Integrating the 3+1 SysML View Model with Safety Engineering

Kleanthis Thramboulidis[1] *Member, IEEE,* Sven Scholz[2] *Member, IEEE*

[1] University of Patras, Electrical & Computer Engineering, Greece,
[2] University of Dresden, Transportation Systems Engineering, Germany
thrambo@ece.upatras.gr, Sven.Scholz@tu-dresden.de

***Abstract*—System safety is the property of the system that characterizes its ability to prevent from hazards, which may lead to accidents or losses. Traditionally, system developers are not familiar with system safety analysis processes which are performed by safety engineers. One reason for this is the gap that exists between the traditional development processes, methodologies, notations and tools and the ones used in safety engineering. This gap makes the development of safety aware systems a very complicated task. Several approaches based on UML have been proposed to address this gap. In this paper, an approach to integrate safety engineering with a SysML based development process that is expressed in the form of the V-model, is presented. Preliminary hazard analysis is adopted and applied to a SysML based requirements specification of the mechatronic system that exploits essential use cases. A case study from the railway domain is used to illustrate the proposed approach.**

I. INTRODUCTION

IEC standards such as the 61131 and 61499 define the languages to be used for the implementation of control and automation systems. The 61131 is widely used by the industry and the 61499 is proposed to address the always increasing complexity of today's applications [1]. A lot of work has been done to integrate domain specific languages such as 61131 and 61499 with the Unified Modeling Language (UML) in order to provide a more effective development process. UML has been successfully used in many real-time and embedded domains, including aerospace, military and manufacturing even within safety critical contexts. This was the reason for using UML in safety engineering. However, UML imposes several constraints for the modeling of mechatronic systems. These constraints are addressed by SysML, a general purpose modeling language for systems engineering applications.

Several industrial standards were developed for system and software safety. Some of them are of general purpose, such as the IEC 61508-3 [2], while others are domain specific such as the CENELEC 50128 [3] and IEC 62227 [4] for the railway domain. According to these, a safety-critical system is a system that may contain electronic, mechanical, and software aspects, and presents an opportunity for accidents to occur. Testing and verification is of great importance for these systems and fault tolerance is also used to increase their reliability. However, reliability does not necessarily guaranty safety. A system is safe if it does not generate hazards that cause accidents to or harm its environment and if it prevents from hazards that it is exposed to. A hazard is defined as a condition which can lead to an accident that may result in a loss in human life. Such a condition can be the state of the system combined with external environmental conditions. Hazards are normally classified as to severity. For example, the IEC 61508 [2] identifies four severity levels: catastrophic, critical, marginal, and negligible. The risk of a hazard is defined as the product of its probability of the occurrence (likelihood of occurrence) and its severity, i.e. $Risk_{hazard}$ = $probability_{hazard}$ x $severity_{hazard}$. Safety engineering is concerned with the safety issues of this kind of systems. There are several methods for performing a safety analysis such as Fault Tree Analysis (FTA), Failure Modes and Effects Analysis (FMEA), Hazard and operability studies (HAZOP) and Functional hazard assessment (FHA). FTA, for example, is a common technique for analyzing how faults lead to hazards and how to add safety measures to address these concerns [5].

When domain specific languages such as 1131 and 61499 are used for the development of safety critical systems such as airplanes, ships, trains, medical systems, and manufacturing systems that directly affect human life and impose a more rigorous development process, consistency with industry safety standards is mandatory. In [6] where a safety related evaluation of programming language constructs is given, the function block diagrams of 1131 that are based on verified libraries are considered suitable for safety-related control applications that should meet safety requirements of different SILs according to IEC 61508. Function block diagrams that are developed on the basis of verified libraries, such as the safety library of PLCOpen (http://www.plcopen.org), are suitable to meet the requirements of the second upper Safety Integrity Level, i.e. SIL 3. We are not aware of any evaluation of 61499 regarding safety but the use of multitasking by the run-time environments of this paradigm [7] is a great constraint even for SIL 2.

It should be noted that the use of 61131 function block paradigm even in the case of using safe libraries does not guaranty the safety level of the system. Safety analysis should be performed during development and the integration of the safety analysis with the traditional development process of 61131 or 61499 based industrial automation systems is not obvious. Safety analysis may be performed in the early phases of system development to increase the effectiveness of the development process. Hazards may be identified before and during the development process of the system and safety measures must be defined to reduce the risk imposed by the use

978-1-4244-6850-8/10/$26.00 ©2010 IEEE

of the system.

Traditionally, safety analysis is carried out by safety engineers and software and system engineers are not familiar with safety and certification aspects that make the communication between them a difficult task. An integration of the two processes will bridge the gap that exist between the two disciplines and will provide a framework for a more effective collaboration between experts. The objective of this paper is to describe an effective integration of the 3+1 SysML view model, which provides a framework for using 61499 or 61131 with SysML [8], with the safety engineering process. The integration of the safety engineering into the Mechatronic System (MTS) V-model is considered prerequisite for the development of systems that have to be compliant with various safety standards. SysML is adopted as modeling notation for the system engineering process that has been modeled by the MTS V-model of the 3+1 SysML View Model paradigm [9]. The main idea is to capture the most important safety critical concepts on the system model, i.e. the SysML model, and allow the safety analysis to be done on the mechatronic level. Preliminary Hazard Analysis (PHA) is adopted to perform a safety analysis very early in the development process using as input the requirements specification of the system. Safety analysis methodologies may be adapted to use as source information requirements expressed using UML use cases as is the case for [10]. A more effective approach for capturing requirements at the system level using SysML is used in this paper to provide a more solid background for performing safety analysis on the system model.

