Chapter 8
Run-Time Compositional Software Platform for Autonomous NXT Robots

Ning Gui
University of Antwerp, Belgium

Vincenzo De Florio
University of Antwerp, Belgium

Chris Blondia
University of Antwerp, Belgium

ABSTRACT

Autonomous Robots normally perform tasks in unstructured environments, with little or no continuous human guidance. This calls for context-aware, self-adaptive software systems. This paper aims at providing a flexible adaptive middleware platform to seamlessly integrate multiple adaptation logics during the run-time. To support such an approach, a reconfigurable middleware system "ACCADA" was designed to provide compositional adaptation. During the run-time, context knowledge is used to select the most appropriate adaptation modules so as to compose an adaptive system best-matching the current exogenous and endogenous conditions. Together with a structure modeler, this allows robotic applications' structure to be autonomously (re)-constructed and (re)-configured. This paper applies this model on a Lego NXT robot system. A remote NXT model is designed to wrap and expose native NXT devices into service components that can be managed during the run-time. A dynamic UI is implemented which can be changed and customized according to system conditions. Results show that the framework changes robot adaptation behavior during the run-time.

DOI: 10.4018/978-1-4666-2056-8.ch008

INTRODUCTION

Autonomous robots can perform their intended tasks in unstructured environments without (or with minimal) human guidance. An autonomous robot may also learn or gain new capabilities like adjusting strategies for accomplishing its task(s) or adapting to changing surroundings. Such a high degree of autonomy is particularly desirable in fields such as space exploration, cleaning floors, mowing lawns, and waste water treatment.

A basic concept that is applied in autonomous robot control is the closed control loop. Each autonomic system consists of managed resources (controllable hardware or software components) and an autonomic manager for steering the underlying managed resources. Normally, this system includes automated methods to collect the details it needs from the system (Sensor); to analyze those details and determine if something needs to change (Analyzer); to create a plan, or sequence of actions, that specifies the necessary changes (Planner); and to perform those actions (Actuator). As we can see, in such system, Analyzer and Planner play a key role in such a control loop. These adaptation modules can greatly influence robot adaptation behavior. Several approaches have been proposed to provide a more flexible adaptation strategy – examples include (Hashimoto, Kojima, & Kubota, 2003), which uses evolutionary computation and fuzzy systems, or (Inamura, Inaba, & Inoue, 2000), using Bayesian networks. However, these works focused on designing certain adaptability algorithms for autonomous robots. Their control logics are statically linked and mingled with other system modules such as sensors and actuators. This static nature makes it very hard for autonomous robots to change their adaptation strategies. Most of these approaches can only effectively adapt within certain known environments or under certain predefined conditions.

A typical example is given by the Nasa Mars Rovers (Jet Propulsion Laboratory, n. d.). These robots, 170 to 320 million kilometers away from the Earth, are able to receive and send quite some information, either directly or via the Mars Orbiter (satellite around Mars). However, the downside of this communication is that their latency is very high (about 20 minutes), which obviously means that the robots cannot be remotely controlled. To perform their mission for NASA these robots had a software system that ran over a list of actions that were uploaded during the communication moment. A problem with this software is the static nature of its software platform. When the software needs an update to fix some problem or when some new features are needed to face unprecedented conditions, the robot software needs to be completely replaced.

The work described in this paper applies a new approach to implementing the adaptation loop in autonomous robot systems. In a nutshell, our architecture model realizes an adaptation loop which can be run-time revised so as to better match the current context. This strategy allows the application configuration to be modified outside of the application business logics. In order to deal with changing environments or/and robot status, our adaptation framework is designed to systematically support multiple adaptation logics. An adaptation plan is generated by run-time selected adaptation modules according to context to date. Robot applications, built from individual component instances, are composed and reconfigured by these run-time generated policies. This work is based on our ACCADA framework proposed in (Gui, De Florio, Sun, & Blondia, 2009b).

In order to seamlessly integrate the NXT robot into the ACCADA framework, a remote NXT model is designed to expose native NXT sensors and actuators as run-time manageable components. This remote NXT model allows future more advanced sensors/actuators to be easily plugged into our framework. Our modular middleware solution can effectively support adding/removing context-specific Planners during run-time, selecting the right adaptation Planner according to context by e.g. using a battery oriented adaptation scenario. Other adaptation strategies, such as fault-tolerant adaptation, can be injected into the system during