UML Framework for the Design of Real-Time Robot Controllers

L. Carroll, B. Tondu, C. Baron, and J.C. Geffroy

LESIA - INSA, Complexe Scientifique de Rangueil
31077 Toulouse Cedex 4, France
Luis.Carroll@insa-tlse.fr

Abstract. New types of robotic applications, where tasks are accomplished in cooperation with humans and under unstructured environments, have led researchers to consider dependability requirements as fundamental design criteria. Particularly, robot controllers must include additional features allowing to cope with system failures in order to improve reliability and safety. In this article, we suggest the use of the UML within a development method including dependability means, for the design of real-time robot controllers.

1 Introduction

Robot controllers integrate the set of high level processes (robot programming languages and task elementary motion planning) and low-level processes (trajectory generation and joint control). Specific properties coming from electronic and mechanical components and specific properties from the application environment lead industry to develop dedicated controllers without generic considerations (flexibility according to upgrade evolutions or environment perturbations, etc.). In addition, the growing complexity of robot functionalities (integration of external data, management of redundant joints, position and force control, etc.) implies the mastering of the design of evolutionary systems. Moreover, in new types of applications, robots have to cooperate with human operators and to share an unstructured workspace. Therefore, safety performances are required.

2 Robotic Systems Requirements

We consider a robotic system as composed of a robot and its controller, and the service it delivers as the completion of a task. The controller provides the manipulator with the intelligence to perform tasks as described by its user. These tasks generally consist of a set of actions intended to position the robot's end-effector for the manipulation of objects. A failure occurs when the robot deviates from the desired trajectory. This failure can be the result of changes in the robot kinematic parameters or problems in the system software or hardware. Failures resulting from changes in robot parameters can be avoided by implementing highly robust control laws. Failures due to hardware or software call for a particular analysis on the critical components of the system. Recent research has been done on different aspects of dependability in the field of robotics [1], [2].

The basic robotic systems requirements deal with the final static and dynamic performances and with dependability criteria: *availability*, *reliability*, *maintainability* and *safety*. Particularly, robot safety is concerned with preventing the occurrence of damage to the robot itself and preventing the robot from damaging its environment, e.g., human operators [3]. Rules and guidelines of safety standards and a selective use of engineering technologies may be used in order to guarantee a safe robot behavior according to its environment. Besides, we identify two other groups of criteria associated with the robustness of the software architecture (reusability, modifiability and traceability) and the quality of the development process (verifiability, understandability, development ease and expression capability).

3 Means for Dependability

A fault can cause the system to get into an error state. This error state will lead the system into an undesirable behavior considered as a failure. Faults can be introduced during the development of the system or they can occur during the useful lifetime. In both cases, we must identify the possible techniques to cope with them, and the system components potentially involved in its origins.

Concerning development faults, fault avoidance and fault removal methods must be used during the specification, design and implementation of the controller. These techniques reduce the faults introduced during the system development by humans and the tools they use. Concerning faults occurring during the useful lifetime, we must implement a real-time controller able to cope with system failures. This controller acquires information data of the current system state from the robot sensors or other sensors located in its workspace and from the controller itself. The different software modules inside the controller make the robot react in the desired way. It is then suitable to include on-line testing for fault detection, aimed at detecting faults not only from sensors or actuators, but also from the controller itself.

In order to develop a dependable robot controller, we consider that the use of a real-time system development method must be associated to a dependable methodology including fault prevention, fault removal and fault tolerance processes as proposed by [4]. This methodology deals with the avoidance and removal of faults during the development phase, and also enables the introduction of fault tolerant routines inside the normal life cycle of the controller.

4 A Real-Time Robot Controller

We have chosen the UML [5] for the modeling of a real-time robot controller. The comparative study we have presented in [6] highlights the qualities of the UML for the design of the controller. Besides its description and representation

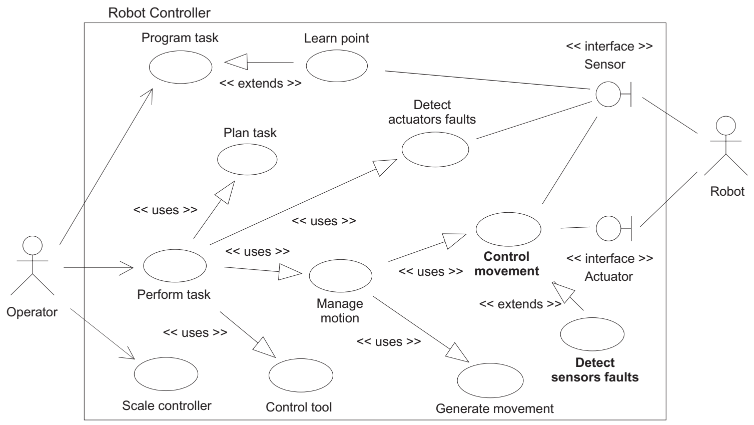

Fig. 1. Robot controller Use Cases Diagram.