The remainder of this paper is organized as follows: Section 2 briefly presents the MTS V-model and discusses the use of UML for the development of safety related systems with main focus on the existing related work in this domain. Section 3 briefly describes the proposed integration of the 3+1 SysML view model with safety engineering. Section 4 presents the solution-independent safety analysis that is applied to the first phase of development and results to the integration of the preliminary hazard analysis with the MTS requirements specification process. A case study from the railway domain is used to present the proposed integration. The solution dependent safety analysis is presented in section 5 and the paper is concluded in the last section.

II. Background And Related Work

A. The 3+1 SysML view model

The 3+1 SysML architectural view-model [9] addresses the synergistic integration of the constituent parts of mechatronic systems. The main view of this model is the SysML one that corresponds to the mechatronic layer of the MIM Architecture [11]. This view captures the system model that is the one constructed by the MTS developer. Each of the other three views is used to describe the system from the perspective of the corresponding discipline. Specific tools of every discipline are exploited for the model execution and analysis of the SysML models of the mechatronic system. In [12] the 3+1 SysML view model is applied in the industrial automation domain using the evolving domain specific IEC61499 function block model and Modelica which is an object-oriented language for describing differential algebraic equation (DAE) systems combined with discrete events.

The V-model from the software domain was adopted and extended to address the needs of mechatronic systems development. The resulting MTS V-model is shown in Fig. 1. According to this the system modeling process is applied down to the primitive mechatronic component (MTC) level, as shown in the left-hand part of the V-model. The MTC is defined as stereotype based on the SysML block construct. For primitive MTCs that have to be constructed, a concurrent engineering process of all three constituent parts, i.e., mechanics, electronics and software is adopted, as depicted in the bottom of the MTS V-Model. The system integration and verification process is depicted by the right-hand side of the MTS V-model.

MTS-level requirements are captured using the SysML requirements diagram. Functional requirements captured at the MTS level are refined by essential use cases that are used at this level to represent an abstract, simplified, and independent of technology and implementation, way of representing the functional requirements.

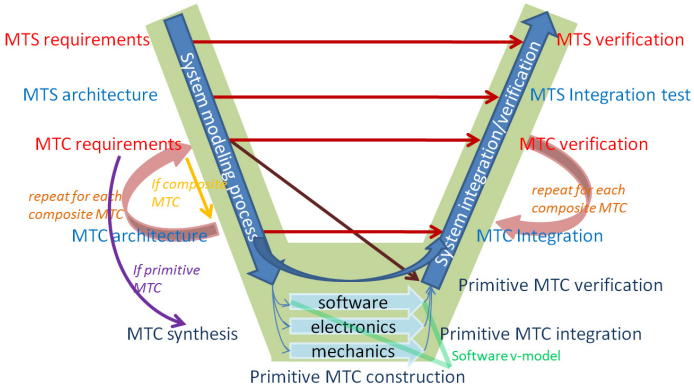

Fig.1. The mechatronic system V-Model (MTS V-Model) [9].

B. Integrating Safety Engineering with UML based development processes

A lot of work that has been done in the direction of using UML to represent the safety engineering concepts on the system model that is generated by a traditional development process that utilizes UML. Most of these works attempt to define a UML profile for safety-critical systems. A UML profile allows software engineers to capture safety related concepts and properties on UML models.

Authors in [13] describe a proposal toward the definition of a UML profile for the dependability assessment of real-time and embedded systems which, as they claim, is compliant with the RFP for a new UML profile for "Modeling and Analysis of Real-Time and Embedded Systems" (MARTE). The authors claim that they have exploited the best practices proposed in the literature, on extending UML with dependability modeling capabilities in order to draw up a check list of requirements that was used as guideline for the definition of a dependability analysis profile. This profile can be used, according to the

authors, to enrich the UML models with dependability requirements and properties and its main objective is to support the transformation of the UML-DA annotated models into suitable dependability models, such as fault trees, Bayesian networks, and stochastic Petri Nets, for the quantitative assessment of the system. In the same paper the authors present a case study where the DA profile is used to specify the dependability requirements and properties in the UML design of a gas turbine control system for safety and availability analysis.

