Robotics and Autonomous Systems 60 (2012) 1563–1578

Contents lists available at SciVerse ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Rigorous design of robot software: A formal component-based approach☆,☆☆

Tesnim Abdellatif[a], Saddek Bensalem[a], Jacques Combaz[a], Lavindra de Silva[b], Felix Ingrand[b,*]

[a] Verimag/CNRS, Grenoble I Uni., France

[b] LAAS/CNRS, Toulouse Uni., France

ARTICLE INFO

Article history:
Received 21 November 2011
Received in revised form
7 September 2012
Accepted 14 September 2012
Available online 23 September 2012

Keywords:
Robotic software architecture
Controller synthesis
Verification and validation
Robotic functional layer
Robust software

ABSTRACT

We have recently started an effort to combine a state of the art tool for developing functional modules of robotic systems (GenoM) with a component based framework for implementing embedded real-time systems (BIP). Unlike some works which study the connection between formal approaches and the highest (decisional) level of the robot software architecture, where deliberative activities such as planning, diagnostics, and execution control are conducted, we tackle the problem of using formal methods for developing modules of the functional level of robots. Little attention has been drawn to the development of these modules whose robustness is paramount to the robustness of the overall platform.

To this end, we have successfully developed the GenoM/BIP component based design approach and applied it to the functional level of a complex exploration rover. Here, we report on this work, and show how we: *(i)* produce a very fine grained formal computational model of the robot functional level; *(ii)* run the BIP engine on the real robot, which executes and enforces the model semantics at runtime; and *(iii)* check the model offline for deadlock-freedom, as well as other safety properties.

Moreover, we also extended this paradigm in a number of promising directions: *(i)* introduced a real-time BIP engine which can now use and control a timed BIP model; *(ii)* distributed the model and the engine over multiple CPUs; *(iii)* proposed a user-friendly language for specifying constraints on the model; and *(iv)* linked the model with a temporal plan execution controller. Interestingly, although our approach was initially proposed for the lowest level of robot architectures, these more recent extensions now allow us to model and manage the deliberation taking place at the decisional layer.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

For the large scale deployment of robots in places such as homes, shopping centers and hospitals, where there is close and regular interaction with humans, robot software integrators and developers may soon need to provide guarantees and formal proofs to certification bodies that their robots are safe, dependable, and behave correctly. This also applies to robots such as extraterrestrial rovers, used in expensive and distant missions, which need to avoid equipment damage and mission failure. Such guarantees may involve proofs that a rover will not move while it is communicating or even worse, while it is drilling, that the navigation software has no fatal deadlock, or that a service robot will not extend its arm dangerously while navigating or will not open its gripper while holding a breakable object.

The most common method to ensure the correctness of a system is testing (see [1] for a survey). Testing techniques have been effective for finding bugs in many industrial applications. Unfortunately, there is, in general, no way for a finite set of test cases to cover all possible scenarios, and therefore, bugs may remain undetected. Hence, in general, testing does not give any guarantees on the correctness of the entire system. Consequently, these approaches are impractical with complex autonomous and embedded systems for even a small fraction of the total operating space.

We have successfully proposed a novel software engineering methodology for developing safe and dependable robotic systems [2,3]. With our approach, one can provide guarantees that the robot will not perform actions that may lead to situations deemed unsafe, i.e., those that may eventuate in undesired or catastrophic consequences. Our approach (Section 3) relies on the integration of two existing state-of-the-art methods, namely, (i) the GenoM tool of the LAAS architecture [4], used for specifying and implementing the lowest level of robotic systems, and (ii) the BIP software framework for formally modeling and verifying complex, real-time component-based systems [5]. In this paper, we extend this approach to be used on complex robotic systems for designing both the decisional and functional levels. We first present (in Section 4)

☆ The authors are in alphabetical order by last name.

☆☆ Part of this work is funded by the ESA/ESTEC GOAC project and by the FNRAE MARAE project. We thank Rongjie Yan for some useful discussions.

* Corresponding author. Tel.: +33 549056432.
E-mail addresses: tesnim.abdellatif@imag.fr (T. Abdellatif), saddek.bensalem@imag.fr (S. Bensalem), jacques.combaz@imag.fr (J. Combaz), ldesilva@laas.fr (L. de Silva), felix@laas.fr (F. Ingrand).

0921-8890/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2012.09.005

a high-level language that allows roboticists to easily express specific constraints on the system. In light of early experimental findings, we also provide insights into more recent work focused on real-time features of a system. Indeed, we developed a real-time version of BIP (Section 5), which takes into account execution time and deadlines. We also used the real-time BIP engine as a temporal-plan execution controller (Section 6). Section 7 presents results on all of the above work. We then conclude the paper with a future work section (Section 8), presenting a multi-CPU distributed version of BIP which allows us to run it on modern robotic platforms, and how we plan to use GenoM3 to extend our approach toward the ROS ecosystem, and a discussion section (Section 9).

