Safety Oriented Software Engineering Process for Autonomous Robots

Vladislav Gribov, Holger Voos
Faculty of Science, Technology and Communication,University of Luxembourg
6, rue R. Coudenhove-Kalergi, L-1359 Luxembourg, Luxembourg
{vladislav.gribov,holger.voos}@uni.lu

Abstract

In this paper, a safety oriented model based software engineering process for autonomous robots is proposed. Herein, the main focus is on the modeling of the safety case based on the standard ISO/DIS 13482. Combined with a safe multilayer robot software architecture it allows to trace the safety requirements and to model safety relevant properties on the early design stages in order to build a reliable chain of evidence. The introduced engineering processes consist of the Domain Engineering, which is dealing with the development of a set of interlinked formalized safety cases and software components. Finally, the proposed engineering process is demonstrated on the example of the assembly assistant robot and ROS (Robot Operating System).

1 Introduction

Safe service and personal care robots have become a new exciting research topic over the last years. However, the close interaction of humans and robots also leads to completely new safety problems. Physical segregation of robots and humans works fine in an industrial environment with stationary robot manipulators [21], accompanied by the corresponding safety trainings for personnel. For obvious reasons, such measures are not possible if pHRI (physical human-robot interaction) is required by the robot application and if no intrinsically safe robot design is possible. Autonomous robots and (untrained) humans sharing the operation space and cooperating with each other create new kinds of risks and requirements with regard to robot safety. Safe human-robot interaction during the autonomous operation becomes essential and necessary [10], but also requires new safety standards.

The request for a new safety standard is followed by the request for a corresponding practical engineering process, especially for the robot software (SW). The existing service robots, both commercial and academic solutions, are mostly designed in a more intuitive way without providing a clear and safety-oriented engineering process for robot design. The future plans for a wide deployment of service and personal care robots also suggests limited costs for the robot development phase. However, this is only possible if an engineering process is applied which deals with safety requirements in a formal way and reuses safety solutions.

2 State of the Art

If approaching safety issues in general, according to [11] the best way to deal with safety is the definition of a safety case, which "should communicate a clear, comprehensive and defensible argument that a system is acceptably safe to operate in a particular context". A safety case is constructed from objectives, arguments and evidence. The formalization of the safety case could be done with Goal Structuring Notation (GSN), which is used to support safety case management on different stages: preliminary safety argument, safety cased development, change management, maintenance.

A product family is an approach to reuse safety experience from one product for another new product from the same family[13]. It comprises ontological methods - defining domain and vocabulary - as well as the establishment of domain assumptions - mainly finding common and variable features of systems in the domain. The variations are formalized and parameterized to build a requirements model. The product family safety requirements are derived after preliminary hazard analysis using reviews and fault tree analysis. The safely reuse of the engineering artifacts for Safety-Critical Product Lines was also introduced in [9]. The model based approach leads to a safety meta-model which captures the dependencies between the safety case, the safety assessment and the development of the artifacts in a product line.

Another approach to integrate safety engineering in a SysML based model-driven development process in the form of the V-model is proposed in

978-1-4799-0864-6/13/$31.00 ©2013 IEEE

[22]. The SysML, addressing the needs for modeling mechatronic systems better than UML, provides the requirements diagram which can be fused with IEC 61131 and IEC 61499 languages parametrized, extended with FTA (fault tree analysis) and HAZOP (hazard and operability study) and combined with IEC 61508. The proposed V-model has several layers and especially deals with mechatronic system requirements and architecture.

Modeling of the safety standard IEC 61508 and of the chain of evidence is also discussed in [18]. Here the main focus is on providing a chain of evidence already on the design stage and not post-factum, as it is still made in practice. Other topics are the demonstration of the traceability for a seamless chain of evidence and to make a step from the generic safety model to a specific context.

The systematic hazard analysis applied to the UML component and deployment models as proposed in [8] is an approach to integrate safety and component failure issues to UML design.

In the application of mobile service robots, the human robot interaction becomes important and even unavoidable during the autonomous operation [10]. Therefore, the existing standards for industrial robots (ISO 10218 [7]) have to be extended and updated for service robots. The ISO/DIS 13482 [3] is an approach to do that, making accent on pHRI and safety issues due to the robot mobility. Here non-industrial robots are addressed, explicitly avoiding an industrial environment with its more conservative safety requirements.