In [6] the authors recognize the strength of UML to describe the overall architecture of the application and the strength of the Safety Critical Application Development Environment (SCADE) to formally describe the behavior of safety-critical software parts. Based on this they propose an approach that combines UML and SCADE for the development of safety-critical systems. They use UML to specify the system's high-level requirements and architecture, and then they use SCADE to formally specify the software behavior. In this way, as authors claim, a seamless flow from the initial requirement analysis phase down to the final integration on the target platform is provided. This flow, based on the connection of UML tools with the SCADE environment, allows the developer to exploit industry standards such as UML2, XMI, and DO-178B in order to provide a solution that is exactly tailored to the specific needs of safety critical projects. A complete workflow that combines the respective strengths of UML and SCADE to tackle large-scale safety-critical system development is also presented.

Authors in [14] in order to cope with the complexity and safety-related requirements of embedded real-time systems, propose to combine in their development process well-proven subsets of programming language structures fulfilling the respective requirements of the four Safety Integrity Levels of IEC 61508 with modern techniques of object-orientation. This, according to the authors, will allow the designers to both design abstract solutions based on the systems requirements, and implementation solutions based on appropriate subsets. The main objective of their work is to design systems in such a way that component faults do not result in system failures. They recognize that the architecture is also a crucial element in the life-cycle of safety critical systems, using as argument the possibility to provide well-known safety mechanisms such as forms of design diversity, e.g., N-version programming. They base their approach on a V-model of the traditional development process and propose an integration on this model with the safety engineering processes. As a solution they present a UML profile based on standards to enable the description of safety-related software. This profile extends UML by constructs oriented at the well designed and industry-proven facilities available in the "Process and Experiment Automation Realtime Language" (PEARL) that has been enhanced towards distributed systems and object-oriented design of applications having to meet severe safety requirements. The proposed extension is also oriented at the high-level graphical language Function Block Diagram (FBD) for programmable logic controllers as defined in the international standard IEC 61131-3. The objective is to exploit the ideas incorporated in these languages and the safety elements defined by standards and combine these with software based structural patterns that foster on dependability for the model-based and object-oriented development of safety-related embedded real-time control systems. UML components and classes are used to model the elements in safety-related programming and tagged values and constraints that are assigned to components are used to represent safety-related conditions and guards. The presented UML profile fulfills, according to the authors, the requirements of the four Safety Integrity Levels of IEC 61508.

In [15] the authors present an approach to improve the line of communication between safety engineers and software engineers. They first identify a list of safety-related concepts that are extracted from RTCA DO-178B that is the software de-facto standard for commercial and military aerospace programs. They next define the requirements for such a profile and present a UML profile that meets these requirements and enables modeling based on the above concepts. According to the authors, this profile allows software engineers to model safety related concepts and properties in UML. The authors also present their approach to generate certification-related information from UML models. They use as a case study the development of a navigation controller subsystem of an aircraft to illustrate the proposed approach.

In [16] the authors identify that the application of HAZOP studies require significant manual effort and tedious work of several costly experts. To address this problem they propose a knowledge-based approach to support the HAZOP analysis and to reduce the required manual effort. The main ideas behind their work are, according to the authors, to incorporate in a rule base knowledge about typical problems in automation systems, in combination with their causes and their effects and then to apply this rule base by means of a rule engine on the description of the automated system under consideration. This results to a list of possible dangers regarding safety risks and performance reductions which can be used by the automation experts to improve the system's design. An example application is also presented to demonstrate the effectiveness of their proposal. In this example application the system design is given in the form of a UML class diagrams, and the HAZOP study focuses on hazards caused by faulty communication within the distributed system. It is clear that the authors apply safety analysis on the finalized design and they do not propose an integration of development process with safety engineering. However, the work presented by authors may be exploited in the context of this work on the safety analysis of the design model.

A new modeling language based on UML for model-based safety engineering is proposed in [17] to formalize safety-related concepts and also show their relations with system modeling. The proposed model is applied to accidents and it is used to assess their tolerable frequency. Risk analyses are in the focus of this paper and safety issues, e.g. hazards and hazard's consequences are expressed in UML terminology and notation, but there is no link to the design model.

III. The Proposed Integration

We have adopted Preliminary Hazard Analysis (PHA) and apply it using as source information the MTS requirements, as shown in Fig. 2, where a draft model of the integration of safety engineering with the 3+1 SysML view model is presented. PHA and operations Hazard Analysis are methodologies that are usually performed at this stage of development and their objective is to identify hazards, determine their causes, and plan their elimination or mitigation.

The first activity of the safety engineer is to apply Hazard Analysis based on the MTS requirements which are composed of SysML requirements diagrams and essential use cases. During this process a list of hazards and their causes are identified. These artifacts are next used for the risk analysis, which is part of the PHA that is executed for each hazard. The result of this activity is the identification of the severity and the probability for each hazard and then the estimation of the associated risk. The process of safety measure definition complements this solution-independent phase of safety analysis and has as result an updated MTS requirements specification that is consistent with the safety requirements of the system. This specification is next used to define the system's architecture, i.e. the MTS architecture.

A second phase of safety analysis, a solution-dependent hazard analysis that results among others in the definition of required Safety Integrity Levels (SILs) for the system components and their assigned functions is performed at this stage. Then, hazard analysis is applied to every MTC iteratively down to the primitive MTC level. The hazard analysis for each MTC is composed of a PHA followed by a solution dependent HA. A detailed discussion on the solution dependent safety analysis is given in [18]. The following sections provide a detailed description of the above process using a case study from the railway domain.

