide a satisfying integrated system simulation that applies to question number one. A well established and standardized multi-physics modeling language is Modelica which is considered in the following for further investigation.

The other two questions have been addressed by several work on modeling languages that are discussed in the following section.

3. Background and Related Work

3.1 Introduction to SysML and Modelica

SysML is a general purpose language used in the field of systems engineering. It is defined as a UML profile which reuses subsets of UML constructs and extends them with some additional modeling elements. The SysML is capable to capture the textual requirements and to allocate them with the design models and test cases. However due to loosely defined executable semantics SysML is not capable to model physical systems in an executable way. In contrast to that, Modelica is an object oriented and equation based modeling language for multi-domain physical systems. Graphical modeling is supported by the object diagram which offers an intuitive way to describe power transmission through acausal connections as well as directed signal flows. The strong semantics allow the generation of executable models of continuous as well as discrete systems. Object oriented language constructs enable the efficient reuse of models and the design of comprehensive and easy to use model libraries. A language which integrates the descriptive modeling power of SysML and the formal executable simulation power of Modelica seems to be a promising approach for the requirements engineering in industrial automation systems. An overview on latest research activities in this field is given in the following.

3.2 Requirements Specification

Several researchers have already used and extended the SysML as requirement specification language. In Dubois et. al. [2] a requirement meta model to enforce the traceability concept in SysML in the automotive domain is presented, in which a traceability model connects three independent flows (requirement model, solution model and verification and validation (V&V) model). However the traceability of the V& V model to the other models is not clearly defined.

The vVDR methodology [14] addresses the virtual verification of systems requirements. The other contribution of this work is an approach to formalize requirements such that they can be quantified and tested effectively. The vVDR approach is considered in the later case study.

3.3 Integration of SysML and Modelica

Main target of the ModelicaML language by Schamai et. al. [12, 13] is to provide an executable graphical language for hybrid modeling and simulation while enabling an effective way to create, read and maintain Modelica models. In contrast to the original ModelicaML [11], the new ModelicaML is implemented as a pure UML profile with no direct dependency on the SysML. Instead, the new ModelicaML uses a subset of UML, extends the UML meta model (us-

ing the UML profiling mechanism) with new constructs in order to introduce missing Modelica concepts and to reuse several SysML concepts like the requirement constructs. ModelicaML is defined as a graphical notation that facilitates different views (e.g. composition, inheritance and behavior) on Modelica models. Those graphical notations can be translated into Modelica code and simulated with any Modelica tool.

The SysML-Modelica Transformation Specification is another research activity on the integration of SysML and Modelica. The main target of this work is to specify a standardized bi-directional transformation as foundation for later implementations that support the unambiguous, efficient and automatic transfer of model information between SysML and Modelica. In order to define a formal transformation between SysML and Modelica, an extension to SysML called SysML4Modelica profile, that represents most common Modelica constructs, is developed. In this profile, each Modelica construct will be checked whether there is already an equivalent construct in SysML from a semantic point of view. When the corresponding construct does not exist, a stereotype will be created to extend the SysML language accordingly. A mapping between the SysML4Modelica profile and the Modelica meta model is specified which enables a round-trip transformation from SysML to Modelica and vice versa [10].

The SysML-Modelica Transformation addresses the need of SysML users to define analytical relations in a mathematical form in addition to the existing descriptive constructs. The SysML4Modelica profile is simply applied to a selected sub part of the system model which is constraint through mathematical relations to be analytically resolved. This modeling and simulation approach is pretty straight forward in case of rather simple relations and measures between parameters, referred to as parametrics. In case of advanced simulation models this approach loses its practical relevance. The expressiveness of the Modelica object diagram in conjunction with the large number of easily reusable models in Modelica libraries is indispensable for even small system models.

The general approach of separating the descriptive model from the analytical model is useful. The design process is an incremental procedure which implies that not all parts are defined on the same level of detail at a time. Furthermore it is common that different parts of the model shall be simulated under different circumstances. This aspect is not addressed in the ModelicaML profile which considers the design and the simulation model as one consistent instance of the system.

4. The MDRE4BR Profile

The MDRE4BR profile is structured in three parts, namely the requirements definition package which classifies different types of requirements; the requirements traceability package, which details different traceability links related to requirements allocation; the requirements verification package, which covers the verification of the design against the requirements by means of an executable model. These packages are defined in detail right after the description of the underlying concept.