Safe reuse of the commercial off-the-shelf software is another topic which is especially important for the practical purposes according to [14].

A number of robotics SW approaches and frameworks has been developed so far in academics, to name the most successful: Orocos[6], CLARAty [16], Orca [5], ROS[15]. Taking the number of successful implementations in different robotic applications into account, ROS seems to be the most promising of all robotic software frameworks and comes with the widest community support.

However, the engineering process for robot software is a less discussed issue. It has been fragmentarily addressed as component-based SW engineering in Orca and CLARAty. A component-based engineering approach for mobile robot applications, SmartSoft, is also proposed in [20]. The authors claim that the complexity and variety of domains in robotics and the huge problem and solution space demand new engineering approaches in order to create synergies between domains and reduce time and costs of the development process. The key to handle the complexity are robustness by design and robustness by adaptation. The model-driven paradigm seems to be suitable and is a base for SmartSoft, supporting modeling from platform independent model (PIM) through platform specific model (PSM) to platform specific implementation (PSI). The provided Eclipse-based toolchain is compatible to several middleware solutions and is able to deal with features like real time, dynamic wiring and online reconfiguration. However, an explicit consideration of safety aspects is not included.

Another model-driven approach for robot software engineering is proposed in [4], introducing a tool chain and component-based architectural models and correspondent domain specific language.

Finally, the BRICS project [12] presents a very detailed Robot Application Development Process, but still without an accent on safety of the robot software.

3 Proposed approach

The previous overview already suggests a direction for the research on a safe engineering process for robotics. Herein, we assume that it is possible to cover a wide range of robotic applications by using the same HW and only by changing the robot software. Therefore we will focus on the engineering of the robotics SW for that matter, keeping in mind however that SW is still only a part of the robot as a system. The questions to be answered are:

- how to represent the relevant safety issues in a formalized way
- how the safe robot SW architecture shall look like
- which engineering process is able to provide sufficient evidence of safety and a safe reuse of SW

3.1 Safety Model

Representing a safety case and a chain of evidence in a formal way has already been proposed as an effective way in safety engineering ([9], [22], [18]). We will continue on that while approaching the ISO/DIS 13482 [3] on a model-driven base. This safety standard addresses the class of non-industrial (mobile) service robots, providing safety requirements according to the risk assessment process described in the ISO 12100.

The conceptual model of the ISO/DIS 13482 is shown in Fig. 1. Strictly speaking, the risk assessment and the risk mitigation processes are described in the ISO 12100, while the ISO/DIS 13482 rather presents the result of applying those processes to the domain of service robots.

Risk assessment comprises the identification of the hazard from the different sub-domains of service robots and the evaluation of the risks assigned to those hazards. After the risk evaluation the decision for *risk mitigation* shall be made. Options for the protective measures are:

