Model Checking for Robotic Guided Surgery

H. Mönnich, J. Raczkowsky, and H. Wörn

IPR Karlsruhe, University of Karlsruhe, Karlsruhe, Germany
{moennich,rkowsky,wörn}@ira.uka.de

Abstract. This paper describes a model checking approach for robotic guided surgical interventions. The execution plan is modeled with a workflow editor as a petri net. The net is then analyzed for correct structure and syntax with XMLSchema. Petri nets allow checking for specific constraints, like soundness. Still the possibility to prove the net with runtime variables is missing. For this reason model checking is introduced to the architecture. The Petri-Net is transformed to the Model Checking language of NuSMV2, an open source model checking tool. Conditions are modeled with temporal logic and these specifications are proved with the model checker. This results in the possibility to prove the correct initialization of hardware devices and to find possible runtime errors. The workflow editor and model checking capabilities are developed for a demonstrator consisting of a KUKA lightweight robot, a laser distance sensor and ART tracking for CO2 laser ablation on bone.

Keywords: Workflow Systems, service oriented architecture, CORBA, knowledge management, temporal logic.

1 Introduction

One important aspect of the AccuRobAs project is to provide a modular system architecture for robot assisted surgical interventions. This goal of a modular system is hard to reach combined with the requirements of surgical applications like safety and hard realtime environments. The realtime problem has to be solved with special protocols and software environments. The safety problem is more complex and cannot be solved in general. The reason is the still unsolved problem of understanding a language and the underlying semantic as pointed out by[1][2]. But different approaches to solve this problem partially have been developed. One example is model checking that is successfully used for circuit design. Model checking is the process of checking whether a given structure is a model of a given logical formula. The model is often simplified. The concept is general and applies to all kinds of logics and suitable structures. A simple model-checking problem is testing whether a given formula in the propositional logic is satisfied by a given structure. The main drawback of model checkers is the state explosion problem that is heavily limiting the usage of these tools. Model checking was already the interest of the NASA project PLEXIL[3]. Plexil offers a special programming language for execution plans. These execution plans can then be proved by a model checker. PLEXIL uses the LTSA model checker [4]. This already showed the successful usage of a model checking tool for a planning

P. Kostkova (Ed.): eHealth 2009, LNICST 27, pp. 1–4, 2010.

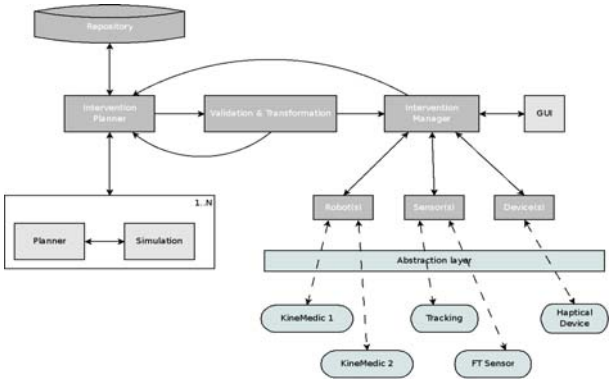

Fig. 1. System Architecture

language. The main drawback is the usage of a special planning language that is not intuitive for the user. Workflows are more common to field of medical applications. A workflow formalism that is well proven is the petri net workflow as described by van der Aalst et al.[5]. A petri net has formal semantics with a clear and precise definition. Several enhancements for classical petri nets have been developed and defined formally like colored, timed or hierarchical petri nets. Twenty different workflow patterns were discovered for workflow nets that are all supported by petri nets, even the more rarely used implicit or operation [6]. For this reason they were chosen for the AccuRobAs project for the surgical application field. In figure 1 the system architecture is shown. In this paper the Validation and Transformation step is described.

2 Transformation and Validation

The validation and transformation step is important to ensure the correct behaviour of the plan. Validation can be divided into two steps. The first step is to check the syntax and the corresponding structure of a plan. This part is well known from constructing compilers in computer science and can be easily adapted to the plans with XMLSchema. The second step is more challenging. To ensure the correct behavior of programs a model checker is used to solve this problem partially. The last step is the transformation into a state machine for real-time execution. The workflow plan is transferred into a state machine that is executed by the Intervention Manager, currently an implementation based on SCXML Apache.

2.1 Model Checking

The software NuSMV [7] is used here as a model checker. NuSMV is an updated version of the SMV symbolic model checker. NuSMV has been developed as a joint project between ITC-IRST (Istituto Trentino di Cultura, Istituto per la Ricerca Scientifica e Tecnologica in Trento, Italy), Carnegie Mellon University, the University of Genoa and the University of Trento. It is a reimplementation and extension of SMV, the first model checker based on Binary Decision Diagrams (BDDs).