4.1 Overall Concept

The *requirements model* contains all requirements according to the classification below.

The *design model* is a descriptive model to describe structure and behavior of the system. Every design object is uniquely defined in order to describe a complete and consistent model of the system.

The *analytical model* is an executable model. This requires a much more formal description of the behavior. This is performed through the mathematical expressiveness of Modelica. The analytical model consists of one or multiple mathematical models. Each mathematical model describes one aspect of the overall context. In contrast to the design model this may lead to the case that the very same component is represented through different mathematical models. Referring to the press system in section 5 this would refer to the case that i.e. the pressure supply is considered under two different aspects. First in the analysis of the stability of the control circuit, second to approve the proper layout of the suction pipe of the hydraulic aggregate. In the first case a simplified representation as ideal constant pressure source is adequate, while in the other case the dimensions of pump, suction pipe and tank have to be considered in the mathematical description.

If the simulation models are decoupled from each other redundant data is produced. Sooner or later this will lead to inconsistencies that are hard to detect and have a strong impact on the quality of the design. To prevent this situation a relation between the mathematical models and the design objects is required. In order to realize meaningful relations the design object has to reflect the attributes of all related models. In the profile this is easily considered by referencing the attributes of the design objects from the Modelica parameters and constants of the analytical model. In the case study these relations have been established manually which is time consuming and error prone. At this point an improved tooling is needed that supports the binding of related objects. A promising approach would be to provide a component library of predefined objects that contain the relevant attributes of the design object as well as references to a number of meaningful mathematical models of different level of detail.

The mathematical model can be transformed into an executable model. This transformation is performed through compilation of the Modelica model and binding with the simulation run-time. The simulation run-time instantiates and runs the algorithm that is needed to solve the differential-algebraic equations. This procedure requires a definition of the simulation settings such as start time, end time, tolerances and output step size. Furthermore the initial state of the equation system needs to be defined. In the MDRE4BR profile these informations are described through the so-called test scenario. The test scenario defines all these boundary conditions under which a mathematical model shall be executed.

In order to use an executable model for the purpose of requirements verification the test scenario can be extended with so-called test cases. The test case is a mathematical model that evaluates a specific measure by comparing the simulation results with a desired output. Further details are given in section 4.2.

4.2 Requirements Definition

A general definition of the MDRE4BR requirement (see Figure 3) extends the standard requirement definition in SysML with additional attributes which are described as follows:

- The *id* property is the unique identifier of the requirement (as defined in SysML);
- The *text* property is the textual description of the requirement (as defined in SysML);
- The *abstrationLevel* property defines the different abstractions level of requirements;
- The *priority* property defines the importance of the requirement such as mandatory or optional;
- The *version* property records the change history of the requirement;
- The *satisfyState* identifies whether one requirement has been fulfilled by a model object or not;
- The *verifyState* property identifies the verification state of the requirement which can be pending, passed, failed or not verifiable according to the verification results in the test cases.

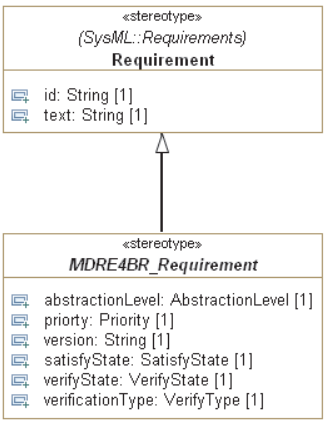

Figure 3. General Requirement Definition

The requirements classification is another additional concept in the requirements definition. The requirements can be classified as analytically-verifiable and not-analytically-verifiable requirements. Analytically verifiable refers to requirements which can be verified through evaluation of the system model against formally described criteria. In the context of this paper this is limited to the case that the behavior of the system is described by an executable simulation model which is verified through test scenarios and test cases. In contrast to that not-analytically-verifiable requirements refer to requirements which require additional knowledge or judgment to decide whether or not the design satisfies the requirement. In this case all related aspects of the design need to be easily accessible to the decision making system engineer. For this purpose the trace or satisfy relation between requirement and design object shall be used.

The following classification, based on the taxonomy proposed in [3], is supported by the MDRE4BR profile through stereotypes.