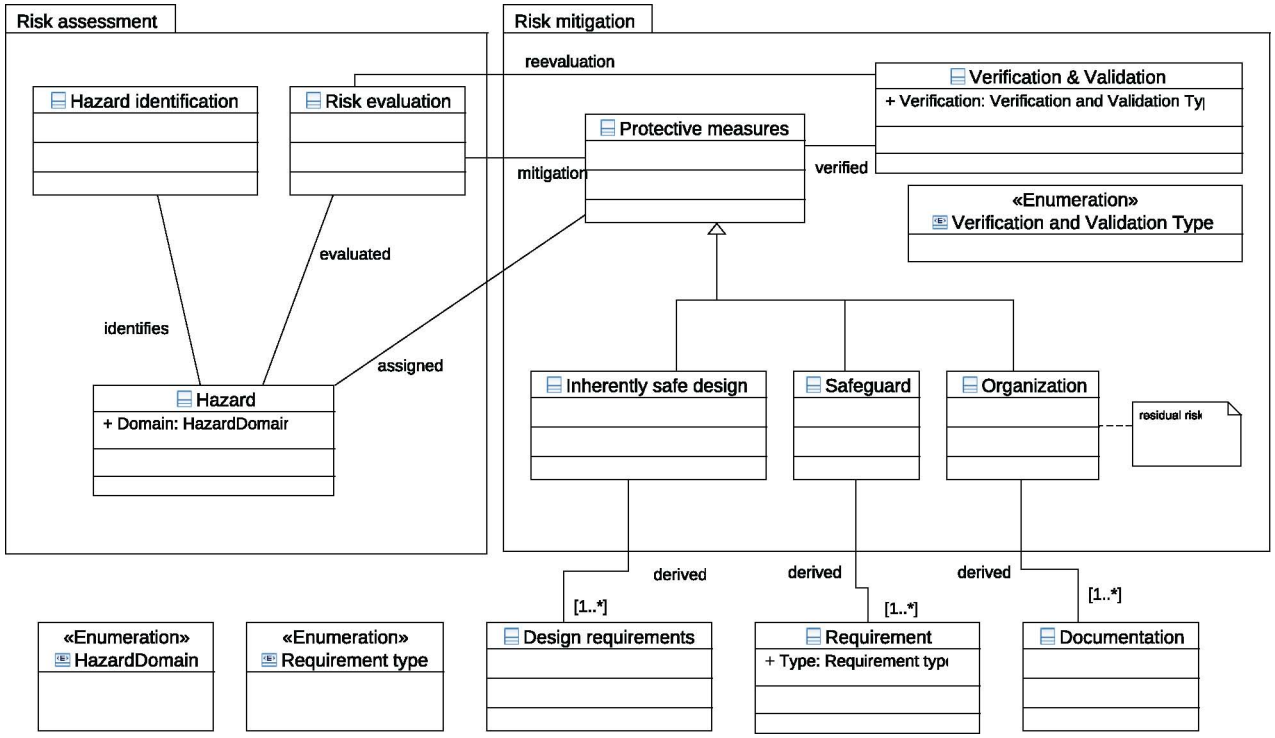

Figure 1. Core concepts of the ISO/DIS 13482

- Update the design to be inherently safe, which most probably leads us from the SW engineering to the higher level robot design, but still can be considered as a part of robot engineering process, described in section 3.3.
- Provide a safeguard addressing the corresponding hazard; could be a safety function or functional safety requirements for the affected software components.
- Organizational measures dealing with the residual risks, which are not covered by previous measures.

Protective measures shall be validated, the ISO/DIS 13482 suggest several methods for that. Applicable for the software engineering process are: test and measurements, review and documentation of the software function blocks.

The stereotypes from the UML profile of the standard are applied to the robot model [19], checking the model on the compliance to the constraints attached to the stereotypes and written in object constraint language. This allows to refine robot model by both updating and extending it to be related to safety standard evidence model.

Modeling the mentioned artifacts of risk assessment and mitigation with SysML allows to create a formal description of the chain of evidence for a specific safety case. The catalog of the safeguards together with addressed risks results in safety requirements. The fulfillment of the safety requirements is proven by a verification and validation process. The modeling is done by building a class hierarchy of the hazards. The hazards are attributed with risks. The safety requirements are assigned to that hazards, representing the safeguards which are intended to mitigate the risks of the hazards. The interconnection of the artifacts provides the traceability of the safety case. The next step here is to relate the software components to the safety requirements. This will include the implementation in the chain of evidence and creates a stronger argument.

The ISO/DIS 13482 covers a set of basic safety cases. However, specific robots may be affected by the risks which are not mentioned in the standard. This can happen due to limitation of the standard or due to development of robot application in new domains. Also the list of protective measures and verification methods may be extended. This shall be done in the context of the engineering process.

An example of applying this approach to model a safety case is shown in section 4.

In general, the described model-based approach to create a safety case can be used in the process of software engineering for a vast categories of robots. However, the software engineering process for service robots can especially profit from it due to the complexity of such a software.

3.2 Safe Robot Software Architecture

The complexity of the robot software depends on the tasks it performs. Autonomous behavior of the robot and the required mobility adds additional complexity to the SW. To deal with that complexity, most software architectures for autonomous robots propose a multi-level approach ([6], [5], [20]). Therefore, we also assume a multi-level software architecture in our approach which will be extended to include safety aspects.