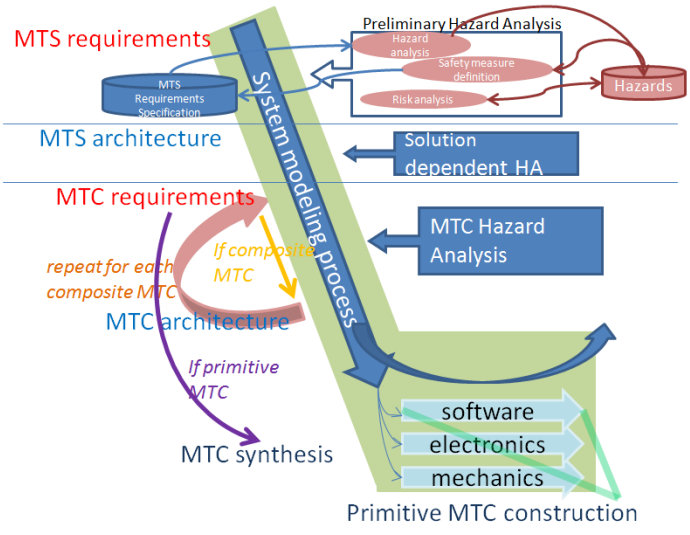

Fig. 2. A draft integration model of 3+1 SysML view model with safety engineering.

IV. The Solution Independent Safety Analysis

A. Safety Issues in Requirements Specifications

A requirement specifies a function that a system must perform (functional requirement) or a non-functional condition such as performance that the system has to achieve (non-functional requirement). The set of requirements that the system has to satisfy comprises its requirements specification.

Safety requirements, as for example those imposed by standards that cannot be expressed in the form of required system behavior, should be captured in requirements diagrams as non-functional requirements, such as MTTF, and be considered later during the development process as constraints. An example of product safety requirement in the railway domain is the "door release for passenger transfer" defined in IEC 62227 as "Passenger transfer doors shall be released for opening under normal conditions only in designated areas for passenger transfer if a pre-selection for the required side of the train was carried out, zero speed status is detected, and full train length occupies the platform area." Safety requirements include product requirements and not process specific requirements that are also defined by standards. These latter should be considered once during the definition phase of the development process. If properly taken into consideration during the definition of the development process, they will be "inherited" by every system that is developed using this process. In this way, the process forces the developer to satisfy these requirements and there is no need for explicitly stating these requirements for every system under development.

The most widely used technique for expressing functional requirements is through UML use cases. However, even though use cases are an effective tool for functional requirements they are not suitable for capturing non-functional requirements, that include performance and availability requirements, safety requirements, physical requirements, etc. SysML to address this problem has introduced the Requirement Diagram which can be used to capture functional and non-functional requirements. Fig. 3 presents part of the requirements diagram of an automated self-propelled train, a subsystem of an Automated Urban Guided Transport (AUGT) system.

There are several approaches that integrate UML use cases with the SysML requirement diagram. The most common is to consider use cases as model elements that are associated through the refine relationship with the specific requirements that they refine. As shown in Fig. 3 the Embarkation requirement with id 3.1 is refined by the Embarkation use case. Moreover, the steps or activities of a use case description may be captured as individual requirements in the requirement diagram to get a more granular traceability between the use case and the remaining model elements. This latter is related to the level of abstraction in use case definition.

We discriminate between core and derived requirements, where the term core is used to refer to the requirements that emanate from the stakeholder's perspective, while derived requirements emanate during the development process, as for example the ones that emanate during safety engineering. For

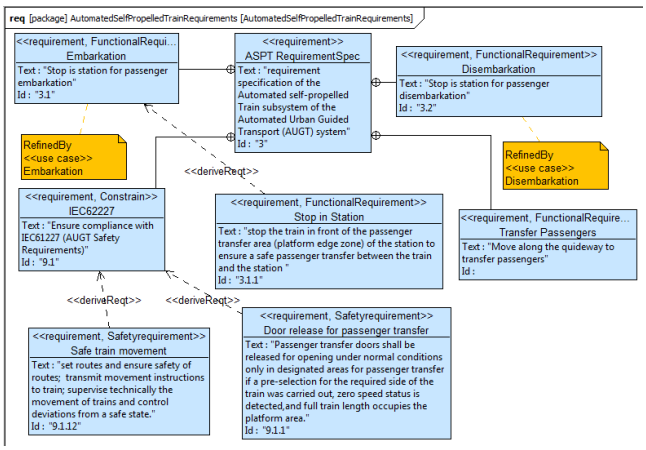

Fig. 3. Automated self-propelled train requirements (part).

example, safety requirements imposed by the IEC62227 standard are first captured on a system level as constraint as shown in Fig. 3 and then specific requirements defined by the standard and related to the system are captured during the safety engineering process as derived requirements using the derive relationship. Both core and derived requirements are stored in the same requirements repository for their better exploitation during the subsequent development phases. Requirement relationships defined by SysML enable the modeler to relate requirements to other requirements as well as to other model elements. These include relationships for deriving requirements, satisfying requirements, verifying requirements, and refining requirements. The refine relationship is used, as already mentioned, to link use cases with the requirement diagram. The concept of slave requirement is introduced by SysML to address the need for requirement reuse across product families and projects. The definition of requirement categories as stereotypes, such as safety requirement, enables the developer to add constraints that restrict the types of model elements that may be used to satisfy the requirement.