- A *functional requirement* is a requirement that should produce an expected reaction to a given stimuli.
- A *performance* is a requirement to check whether a system variable such as timing, speed, volume or throughput is in a desired range.
- A *structural requirement* is a requirement which describes the structural demand of the stakeholder.
- All the other types of not-analytically-verifiable requirements can be modeled with the *other requirement* stereotype.

This classification allows distinguishing requirements according to how they shall be processed in the verification phase. Requirements that shall be verified automatically are identified and described through additional attributes such that they can be processed in the desired fashion.

4.3 Requirements Traceability

Requirements traceability is used to specify the relationships from and to requirements. At first this defines the relations needed to express an appropriate requirements break down structure considering different levels of abstraction and other dependencies among the requirements. Furthermore relations are defined that are needed to trace the requirements from and to the objects of the design model and/or the analytical model.

The current available traceability links in SysML are defined as follows:

- The copy, containment and deriveReqt are defined to model the traceability among requirements;
- Traceability between requirements and design objects are supported by satisfy, trace and refine;
- The relationship between requirements and test cases are defined with verify.

These traceability links defined in the SysML cover the relationships needed in the considered industrial field of application. The traceability between requirements and design objects supports the systems engineer to systematically complete the system model. This is done by incrementally building up the design model through additional design objects that are needed to satisfy a particular set of requirements. Finally all design objects have been checked against all requirements of the related level of abstraction such that all applicable satisfy relations have been established. In this process it may apply that a design object can not be sufficiently specified through the existing requirements. This revealed incompleteness of the requirements is resolved by the iterative refinement of the requirements.

In the described context of the design phase the traceability links are very useful but the application for the desired automated verification is limited due to the lack of formal executable semantics and syntax. The proposed extended traceability link explicitly defines the information to be transferred via this traceability links in order to achieve an automated verification. This is done by allocating a performance requirement with a performance test case, using the MDRE4BRverify relation. The performance variable defined in the performance requirement is associated with the same in the test case as depicted in Figure 4.

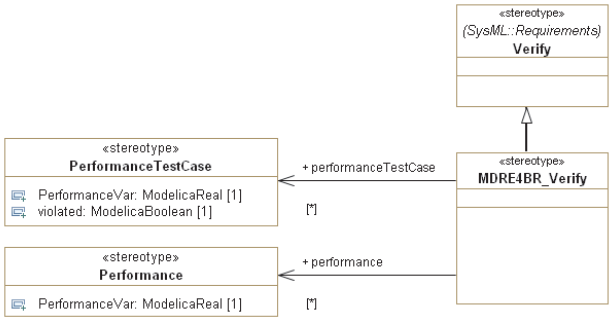

Figure 4. Verify Relation in MDRE4BR Profile

4.4 Requirements Verification

The goal of the requirements verification in the MDRE process is to verify the design against the requirements in an automated and reproduceable way. In the MDRE4BR profile this is achieved by combining the above described extended semantics of the verify relation with the concept of violation monitors and variable binding described in the vVDR methodology [14].

In the MDRE4BR profile, different types of test cases for different kinds of requirements are defined as stereotypes, such as, the performance test case and the functional requirement test case which are derived from the SysML meta class TestCase as shown in Figure 5. The violation monitor is modeled as part of the performance test case or functional test case in order to evaluate the requirement.

A new stereotype called test scenario is defined, which contains multiple test cases referring to at least one mathematical model. The requirements verification can be performed by executing the test scenarios and related test cases defined in the analytical model. The relations among these elements are depicted in Figure 6.

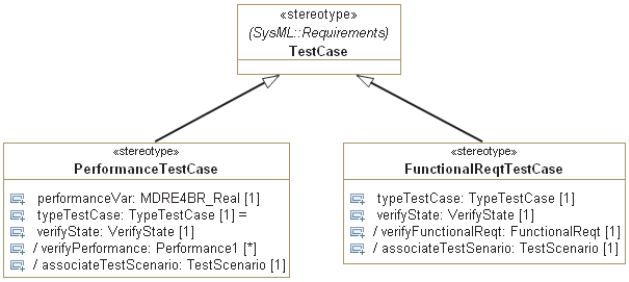

Figure 5. The Performance and Functional Test Cases

In the verification process, the user defines the different test scenarios by referencing the selected test cases and the mathematical models needed to produce the simulation results. The variable binding is used in the test scenario to establish the links between test cases and mathematical model as describe in the vVDR method [14] .