The different tasks performed by the robot SW can be generally classified as "reflexive" (e.g. sensory-motor), "reactive" (e.g. local planning) and "proactive" (e.g. global planning). Those tasks are independent and in that sense decoupled, and form architectural layers of control SW, based on different safety and real-time requirements. The tasks with higher architectural layer functions may implement complex algorithms and depend on unreliable communication links, being in general difficult to validate or even not safe on their own. The software on lower architectural layers shall provide constrains and level of robustness to compensate intrinsically unsafe software from higher levels and to guarantee acceptable safety level of the system. Cohesive functionality of each layer can be naturally represented as components.

From the safety point of view software components can implement safety relevant functions (e.g. obstacle detection). The taxonomy of the components can be done based on the domain and concrete function in that domain. This is reflected in the corresponding properties of the component. The first try to provide the standard interfaces for the human-robot interaction and thus an ontology and a component decomposition for that domain is done by the Object Management Group under the Robotic Interaction Service [17].

The components implementing safety functions are also characterized by the provided functional safety, which is defined by the performance level. The appropriate robot architecture has to offer a sufficient set of safety features providing corresponding functional safety integrity levels (SIL) or performance level (PL) for each of them.

The representation of the component core model a the safety oriented robotic software is shown in Fig. 2.

Requirements concerning software components also have an impact on the hardware it runs on. Deployment of safety-critical software with high performance level shall be done on a hardware system that can provide the required performance level. This is also valid for any communication system involved.

For the implementation of the concepts of the component-based robotics software the corresponding component oriented robotics framework is required. The ROS software has proven over the last few years that it provides a solid base for the robot software development, especially for autonomous service robots. ROS is a multilingual, distributed, network transparent software, supported and extended by a large and active community. It is build upon concepts, which are shown in Fig. 3.

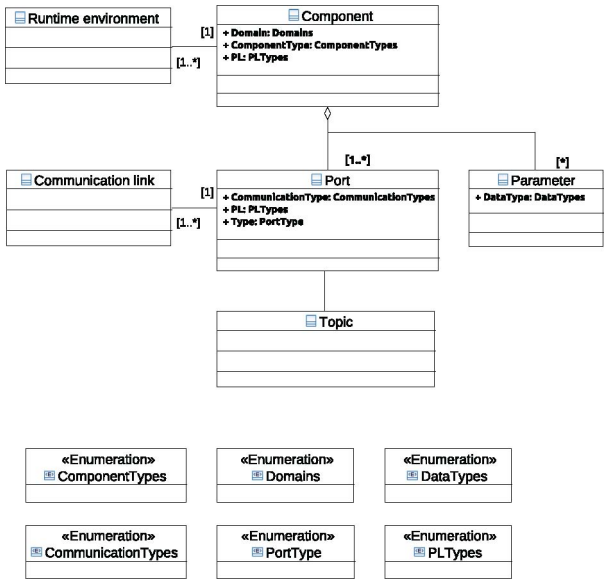

Figure 2. Component core model for the safety oriented robotic software architecture

The main software element is the ROS node. This is independent running SW component which can communicate to the other nodes by sending and receiving messages. This messages are called the topics. If sending specific data, a node is a publisher of the corresponding topic. If consuming the topic, the node is a subscriber. Nodes can also provide services, which perfectly supports confirmed synchronous peer-to-peer communication.

ROS fits very well for building robot SW according to a safe multilayer robot software architecture. Even if some extensions might be required for full covering the requirements of multilayer architecture, the concepts of ROS allows them. The ROS node corresponds to the software component, providing encapsulation of the functionality and autonomy. The ports are implemented by data and control flow using publishing/subscribing topics and services. Network transparency allows deployment of software components on different HW types, providing required performance level depending on the layer component is assigned to.