In the context of this work we have adopted the concept of the essential use case which provides for the preliminary phases of safety engineering, i.e. PHA, a more convenient representation of usage scenarios of the system under development. Essential use cases are abstract, simplified, and independent of assumptions about technology or implementation. They are written as an abstract dialog representing user intentions and system responsibilities, and they are typically small and focused on a highly specific user goal, yielding a fine-grained model of user activity [16]. Essential use case should identify and capture significant responsibilities that are not directly concerned with communicating with the user. Fig. 4 presents the *Embarkation* essential use case that will be used as an example in this paper. Embarkation and Disembarkation activities are covered by recently published standards for urban railway systems, such as IEC 62227 and IEC 62290 by the term Supervise Passenger Transfer. However, we avoid the use of the term *Supervise Passenger Transfer* in this stage since it already implies some kind of safety functionality.

For the system functions that result from the system responsibility part of the use case, their required QoSs should be defined. For example for the *Open passenger transfer doors* system function the time required to fully open the doors should be specified. This is a required QoS characteristic of this service. Another QoS characteristic considers the required safety level of this function that may be expressed as required Safety Integrity Level (required SIL).

From the use case description given in Fig. 4, it is clear that only the main or pure "positive" functionality of the system is captured; the behavior of the system in the case of an exception to this main functionality is not obvious. Most traditional development processes concentrate only on the basic functionality and neglect exceptions. Exception handling provides a mechanism to capture and handle exceptions to obtain system correctness while at the same time guaranty the clarity of the specification. However, not all exceptions are related with system safety. Misbehavior of passengers as well as misbehavior of the system should be taken into account in system development. Exceptions that do not produce any hazardous situation are not considered by safety engineers. However, for those misbehaviors that may lead to hazards, provisions have to be made to prevent injuries to passengers, as for example from misbehavior related with closing and opening the door leaves. It is clear that a systematic integration of safety engineering with the development process is required in order to satisfy system correctness and safety requirements.

The hazard "Doors open on the wrong side of the train" or "wrong travel direction assumed" can be handled as an exception in the Embarkation use case description by adding the statement *"If(doors open at wrong side) then …"* that should be placed after the statement *Open passenger transfer doors* in Fig. 4. However, to avoid the hazard "Doors open during driving", a continuous check on the door status should be done in every state of the system and not only in the context of this use case as exception handling. The doors status has to be supervised all the time and if the doors status "closed and locked" is lost while the train is driving, the train should stop.

passenger intention	*System Responsibility*
Identify direction	
	Stop in station
	Open passenger transfer doors
Embark	
	Wait for passengers to embark
	Close doors
	Depart

Fig. 4. The Embarkation essential use case.

B. The Preliminary Hazard Analysis

In order to apply PHA it is important to know for each use case not only the main actor that is the one that utilizes the specific system functionality, but every actor that interacts with the system in the context of the specific use case. This is due to the fact that misbehavior or failures and errors of these actors may cause hazards, as shown in Fig. 5, where the main

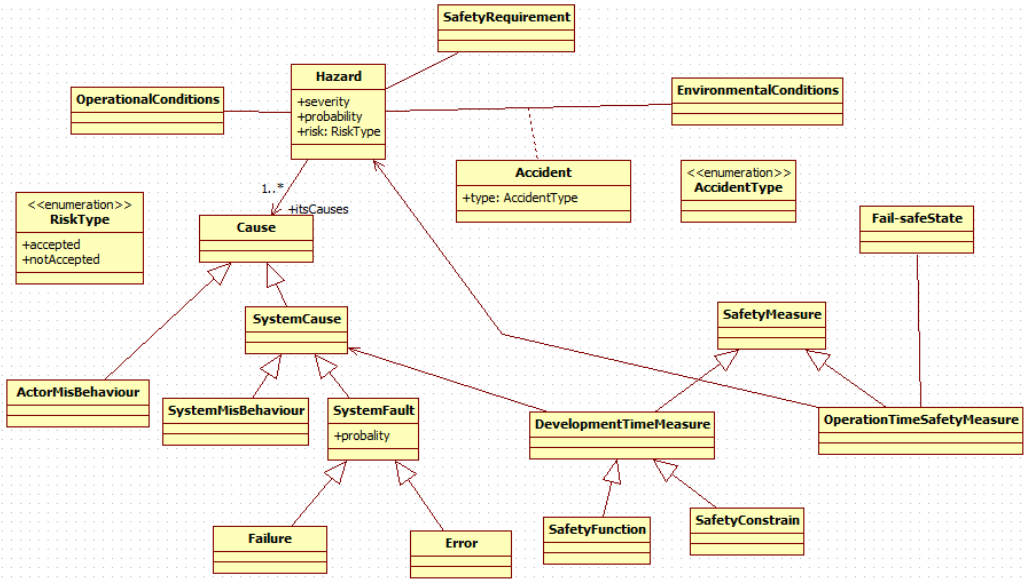

Fig. 5. Main concepts of the proposed integration model (part of the concept map).