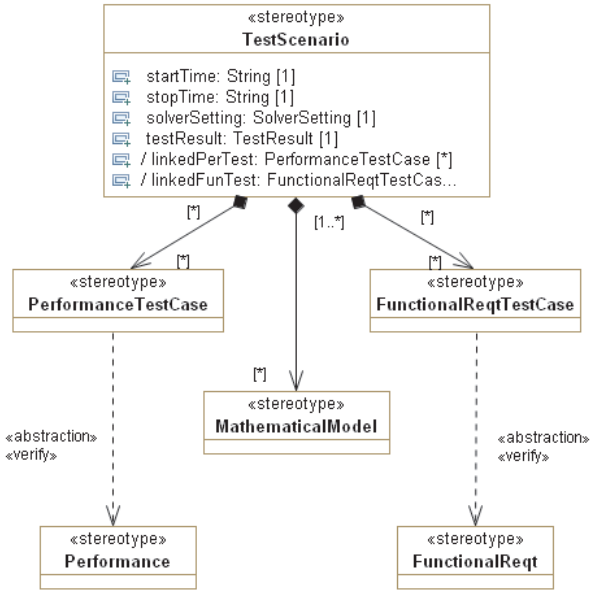

Figure 6. The Relations in Verification Package

5. Case Study: Hydrostatic Press System

In this section, a hydrostatic press system (Figure 7)is used to illustrate the model driven requirements engineering approach by using the MDRE4BR profile. The main steps of the procedure can be summarized as follows:

Figure 7. Hydrostatic Press

1. Capture the customer requirements as stereotyped requirements according to the proposed classification.
2. Derive requirements on a lower level of abstraction using the «deriveReq» relation to the higher level requirements.
3. Create descriptive structural and behavioral design models.