capabilities, it offers large possibilities for the improvement of dependability. Particularly, Use Case Diagrams are well dedicated to the capture of requirements [7]. They drive the designer during the development process and assist the functional testing of the software. Figure 1 shows the Use Cases of the robot controller considered (each one represented by an ellipse). Use Cases may stand on their own, they may use capabilities specified in other Use Cases and they may be extended by other Use Cases. These relations (”uses” and ”extends”) are represented by arrows. Actors (operator and robot) are people or things that interact with the system (robot controller). They may communicate with Use Cases by the way of interfaces (robot sensor and actuators).

A scenario, generally represented by a Sequence Diagram where objects interact for the accomplishment of the requirement, expresses each Use Case. Figure 2 shows the interaction between different objects performing the ”Control movement” Use Case. This Use Case is extended by the ”Detect sensor faults” Use Case as represented in figure 1. This scenario is particularly interesting because it shows on-line detection of sensor faults within the sampling period of the control loop. The use of functionally redundant sensors allows the Fault detector objet to verify data and communicate faults to the Task Planner object.

These UML visual elements are well adapted to capture dependability requirements. For example, Sequence Diagrams facilitate functional test and help to represent temporal constraints and behavior of objects. This feature is useful during the implementation stage in RTOS.

5 Conclusions and Future Work

The evolution of robot systems leads to new types of applications where human beings can actively interact with robots. Dependability must be considered as an important design criterion, providing people with safety inside the robot workspace. The choice of a development process, from the initial specification to

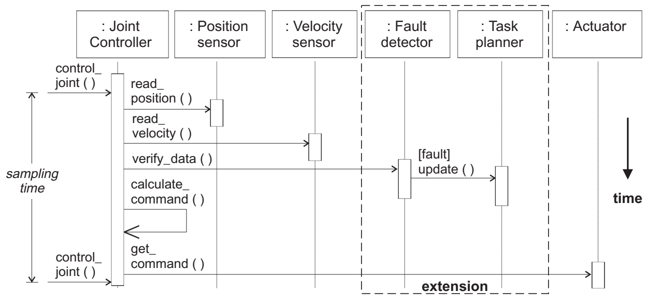

Fig. 2. Sequence diagram for the control of joints including sensors fault detection.

the final robot controller implementation, is important in relation to the functionality and dependability of the whole system. Fault avoidance, fault removal and fault tolerance techniques complement this process during all the system's life cycle.

We have developed a robot controller based on a VxWorks RTOS using the proposed method for the control of a 2 DOF SCARA-type robot. The use of the UML resulted to be appropriated for the controller development and maintainability. It is important to mention that adaptive and perfective maintenance becomes easier for the removal of specification faults and improvement and test of different detection and tolerance routines. This allows the implementation of the different techniques proposed by other authors in the field of fault tolerant robotic systems. Anyhow, it must be completed with formal methods or simulation tools aimed at the verification and validation of the system.

References

[1] Visinsky M.L., Cavallaro J.R., Walker I.D.: Robotic Fault Detection and Fault Tolerance: a Survey, Int. J. Reliability Eng. and System Safety, Vol. 46 (1994) 139-158
[2] Tso K.S., Hecht M., Marzwell N.: Fault-Tolerant Robotic System for Critical Applications, IEEE Int. Conf. Robotics and Automation, Atlanta, GA, USA (1993) 691-696
[3] Dhillon B.S.: Robot Reliability and Safety, Springer-Verlag (1991)
[4] Laprie J.C.: Dependability of Computer Systems: Concepts, limits, improvements, IEEE Int. Workshop CAD, Test and Evaluation for Dependability, Beijing, China (1996) 23-33
[5] Booch G., Jacobson I., Rumbaugh J.: The Unified Modeling Language for Object-Oriented Development, Version 1.0, Rational Software Corporation (1997)
[6] Carroll L., Tondu B., Baron C., Geffroy J.C.: Comparison of Two Significant Development Methods applied to the Design of Real-time Robot Controllers, IEEE Int. Conf. Systems, Man and Cybernetics, LaJolla, USA (1998) 3394-3399
[7] Jacobson, I.: Object-Oriented Software Engineering: an use case driven approach, Addison-Wesley, New York (1992)