2. State of the art in building robot software using formal methods

Despite a growing concern to develop safe, robust, and verifiable robotic systems, overall, robot software development remains quite disconnected from the use of formal methods. Moreover, the extent to which formal methods is used varies different between the functional level and the decisional level of robot software architectures.

2.1. The decisional level design

Formal methods have been more widely used together with "decisional components" of robotic systems. The main reason is perhaps because these decisional components already use a "model" (for planning, diagnostics, etc.). In [6], the authors propose a system relying on a model-based approach. The objective is to abstract the system into a state transitions based language modeling the dependability concerns. The programmers specify state evolutions with invariants and a controller executes this maintaining these invariants. To do that, the controller estimates the most likely current state – using observation and a probabilistic model of physical components – and finds the most reliable sequence of commands to reach a specified goal (i.e., with a minimum probability of failure). In [7], the authors present the CIRCA SSP planner for hard real-time controllers. This planner synthesizes off-line controllers from a domain description (preconditions, postconditions and deadlines of tasks). CIRCA SSP can then deduce the corresponding timed automaton to control the system on-line, with respect to these constraints. This automaton can be formally validated with model checking techniques. Similarly, [8] discusses an approach for model checking the AgentSpeak(L) agent programming language aimed at reactive planning systems. The work describes a toolkit called CASP (Checking AgentSpeak Programs) for supporting the use of model checking techniques, in particular, for automatically translating AgentSpeak(L) programs into a language understood by a model checker. In [9], the authors present a system that allows the translation from MPL (Model-based Processing Language) and TDL (Task Description Language) – the executive language of the CLARAty architecture[10] – to SMV, a symbolic model checker language.

In [11], the authors discuss an approach for automatically generating correct-by-construction robot controllers from high-level representations of tasks given in Structured English, which are translated into a subset of Linear Temporal Logic and eventually into automata. In their work, complex and continuous missions can be specified using the basic prepositions 'between', 'near', 'within', 'inside', and 'outside'. An example of such a mission is "stay near A unless the alarm is sounding", where A is a location. Likewise, [12] also deals with the synthesis of correct-by-construction controllers based on temporal logic specifications. Here, finite state automata based controllers are synthesized by a trajectory planner to satisfy a given temporal specification, which is based on an abstract model of the physical system. The authors show how the correct behavior of an autonomous vehicle can be maintained using the robot controller automatically synthesized.

2.2. The functional level design

On the functional side of robotic systems, the situation is quite different. There are many popular software tools available (e.g., OROCOS [13], CARMEN [14], Player Stage [15], Microsoft Robotics Studio [16], and ROS [17]) to develop the functional level of robotic systems. There are even some works which compare them, e.g., [18,19]. Yet, none of these architectural tools and frameworks proposes any extension or link with formal methods, and validation or verification tools.

Recently, we proposed the R^{2}C [20], a tool used between the functional and decisional levels of a robotic system. The main component of R^{2}C is the *state checker*. This component encodes the constraints of the system, specified in a language named ExoGen. At run-time it continuously checks if new requests are consistent with the current execution state and the model of properties to enforce. Another interesting early approach to prove various formal properties of the functional level of robotic systems is the ORCCAD system [21]. This development environment, based on the Esterel [22] language, provides extensions to specify robot "tasks" and "procedures". However, this approach remains constrained by the synchronous systems paradigm.

More generally, as advocated in [23], an important trend in modern systems engineering is model-based design, which relies on the use of explicit models to describe development activities and their products. It aims at bridging the gap between application software and its implementation by allowing predictability and guidance through analysis of global models of the system under development. The first model-based approaches, such as those based on ADA, synchronous languages [24] and Matlab/Simulink, support very specific notions of components and composition. More recently, modeling languages, such as UML [25] and AADL [26], attempt to be more generic. They support notions of components that are independent from a particular programming language, and put emphasis on system architecture as a means to organize computation, communication, and implementation constraints. Software and system component-based techniques have not yet achieved a satisfactory level of maturity. Systems built by assembling together independently developed and delivered components often exhibit pathological behavior. Part of the problem is that developers of these systems do not have a precise way of expressing the behavior of components at their interfaces, where inconsistencies may occur. Components may be developed at different times and by different developers with, possibly, different uses in mind. Their different internal assumptions, when exposed to concurrent execution, can give rise to unexpected behavior, e.g., race conditions, and deadlocks.

All these difficulties and weaknesses are amplified in embedded robotic systems design in general. They cannot be overcome, unless we solve the hard fundamental problems concerning the definition of rigorous frameworks for component-based design.

3. Our approach

In past work we proposed an approach [27,3] to develop safe and dependable functional levels of complex, real-world robots, which relied on the integration of two state-of-the-art technologies, namely: (i) GenoM [4] – a tool (part of the LAAS architecture toolbox) that is used for specifying and implementing the functional level of robots; and (ii) BIP [5] – a software framework for formally modeling complex, real-time component-based systems, with supporting tool-sets for verifying such systems.

Download English Version:

https://daneshyari.com/en/article/411830

Download Persian Version:

https://daneshyari.com/article/411830

Daneshyari.com