Current implementation of ROS is unfortunately not supporting real-time or safety features, but the request for that characteristics is already clear to the community. A ROS multi-master is aiming to achieve more reliable solutions based on ROS [2]. The

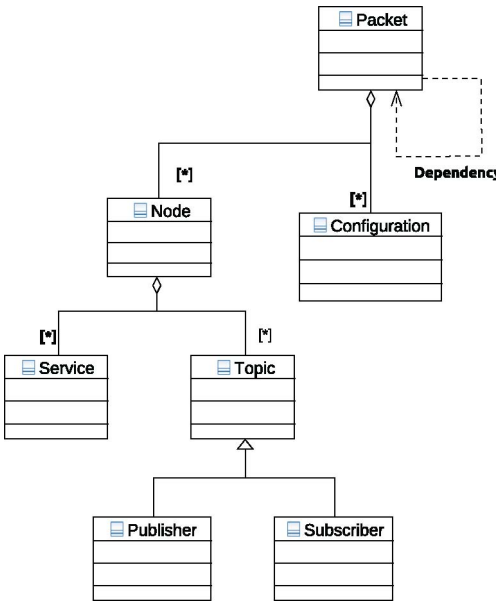

Figure 3. ROS core concepts

projects like μROSnode are focused on running ROS on the embedded hardware [1]. Deploying ROS nodes on an embedded platform allows to improve real-time characteristics of the corresponding component. The same is valid for the safety properties and the performance level of the component. All this measures are not conflicting with the proposed component based modeling approach. The embedded nodes or even whole subsystems still can be modeled as components and be interfaced to the rest of the robot SW, which is ROS based. The principle of representing robot software as a component model however is not affected by the current development stage of ROS. The complex safe robot architecture will not result in a homogeneous solution, both in terms of software and hardware.

3.3 Safety Oriented Engineering Process

The safety oriented engineering process shall give an answer on the question: "which (safety) requirements are relevant and which components shall be put together to satisfy those requirements?" We have already specified a model based approach to represent a safety case and proposed a software architecture of a safe robot. In addition, we propose an engineering process where repositories of modeled safety requirements and corresponding software components are used to build the robot SW, as shown in Fig. 4.

The proposed software engineering process for component-based robot software is similar to the software product line approach. The overall engineering process is divided into two separate but still interconnected sub-processes - development of software components and building applications out of them. The development of the software components is called Do-

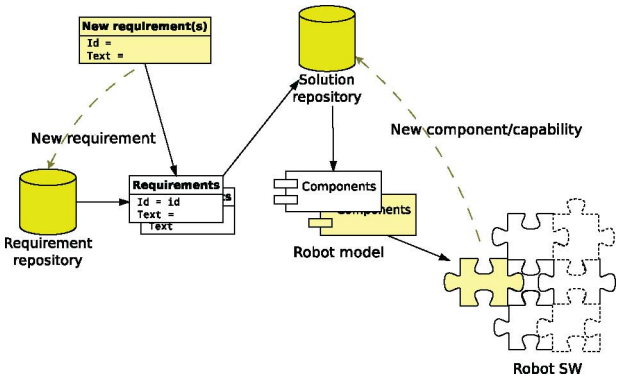

Figure 4. Engineering process

main Engineering. The results of the Domain Engineering are used as building blocks for the development of the safe robot software. This sub-process of application engineering is called Robot Engineering.

The Domain Engineering can be characterized by the following steps:

1. Perform hazard estimation and mitigation, make sure that the model of corresponding safety cases and requirements are formalized as described in the section 3.1. Here the modeled standard ISO/DIS 13482 is used as a base. Also applicable domain relevant repositories extending the safety case coverage are included.

2. Provide the component model according to the safe multilayer robot architecture as described in the section 3.2. The model representation of the component includes the component domain and specific component type (functionality) in that domain. The direct and indirect safety relevant properties shall be considered:

   (a) types of platforms supported for the component to be deployed on;

   (b) communication links, published and subscribed topics, services, data flow;

   (c) software performance level;

   (d) real-time properties;

   (e) dependencies on other software components.

3. Use requirements and modeled artifacts from the previous steps to develop the component. The component is implemented as a ROS node. The internal component design, choice of software paradigm and language is also justified by the requirements from previous steps.

4. Provide complete traceability from the built component to the safety case, as specified in section 3.2.

5. Provide test and verification for the built component. The basic tests are suggested by the standard ISO/DIS 13482. ROS allows to perform some tests in a fully-simulated environment. The performance/throughput/consumption tests have to be deployed on the real HW. The infrastructure for that tests can be provided by ROS in a "component-in-the-loop" scenario, where dependencies of the component are satisfied by the additional test environment. In general, component testing is similar to the advanced unit test which is common in software engineering. The formal verification can be limited by the complexity of the component. However, due to the software architecture higher reliability is expected from lower layer, and thus most likely simpler, components.