concepts involved in hazard and safety analysis are captured. So, for the embarkation use case except from the main actor of the use case that is the passenger, we identify as extra actors the train driver and the platform and consider them during the hazard identification process. It is clear that a hazard may be caused either by a misbehavior of the system as defined by the essential use cases or a misbehavior of its actors, but also from a system failure. By the term *SystemMisBehavior* we refer, in the model of Fig. 5, to a misbehavior that results from the requirements specification. During operation-time a system fault may also result to a hazard but this is not originated by the requirements specifications. The objective of PHA is to identify the system and actor misbehavior as sources of potential hazards and to plan mitigation strategies for these hazards. System faults are also considered at this stage at a very high level of abstraction. System faults introduced by the specific implementation will be considered later.

First of all, the functionality that has been captured in the essential use case has to be examined for potential hazards that may be caused by a system or actor misbehavior. The application of the PHA on the requirements specification, i.e. the defined use cases, will result in a list of hazards and a list of causes per each hazard. For example for the *Stop in station* function and taking into account the involved use case actors we identify the following possible unintended behaviors: a) train does not stop in station (driver fails to stop the train or the train fails to stop), and b) train does not stop at the correct position, i.e. part of the train is outside of the station.

The result of these misbehaviors are considered only annoying for the embarkation use case and are not considered as immediate causes of hazards. The case in which such a behavior may be the cause for a hazardous situation especially in combination with other failures will be discussed later.

Analyzing the function *Open passenger transfer doors* we identify the following possible unintended behaviors and failures: a) doors do not open due to a mechanical problem (SystemFault, i.e. failure of train equipment), b) driver forgets to open the doors (ActorMisBehavior), and c) doors open to the wrong side which is either ActorMisBehavior (driver's mistake) or SystemFault.

It is obvious that only the third one may impose a hazard, while the others may only result in annoying situations. In case the train doors are opened to the wrong side, no platform might be present and even the adjacent track may be accessible from this side of the train. The associated hazard would then be "Passengers fall from train on track". So, this unintended behavior leads to a hazard, i.e. to a given system state that combined with environmental conditions, e.g. other track present, but also with operational conditions, e.g. train running on the adjacent track, of the system, at the time of hazard appearance, may lead to an accident, as shown in Fig. 5. For example, a human misbehavior to press the front wheel brake in a motorcycle leads the system to a state that given specific operational conditions such as high speed will result in a system state that is characterized as hazard. This hazard may lead to an accident only if combined with specific environmental conditions, e.g. ice on the road. To address all hazards related with door opening, the safety function "Authorize train doors opening" that will be responsible to supervise all necessary conditions for door opening, can be defined.

It is clear that the isolated analysis of each function may not be sufficient because certain safety critical states rise only when combining several functions, and therefore several system states. For example, the second failure of the *Stop in station* function may impose the "people fall out of moving train" hazard for the disembarkation use case if combined with the function "open doors".

The above artifacts, i.e. hazards and causes, are next used for the risk analysis that is executed per each hazard. The result of this activity is the estimation of the severity and the probability, i.e. the risk for each hazard. However, it should be mentioned that the estimated risk has to be interpreted with a reasonable error approximation. The safety engineer should

also identify the possible accidental consequences of the hazard.

For the mentioned hazards the following accidents types for the passenger are possible: a) injury due to fall, b) injury due to electrocution (in metro operation a third rail for power transmission next to the track is standard), or c) injury due to running train on adjacent track.

The risk of each hazard which is characterized according to its severity and probability is classified as accepted or not accepted. In case the risk associated to a hazard is considered accepted this hazard will not affect any more the development process, i.e. further strategies to mitigate the risk are not required. For the not accepted hazards the activity of defining the safety measures has to be executed. A detailed analysis should identify the different causes of each hazard and propose safety measures, i.e. recommendations to eliminate or mitigate the hazard, such as the "Authorize train doors opening". These measures may be classified into two categories: development-time safety measures and operation-time ones, as shown in Fig. 5. An operation-time safety measure enables an operator to handle during operation-time a system state that may lead to a hazard or accident. In most of the cases this is expressed in the form of operational procedures or rules on how to react on certain situations. Complementary, a technical measure, e.g. a supervision system may be implemented to run in parallel with the nominal system functions. On the other hand, development-time safety measures define the proper system behavior that is required to prevent hazards. This type of measures may result in modifications of existing use cases regarding the user interaction with the system but may also result in a set of derived requirements of two kinds: a) new safety constraints, and/or b) new safety related functions which the system should implement or even new use cases.