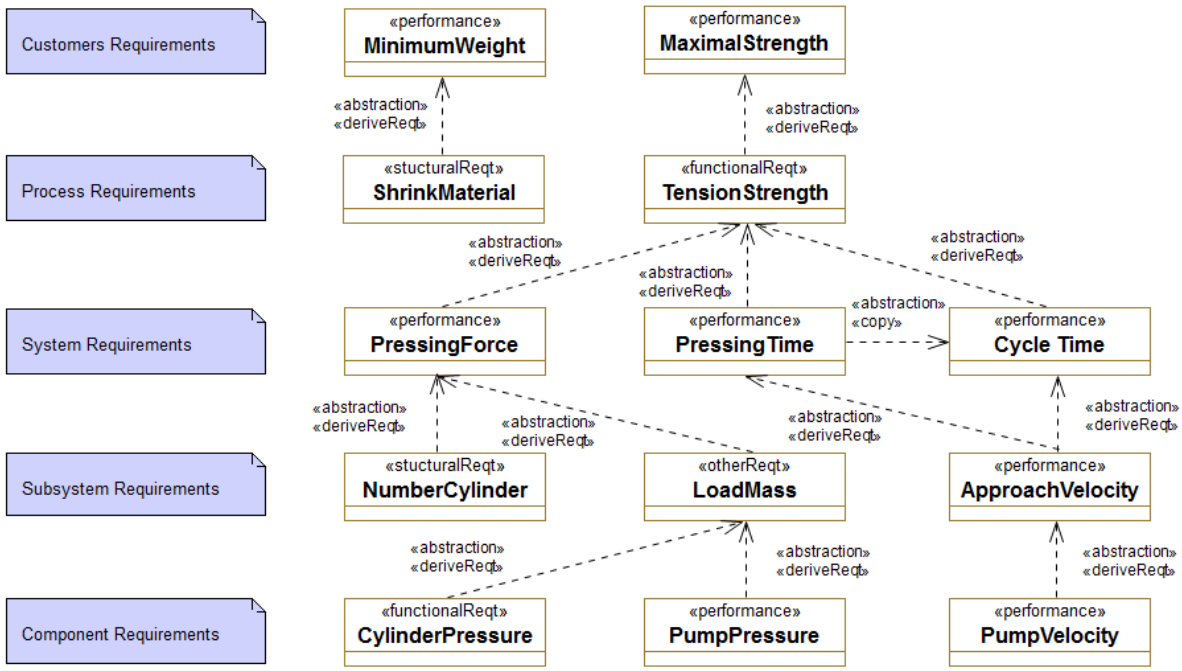

Figure 8. Derive Requirements from Different Abstraction Levels

4. Establish «satisfy» relations between requirements and design objects.
5. Define test cases for those design objects that need to satisfy verifiable requirements.
6. Establish «verify» relations in order to bind the result of the test case to the verifyState of the related requirement and to retrieve the measures from the requirement and apply them to the related test case variable.
7. Create a mathematical model of the related design objects on the considered level of abstraction.
8. Define test scenarios which execute the mathematical models under defined boundary conditions.
9. Integrate those test cases in the test scenarios which are applicable.
10. Run a model verification that executes all related test scenarios and updates the verifyStatus of all related verifiable requirements.

This procedure is performed in an incremental fashion until the design has reached a satisfying level of detail that is also reflected in the analytical models. After the final verification all verifiable requirements shall be approved through the related test scenarios.

In the case study a press system is considered which shall be used to demonstrate the procedure on behalf of a real industrial application.

5.1 Requirements Capture and Refinement

The hydrostatic press shall be considered in the context of the OEM-supplier relation as it applies to a typical Bosch Rexroth engineering project. In this case the high level requirements have already been refined to the subsystem level. Figure 8 shows some exemplary requirements reflecting different levels of abstraction.

Because of the limited space in the diagram, the descriptions of the requirements are to be found in Table 1 and 2. In the following the focus is on the the requirements from the abstraction level system requirements and below. As shown in Figure 8, these system requirements are derived from the customers and process requirements and can be classified according to the subtypes of the requirements definition in the MDRE4BR profile. The beginning letter of the ID of the requirement refers to the type of the requirement.

Name	Description
MinimumWeight	Steal structure shall have minimum weight.
MaximalStrength	Steal structure shall have maximum strength.
ShrinkMaterial	In-mold hardening shall not shrink the material below manufacturing tolerance of 0.1mm.
TensionStrength	In-mold hardening shall increase the tensile strength to $1000N/mm^2$

Table 1. The Customers and Process Requirements

5.2 Structural and Behavioral Design

In Figure 9 the overall design on the subsystem level is depicted using the SysML internal block diagram. This rough design refers to an early phase of the design process in which the subsystems and the interfaces are in the main focus.

The behavior of the system is described through the SysML activity diagram in Figure 10 according to the working process shown in Table 3. The conditional state-

ID	Description
P1	The pressing force of cylinder shall be $180000kN$.
P2	The pressing time shall shall be $8s$.
P3	The cycle time shall not exceed $27s$.
P4	The approach velocity of the load shall be $0.5m/s$.
P5	The pressure drop across the pump shall not exceed 340 bar.
P6	The driving speed of the pump is 3000 rpm in approach phase.
F1	The cylinder pressure can not exceed 120 bar.
S1	The number of driving cylinders is 2.
O1	The moving mass shall be 40t.

Table 2. The System Requirements List

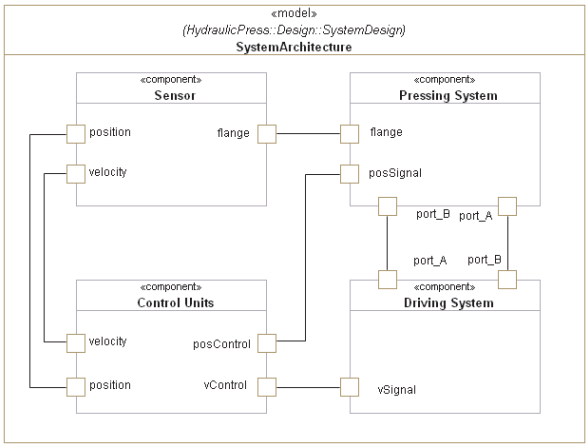

Figure 9. The System Architecture

ments precisely describe the transitions that refer to different controller modes. In a later step these subsystems need to be refined on the component level in order to specify how the system shall be physically built. In the opinion of the authors this kind of detailed design is done best based on a domain-specific language as i.e. the ISO1219 in case of hydraulic circuits. In consequence a future engineering tool should integrate an object library of standard components with the graphical representation according the ISO1219. Additional attributes as i.e. material number and position number are needed to reflect the functionality of drawing schematics and generating parts lists as known from engineering tools like D&C Scheme Editor or ePLAN.