The Domain Engineering relates the software component to the corresponding safety case. The function delivered by the component determines the domain the component belongs to. Developed artifacts are stored in the corresponding repositories for reuse. In a safety case those are: new hazards, new risks, new safeguards and new verification methods. In case of the software repositories the artifacts are SW components.

The developed software artifacts are reusable components. Additionally to the modeled interfaces, reflecting the component functionality, the component parameters and the safety critical characteristics of the components are also reflected in the component representation, making it safe to be reused in the software model.

The Robot Engineering sub-process contains the following steps:

1. Create a component model of the robot software.
2. Check whether all safety requirements are covered with corresponding software components.
3. If required, go back to the system design to make the robot inherently safe.
4. Model the robot SW:
   (a) deployment-relevant issues; that includes the type of platforms the components are deployed on and the type of communication links between them;
   (b) resource utilization; that includes memory, processing unit, communication links;
   (c) real-time characteristics;
   (d) function safety coverage (based on the component dependencies);
   (e) avoid dependency of lower layer functions from higher layer function.
5. Build robot SW from components. The ROS launch files (XML) and configuration and parametrization files for the ROS packages can be generated from the software component model.
6. Perform tests from both robot and application domains. If required, repeat Domain Engineering sub-process for the effected components.

The engineering process supports designer in creating the domain (robot) model and checking it against the safety standard. This allows to see how the components of the system are affected in terms of safety relevant requirements. The overall safety characteristics of robot can be estimated already at the modeling stage. The resulting software is supported by safety evidence according to the relevant safety norm or an additional safety guideline.

After the development and disposal, the robot has to be maintained. Maintenance in general comprises the iterative repetition of an engineering process. In case of a safety failure maintenance can be seen as a repetition of the testing phase. If a failure is detected, the situation leading to it will be transferred to the testing phase and the iteration of the test step from the Robot Engineering sub-process is repeated. If a new hazard is identified, the Domain Engineering sub-process shall be repeated accordingly.

Basic safety requirements, relevant for autonomous service robots, are already addressed in the ISO/DIS 13482. This standard contains the list of typical hazards and provides some basic safeguards and verification approaches. The list can also be extended to a bigger requirement repository with well known methods like failure mode and effects analysis or hazard and risk analysis.

ROS provides advanced tool for the simulation and data flow logging. This allows to perform sophisticated tests in simulation and makes the reproduction of real tests scenarios easier.

4 Experimental proof of concept

To illustrate the approach described above we will consider a simple but comprehensive example. A mobile assembly assistant robot may be used to support human worker during assembly work inside of the car (see Fig. 5). The robot is semi-mobile, it enters the working area (either on its own or supported by some other robot) and performs its tasks. The safety issues are critical for that use case, especially considering the physical human-robot interaction in the working space which is common for both robot and human worker.

Our test case is an assembly assistant robot with the lightweight manipulator, like one shown in Fig. 6.

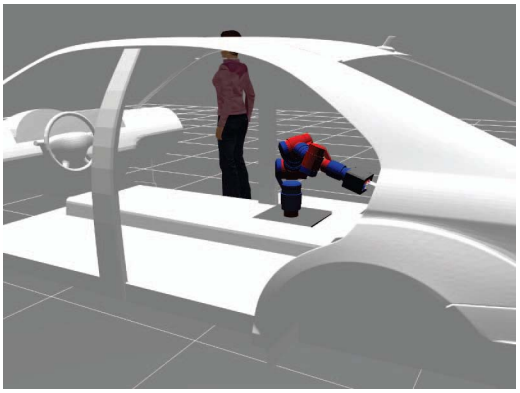

Figure 5. Mobile assembly assistant robot

The control of the robot is done by ROS running on the industrial PC.