The transformation step from a petri net to the model checker NuSMV is the following: 1.) the workflow is constructed inside the yawl workflow editor with all conditions 2.) the yawl workflow is exported to an xml file 3.) the transformation from the xml petri net to the NuSMV language as an text file is done with an xslt stylesheet 4.) NuSMV2 is evaluating the specifications. A similar transformation has been described by Schaad and Sohr[8].

The transformation itself is performed in the following way: Every task and condition is modeled as a variable inside the NuSMV2 language. The variable has two possible values: ”wait” and ”run”. They indicate the current status of the state.

3 Model Checking Examples

Figure 2 is shows a simple example workflow. Here a robotic device, called LBR, is used to move to a specific starting position and to measure the distance between TCP point and the target with a laser distance sensor. Under all circumstances it should be possible to reach a safe state and to move the robot away from the patient. In this case the following specifications could be checked: 1.) Is the state “safePos” reachable, 2.) the state “safePos” should never be activated if the condition is not true and 3.) if the safePos is desired, the next step is always the safe state, excepting the two cases for the start and end state.

Reachability: `SPEC EF (safePos_8 = runs)`
Safety: `SPEC AG (!goToSafePos -> AX safePos_8 != runs)`
Liveness: `SPEC AG (goToSafePos & InputCondition_1 != runs & Output-Condition_2 != runs) -> AX safePos_8 = runs`

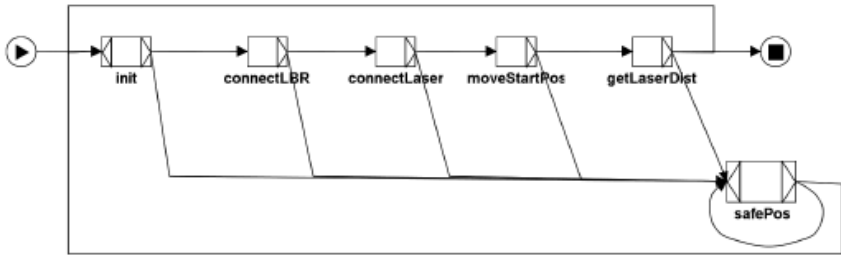

Fig. 2. Simple example workflow

4 Results

In this work an approach for model checking for surgical interventions is presented. A method is described how to prove constraints inside the workflow. The main problem however remains, it can only be shown that a workflow satisfies a given specification or not. It can be shown that a workflow is not violating against a specification but not that the workflow behaves as expected.

5 Future Works

The current conversion from petri net to the NuSMV2 model checker is still very limited and should be compared to other methods. Even the specifications must be inserted by the user after the conversion process. This should be done automatically to show the full potential of this approach. It would be helpful to develop a set of typical rules that must be satisfied for any kind of surgical procedure.

Acknowledgement

This work has been funded by the European Commission's Sixth Framework Program within the project Accurate Robot Assistant - ACCUROBAS under grant no. 045201.

The author would like to thank Henrik Keller who implemented the complete transformation and developed the example during his student research project [9] at IPR Karlsruhe.

References

1. Wittgenstein, L.: Philosophische Untersuchungen, Herausgegeben von Joachim Schulte. Wissenschaftliche Buchgesellschaft. Frankfurt am Main (2001)
2. Searle, J.R.: Minds, brains, and programs. Behavioral and Brain Sciences 3 (1980)
3. Siminiceanu, R.: Model Checking Abstract PLEXIL Programs with SMART. NASA CR-2007-214542 (April 2007)
4. Magee, J., Kramer, J.: Concurrency. In: State Models and Java Programs.: State Models and Java Programs. Wiley & Sons, Chichester (1999)
5. ter Hofstede, A.H., van der Aalst, W.M.: YAWL: yet another workflow language. Information Systems 30(4), 245–275 (2005)
6. Russell, N., ter Hofstede, A.H.M., van der Aalst, W.M.P., Mulyar, N.: Workflow Control-Flow Patterns. A Revised View
7. Cimatti, E.M., Clarke, E., Giunchiglia, F., Giunchiglia, M., Pistore, M., Roveri, R.: NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, p. 359. Springer, Heidelberg (2002)
8. Schaad, A., Sohr, K.: A workflow instance-based model-checking approach to analysing organisational controls in a loan origination process. In: 1st International Workshop on Secure Information Systems (SIS 2006), Wisla, Polen (2006)
9. Keller, H.: Erhöhung der Sicherheit von Robotersteuerungen in der Medizin durch Model Checking, University of Karlsruhe (2008)