For example for the hazard "Person injured by closing doors" the inadequate pressure or force of the closing mechanism of doors may be responsible. However, this may be subject to more detailed causes, such as design errors (too high pressure calculated), incorrect maintenance of doors (too high pressure adjusted). As a consequence the people would be trapped and injured by the train doors. In this case the correct definition of the door pressure (as a safety constraint) during the design process as well as properly defined maintenance procedures would prevent this hazard. The first of these two measures would fall into the category development-time safety measure, while the second one is an operation-time safety measure and is related to actor misbehavior. The "Person injured by closing doors" hazard should also be considered in relation to the "depart" function to prevent the train from departing. For this case the "Supervise the Doors Closed and Locked Status" function could be a proper safety function to address this hazard.

An example of another safety related function is the "confirm vehicle stopped and brakes applied" that should be considered as prerequisite to the *Open doors* function to avoid hazards. This is also a development-time safety measure; it should be mentioned that development-time safety measures are exploited during operation-time.

At this point we have all the information that is required to provide a new updated version of the system's requirements specification that will ensure its required safety levels. All the above derived requirements, functional and non-functional, are captured in the requirements diagram in the same way as the core requirements and stored in the same requirements repository so as to be used along with the core requirements as source for the subsequent architecture definition process as shown in Fig. 2.

It is evident that iterations of the above process have to be executed in order to identify hazards introduced by these new derived requirements, mainly safety related functions and use cases, but also re-evaluate the risk of old hazards taking into account the proposed safety measures. It should be noted that until this point of time, technology dependent information has not been captured in requirements specification so the safety assessment is based only on technology independent models.

V. The Solution-dependent Safety Analysis

The so resulted set of requirements, core and derived, are subsequently used by the design team to propose a system architecture that should satisfy these requirements. The result is a proposal for a solution that is described by the architectural diagrams and later on by the detailed design specifications. A number of artifacts compose this specification. The system architect has to identify the mechatronic components (MTCs) that the system has to be composed of, as well as their interconnections through connectors. He has to either assign use cases to specific MTCs or define, using sequence or collaboration diagrams, the way that system components have to collaborate in order to provide the required by the use case functionality. So, usually a use case is realized by means of sequence or collaboration diagram that is based on the structural model of the system. Fig. 6 presents the architecture that was defined for the system to satisfy the Embarkation use case. At this stage the required safety integrity level (SIL) of the functions that have been assigned to each MTC should be defined. This topic is further discussed in [18].

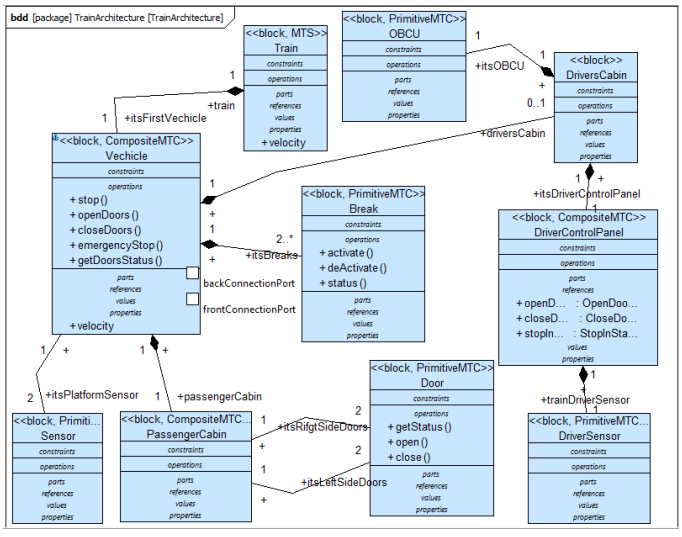

Fig. 6. System architecture for the Embarkation use case.

The architectural design has also to pass from safety assessment to identify possible hazards that have been introduced by the specific solution. This is why the safety engineering activity in this step, shown in Fig. 2, is called solution-dependent Hazard Analysis. Several faults introduced by the specific collaboration of system's MTCs may lead to hazards. Sequence as well as collaboration diagrams are expected to be considered source information in the solution-specific hazard identification process. They allow the engineering and safety staff to understand how components' faults propagate through the system, and propose corrective actions to eliminate or mitigate these faults but also to interrupt the fault propagation in the system. In order to eliminate or mitigate components faults, low-level redundancy may be added to the component. For example, it is possible to add CRC on the attributes of the software part of the MTC, or adopt data replication, and define precondition and post condition checking, to ensure that safety-relevant faults are identified at the component level and handled appropriately. These actions constitute the safety analysis process of the specific design. Fault Tree Analysis (FTA) is a common technique for analyzing how faults lead to hazards and how to add safety measures to address these issues. A major outcome of this solution-dependent safety analysis is a Safety Case which demonstrates that the system fulfills all functional safety requirements set up during the Solution-Independent Safety Analysis.