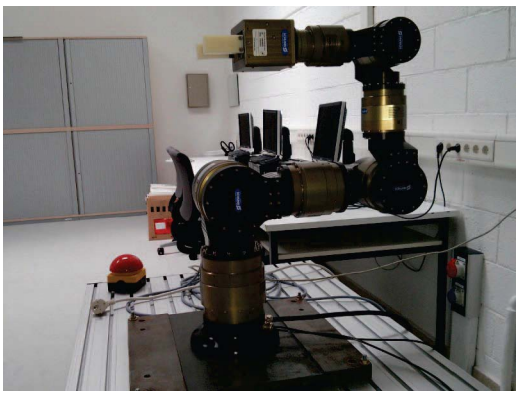

Figure 6. Schunk LWA

The robot engineering process starts with the identification of domains, relevant for the robot. We will skip the application domain, which depends on the task of robot and the tools it is carrying (gripper, drill etc). We will focus on the robot relevant domain. This is robot motion. According to our use case, robot is brought to the working space and performs its task by moving the manipulator. So we will focus here on the domain of robot manipulator movement.

According to the standard ISO/DIS 13482, relevant hazard is a hazardous physical contact during the human-robot interaction. The risk of a collision is high. The requirements for the protective measures on that hazards are shown in Fig. 7.

We pick the requirement for the tactile sensors. To fulfill that requirements, a software component model is designed. As a collision detector we could take artificial sensitive skin, which is covering the arm and is able to sense a touching during the collision.

The component "collision_detector" is directly traceable to the requirement for the tactile sensor. The same is valid for the "emcy_stop" component due to dependence on the component "collision_detector". No specific functional safety requirements or runtime

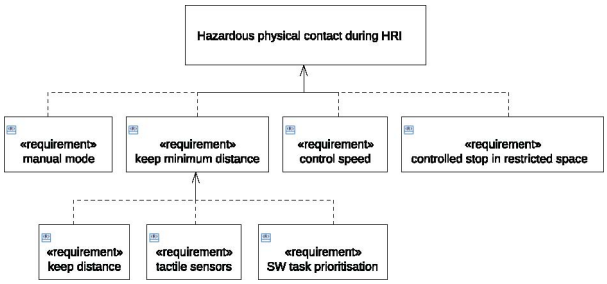

Figure 7. Requirement for mitigation of the risk on the "Hazardous physical contact during HRI"

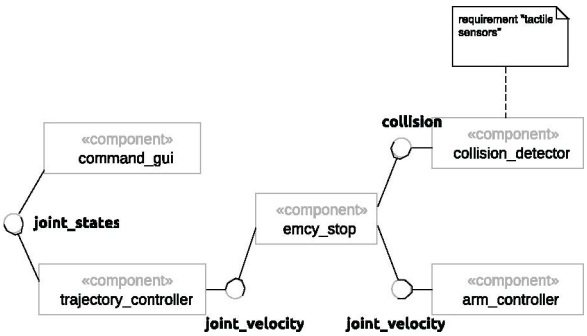

Figure 8. Software model

requirements are put on the components during the first step.

The component model is used to create ROS configuration and launch files, which describes the deployment of the ROS nodes.

Suitable experimental tests then finally proof the engulfment of the safety requirements. Standard ISO/DIS 13482 suggests code review and practical tests.

After reevaluating of the risks we may found out that further protective measures are required. If the collision detection have to be more robust and safe, the correspondent SW components shall run on the more reliable hardware and the software shall fulfill some performance level. These clearly are the new requirements on the SW component. If the component that fulfills that requirements is not available, corresponding Domain Engineering process for the development of the new component shall be started.

The further development of the system may take into account new safeguards, like extending safe pHRI from obstacle recognition to the obstacle avoidance. That will require new sensing capabilities (visual) and corresponding algorithms to perform online path replanning for the avoiding of sensed obstacles.

5 Conclusion

In the paper we have discussed how to deal with the safety case and chain of evidence in the domain of service robots, which architecture is suitable for the safe robot SW, which issues are relevant for engineering of the safe autonomous robots.

We have proposed a model based approach to the safety case on the example of the ISO/DIS 13482, the component-based multi-level safe robot software architecture, the interconnected repositories for store and reuse of the safety case models and software components, the two-stage safety-oriented engineering process for the safe autonomous robots.