State	Entry Condition	E-Drive Speed	Cylinder Velocity
Apporach	$x = 0$	3000	0.50
Brake	$x > 1200$	3000	0.015
Press	$x > 1240$	1500	0.007
Keep Position	$x = 1300$	controlled	0
Slow Return	$\Delta t = 8$	controlled	0.01
Return	$x < 1290$	3000	0.2

Table 3. Desired Working Process

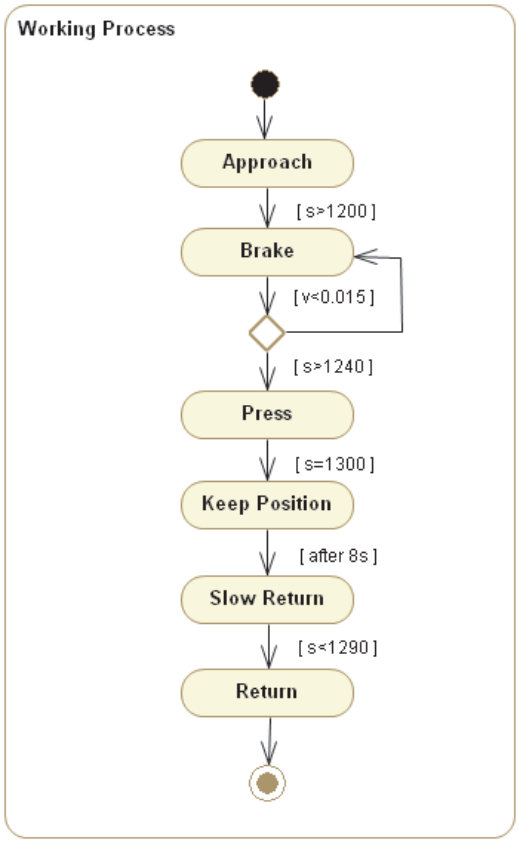

Figure 10. The Behavior of the System

Based on the captured requirements, shown in Figure 8, additional satisfy relationships from the design objects to the requirements have been established. In 11 this has been done exemplarily with the requirement P4: ApproachVelocity from Table 2. A much more expressive representation as relation matrix of design objects versus requirements is not shown due to the limited space. In the relation matrix, the SatisfyState of each requirement is checked to see whether all the requirements are considered in the design. With help of the relation matrix, the coverage of the requirements can be defined.

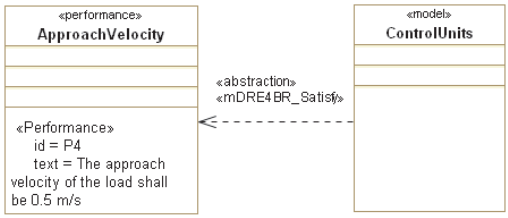

Figure 11. Building Satisfy Relation

5.3 Requirements Verification

The verification of the system behavior shall be performed by means of simulations based on the analytical model. This has been done exemplarily with the requirement P4: ApproachVelocity. This has been linked to a performance test case TestCase ApprochVelocity which is contained by the TestScenario1 as depicted in Figure 12. Moreover the

TestScenario1 contain a mathematical model which is used to stimulate the associated test cases. A performance test case is modeled further by a Modelica state machine as violation monitor to check the performance requirement ApproachVelocity (Figure 13).