VI. Conclusion

Mechatronic systems usually affect human life and are characterized as safety critical Examples of such systems include among others airplanes, ships, nuclear reactors, medical systems, and manufacturing systems. A more rigorous development process is required for this kind of systems. The gap that exists between safety engineering and traditional system developers complicates even more the development process. SysML attempts to address the complexities of the development process of mechatronic systems and has become the emerging standard for the modeling of this kind of systems. To address the communication gap between SysML based modeling of the system and the modeling of its safety related aspects an integration of both processes is required. This integration should focus not only on the process but also on the notations used.

In this paper, an approach to integrate safety analysis with the 3+1 SysML view model, a SysML based architectural approach for mechatronic system development has been presented. Essential use cases are adopted and combined with the SysML requirements diagram to form a convenient base for a solution-independent hazard analysis, i.e. preliminary hazard analysis. This process results in a requirements diagram that will integrate not only safety requirements but also safety measures to provide a coherent repository of the system requirements that will be used for the definition of the system's architecture. Subsequent phases of safety analysis, e.g. FTA or FMEA of the architecture, will also be integrated with the remaining phases of the MTS V-model to constitute a framework for an efficient integration of the traditional development process with safety engineering. In this regards further work is needed to combine the Solution-Independent Safety Analysis with the Solution-Dependent Analysis step.

Acknowledgments

Part of this work has been funded by TIKOSU, a project belonging to the Digital Product Process program of the Finnish Funding Agency for Technology and Innovation (TEKES). The authors would like to thank Jarmo Alanen for fruitful discussions on the subject.

References

[1] Thramboulidis, K., "The Function Block Model in Embedded Control and Automation: From IEC61131 to IEC61499", WSEAS transactions on computer, Issue 9, Volume 8, September 2009.

[2] International Electrotechnical Commission (IEC), IEC 61508 – Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems, December 1998

[3] European Committee for Electrotechnical Standardisation (CENELEC), EN 50128:2001 – Railway Applications: Software for Railway Control and Protection Systems

[4] International Electrotechnical Commission (IEC), IEC 62227 Railway Applications Automated Urban Guided Transport (AUGT) Safety Requirements, IEC 2006.

[5] Douglas "Build Safety-Critical Designs with UML-based Fault Tree Analysis", Embeddeed.com, Available on-line: http://www.embedded .com/design/217200222

[6] Shourong Lu, W. A. Halang, *"A UML Profile to Model Safety-Critical Embedded. Real-Time Control Systems",* Contributions to Ubiquitous Computing, Volume 42/2007, pp. 197-218.

[7] Thramboulidis, K., "IEC61499 Function Block Model: Facts and Fallacies", IEEE Industrial Electronics Magazine vol. 3, issue 4, December 2009, pp. 7-26.

[8] Thramboulidis, K., Buda, A. "3+1 View Model for IEC61499 Function Block Control Systems", 8th IEEE Inter. Conf. on Industrial Informatics, 13-16th July, Osaka, 2010.

[9] Thramboulidis, K., "The 3+1 SysML View-Model in Model Integrated Mechatronics", Journal of Software Engineering and Applications (JSEA), vol.3, no.2, 2010, pp.109-118.

[10] Allenby, K. Kelly, T. "Deriving safety requirements using scenarios", Proceedings. Fifth IEEE International Symposium on Requirements Engineering, 2001, pp 228-235.

[11] Thramboulidis, K., "Model Integrated Mechatronics – Towards a new Paradigm in the Development of Manufacturing Systems" IEEE Transactions on Industrial Informatics, vol. 1, No. 1. February 2005.

[12] Bernardi, S. and Merseguer, J. "A UML profile for dependability analysis of real-time embedded systems", 6th international Workshop on Software and Performance, Buenes Aires, Argentina, February 05 - 08, 2007. (WOSP '07) ACM, New York, NY, 115-124.

[13] A. Le Guennec, B. Dion, "Bridging UML and safety-critical software development environments", Int. Conf. on Embedded and Real-Time Software, ERTS, 2006.

[14] Zoughbi, G., L. Briand and Y. Labiche, "A UML Profile for Developing Airworthiness-Compliant (RTCA DO-178B), Safety-Critical Software", Model Driven Engineering Languages and Systems, Volume 4735/2007, pp. 574-588.

[15] Schreiber, S. Schmidberger, T. Fay, A. May, J. Drewes, J. Schnieder, E. H.Schmidt, "UML-based safety analysis of distributed automation systems", IEEE Conference on Emerging Technologies and Factory Automation, ETFA 2007, 25-28 Sept. 2007, pp: 1069-1075

[16] Constantine, L., "Activity Modeling: Toward a Pragmatic Integration of Activity Theory with Usage-Centered Design", Technical Paper, Available on-line: http://www.foruse.com/articles/activitymodeling.pdf

[17] Cancila, D.; Terrier, F.; Belmonte, F. et al., "SOPHIA: a Modeling Language for Model-Based Safety Engineering", MoDELS'09, ACES-MB Workshop Proceedings, Denver 2009, pp 11-25.

[18] Scholz, S., Thramboulidis, K., "Integrating SysML-based System Modeling with Safety Engineering", Design Automation and Test in Europe (DATE), Workshop on Model Based Engineering for Embedded Systems Design, 8-12 March, 2010, Dresden, Germany, pp. 1-8.