Next steps to do are to complete model of the standard ISO/DIS 13482, to build taxonomy and models of the relevant existing ROS components (reverse modeling) and to extend the use case.

References

[1] sig/Embedded - ROS wiki.

[2] sig/Multimaster - ROS wiki.

[3] ISO/DIS 13482 robots and robotic devices – safety requirements for non-industrial robots – non-medical personal care robot, 2011.

[4] D. Alonso, C. Chicote-Vicente, O. Francisco, J. Pastor, and B. Álvarez. V^{3}CMM: a 3-view component meta-model for model-driven robotic software development. 1, 2010.

[5] A. Brooks, T. Kaupp, A. Makarenko, S. Williams, and A. Orebäck. Orca: A component model and repository. In D. Brugali, editor, *Software Engineering for Experimental Robotics*, volume 30, pages 231–251. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[6] H. Bruyninckx, P. Soetens, and B. Koninckx. The real-time motion control core of the orocos project. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 2766 – 2771 vol.2, Sept. 2003.

[7] DIN EN ISO 10218-1. Robots for industrial environments – safety requirements – part 1: Robot. Technical report, DIN German Institute for Standardization, Berlin, July 2009.

[8] H. Giese, M. Tichy, and D. Schilling. Compositional hazard analysis of UML component and deployment models. In M. Heisel, P. Liggesmeyer, and S. Wittmann, editors, *Computer Safety, Reliability, and Security*, volume 3219 of *Lecture Notes in Computer Science*, pages 166–179. Springer Berlin / Heidelberg, 2004.

[9] I. Habli. *Model-Based Assurance of Safety-Critical Product Lines.* PhD thesis, Department of Computer Science ,University of York, 2009.

[10] C. Harper and G. Virk. Towards the development of international safety standards for a human robot interaction. *International Journal of Social Robotics*, 2(3):229–234, June 2010.

[11] T. P. Kelly. *Arguing Safety – A Systematic Approach to Managing Safety Cases.* PhD thesis, 1998.

[12] G. K. Kraetzchmar, A. Shakhimardiv, J. Paulus, N. Hochgeschwender, and M. Reckhaus. Best Practice in Robotics. Deliverable D-2.2: Specifications of Architectures, Modules, Modularity, and Interfaces for the BROCTE Software Platform and Robot Control Architecture Workbench, 2010.

[13] R. Lutz. Extending the product family approach to support safe reuse. *JOURNAL OF SYSTEMS AND SOFTWARE*, 53:207–217, 2000.

[14] R. Lutz. Software engineering for safety: a roadmap. *Proceedings of the Conference on The Future of Software Engineering*, pages 213–226, 2000. ACM ID: 336556.

[15] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source robot operating system. 2009.

[16] I. A. D. Nesnas. The CLARAty project: Coping with hardware and software heterogeneity. In D. Brugali, editor, *Software Engineering for Experimental Robotics*, volume 30, pages 31–70. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[17] OMG. Robotic interaction service (RoIS). Technical Report Version 1.0, OMG, Feb. 2013.

[18] R. Panesar-Walawege, M. Sabetzadeh, and L. Briand. Using Model-Driven engineering for managing safety evidence: Challenges, vision and experience. In *Software Certification (WoSoCER), 2011 First International Workshop on*, pages 7 –12, Dec. 2011.

[19] R. K. Panesar-Walawege, M. Sabetzadeh, and L. Briand. Supporting the verification of compliance to safety standards via model-driven engineering: Approach, tool-support and empirical validation. *Information and Software Technology*, 55(5):836–864, May 2013.

[20] C. Schlegel, A. Steck, D. Brugali, and A. Knoll. Design abstraction and processes in robotics: From Code-Driven to Model-Driven engineering. In N. Ando, S. Balakirsky, T. Hemker, M. Reggiani, and O. Stryk, editors, *Simulation, Modeling, and Programming for Autonomous Robots*, volume 6472, pages 324–335. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[21] B. Siciliano and O. Khatib. *Springer Handbook of Robotics.* Springer, 1 edition, June 2008.

[22] K. Thramboulidis and S. Scholz. Integrating the 3+1 SysML view model with safety engineering. In *2010 IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, Sept. 2010.