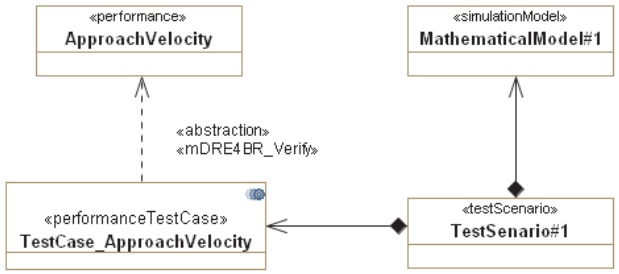

Figure 12. Verification Model

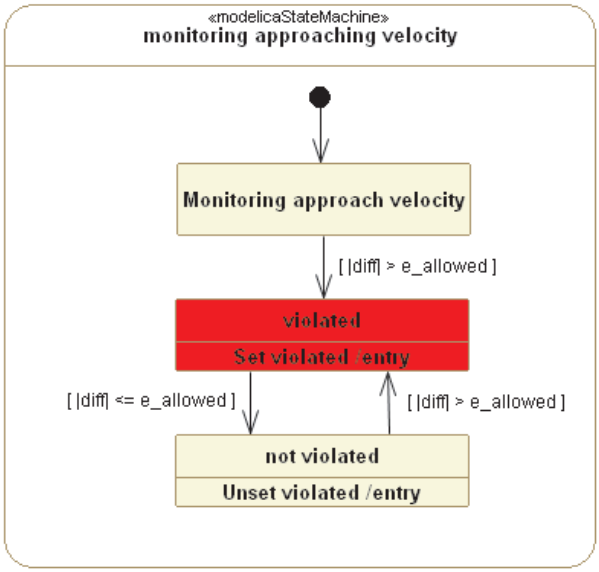

Figure 13. Modeling a Test Case

The mathematical model is defined by a Modelica model that has been build in Dymola. It consists of the four main parts: control unit, pressing system, driving system and sensor. The instances of the subsystem models are traced to the related objects of the design model. Since the focus of this paper is to present the model driven requirements engineering approach, the system simulation model will not be introduced in more detail here.

Currently, the binding of the corresponding variables between test cases and simulation models is very inconvenient. The future work shall address this issue in order to integrate the MDRE4BR profile with the ModelicaML code generator such that an automatic binding can be realized on the code level.

5.4 Verification Results

The simulation results referring to the requirement P4 is presented in Figure 15. It can be seen that, the approach velocity exceeds the desired velocity at the beginning. There-

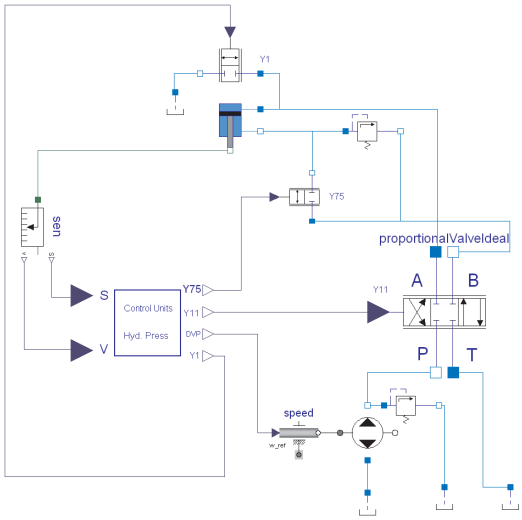

Figure 14. Object Diagram of Hydrostatic Press

fore this requirement is violated in this test scenario. The verifyState of the requirement is set to failed accordingly.

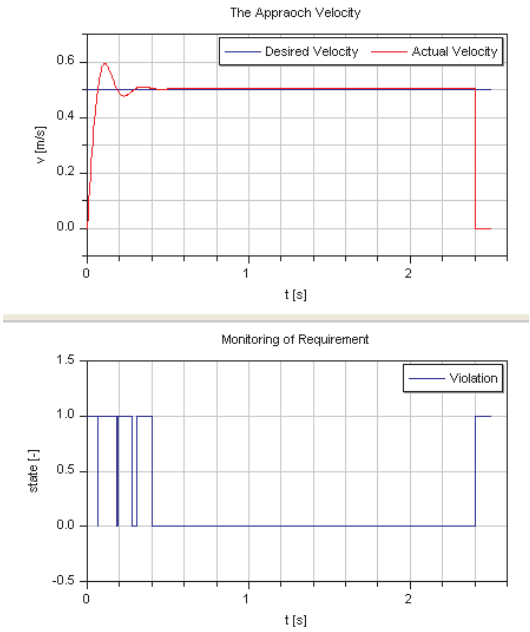

Figure 15. Verification of Requirement

In the same manner, all the other verifiable requirements can be verified. The verification results are summarized in the following figure (Figure 16).

6. Conclusion and Outlook

In this paper the model driven requirements engineering approach has been analyzed in the systems engineering context of industrial automation systems. Recent approaches integrating SysML and Modelica have been investigated according to their capability to describe a holistic model that covers the requirements engineering, the system design and the model verification and validation through simulation. Extensions and modifications have been presented as MDRE4BR profile. The concept has

ID	Description	?
P1	The maximum force of cylinder can not exceed180 kN.	
P2	The pressing time shall not exceed 8 s.	
P3	The cycle time shall not exceed 27s	
P4	The approaching velocity of the load shall be 0.5 m/s.	
P5	The pressure drop across the pump shall not exceed 340 bar.	
P6	The driving speed of the pump is 3000 rpm in approach phase.	
F1	The pressing pressure can not exceed 120 bar.	
S1	The number of driving cylinder is 2.	
O1	The moving mass shall be 40t.	

Figure 16. Summary of Verification Results

been demonstrated by a case study of a typical engineering project.

In the future work the tool support shall be improved. This refers to the establishing of the relations between design model and analytical model, as well as the relations between test case and mathematical model.

Acknowledgments

This work is funded by the Bosch Rexroth AG and the German Federal Ministry of Education and Research (BMBF) in the ITEA2 OPENPROD project.

References

[1] Bosch Rexroth AG. D&C Scheme Editor Manual. Technical report, Bosch Rexroth AG, 2010.

[2] Hubert Dubois, Marie-Agnès Peraldi-Frati, and Fadoi Lakhal. A Model for Requirements Traceability in a Heterogeneous Model-Based Design Process: Application to Automotive Embedded Systems. In *15th IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 2010, Oxford, United Kingdom, 22-26 March 2010*, pages 233–242, 2010.

[3] Martin Glinz. On Non-Functional Requirements. In *15th IEEE International Requirements Engineering Conference, RE 2007, October 15-19th, 2007, New Delhi, India*, pages 21–26, 2007.

[4] Eric Herzog and Asmus Pandikow. SysML - an Assessment. In *Proceedings of the 15th INCOSE International Symposium*, 2005.

[5] Marcos Vinicius Linhares, Alexandre Jose da Silva, and Romulo Silva de Oliveira. Empirical Evaluation of SysML through the Modeling of an Industrial Automation Unit. In *Proceedings of 11th IEEE International Conference on Emerging Technologies and Factory Automation*, 2006.

[6] Jürgen Lüttin. Bosch Requirements Engineering Framework - Overview. Technical report, Robert Bosch GmbH, 2010.

[7] `http://www.modelica.org`.

[8] `http://www.omg.org`.

[9] `http://www.eplan.de`.

[10] Christiaan J.J. Paredis, Yves Bernard, Roger M. Burkhart, Hans-Peter de Koning, Sanford Friedenthal, Peter Fritzson, Nicolas F. Rouquette, and Wladimir Schamai. An Overview of the SysML-Modelica Transformation Specification. In *2010 INCOSE International Symposium*, July 2010.

[11] Adrian Pop, David Akhvlediani, and Peter Fritzson. Towards Unified System Modeling with the ModelicaML UML Profile. In *Proceedings of the 1st International Workshop on Equation-Based Object-Oriented Languages and Tools (EOOLT'07)*, pages 13–24, 2007.

[12] Wladimir Schamai. Modelica Modeling Language (ModelicaML). Technical report, EADS Innovation Works, Germany, 2009.

[13] Wladimir Schamai, Peter Fritzson, Chris Paredis, and Adrian Pop. Towards Unified System Modeling and Simulation with ModelicaML: Modeling of Executable Behavior Using Graphical Notations. In *Proceedings of the 7th International Modelica Conference, Como, Italy, 20-22 September 2009*, number 43 in Linköping Electronic Conference Proceedings, pages 612–621. Linköping University Electronic Press, Linköpings universitet, December 2009.

[14] Wladimir Schamai, Philipp Helle, Peter Fritzson, and Christiaan J. J. Paredis. Virtual Verification of System Designs against System Requirements. In *Models in Software Engineering - Workshops and Symposia at MODELS 2010, Oslo, Norway, October 2-8, 2010, Reports and Revised Selected Papers*, pages 75–89, 2010.

[15] `http://www.omgsysml.org`.

[16] VDI. Design Methodology for Mechatronic Systems (VDI 2206). Technical report, VDI, 2004.