Mapping Mobile Statechart Diagrams to the π-Calculus using Graph Transformation: An Approach for Modeling, Simulation and Verification of Mobile Agent-based Software Systems

Aissam Belghiat, University Constantine 2 Abdelhamid Mehri, Constantine, Algeria
Allaoua Chaoui, University Constantine 2 Abdelhamid Mehri, Constantine, Algeria

ABSTRACT

Mobile UML (M-UML) has been proposed as an extension of UML to model mobile agent-based software systems. As UML, M-UML suffers from lack of formal semantics due to its semi-formal nature which penalize the verification of correct behavior of the modeled systems. This paper provides a graphical yet formal approach for the modeling, simulation and verification of mobile statechart diagrams using graph transformations in the AToM3 tool. The authors have firstly proposed meta-models for mobile statechart diagram and flowgraph. Then, a twofold graph grammar is developed for the automatic mapping of mobile statechart diagrams into flowgraphs and in the same time generates the corresponding π-calculus specification. This graph grammar enables either execution through simulation by flowgraphs or verification through model checking, using existing tools (e.g. the Mobility Workbench, MWB). An illustrative example of the authors’ approach is provided.

KEYWORDS

Mobile Agent-Based Software Systems, Mobile UML, Modeling, Simulation, Verification, π-Calculus

INTRODUCTION

Modeling and verification of mobile agent-based software systems is a difficult task. This is due to different constraints (mobility, security... etc.) which must be taken into account in these systems to build correct software (Hamidi & Mohammadi, 2008). Multiple attempts for providing a full expressive language for these systems are done in the literature such as Mobile UML (M-UML) (Saleh & El-Morr, 2004). M-UML is an extension of UML (OMG, 2015) to model mobile agent-based applications by covering the mobility feature that represents the novel concept inherent to such systems. It provides multiple views and diagrams including the mobile statechart diagram. This latter can be used easily to specify the lifecycle of different agents, but the problem of formal semantics missing from this language prohibits any attempts to apply rigorous automated verification or to execute these models.

Formal methods have been used largely to deal with such problems (Bouzahzah & Maamri, 2015; Kouah et al., 2016; Lin, 2007). They are adopted as target semantics formalisms. In our case, we need a formalism that is capable to model the mobility feature. Thus, we have chosen the π-calculus (Milner,
1999) which is a process algebra for modeling mobile and concurrent systems. It has a rich theory and tools and considered as the most adapted formalism for mobile agent systems. A difficulty with the \( \pi \)-calculus is its rigorous formal nature that has made it hard to be used for a wider audience. Therefore, providing a \( \pi \)-calculus based approach as in our case might be insufficient for stakeholders, although the advantages offered by the approach. The use of a graphical representation for this formalism must significantly facilitate its interpretation. So, a more promising way is to add a pseudo-simulator (without expensive costs) that explains graphically the execution of mobile statechart diagrams in the \( \pi \)-calculus on real time. This will give an easy comprehension of the dynamic evolution of the modeled systems. We have chosen the flowgraphs (Milner, 1979) for this purpose. By this way, we could obtain a framework for modeling, simulating and verifying mobile statechart diagrams.

To implement the ideas evoked above, we have used the AToM3 tool (AToM3, 2002) (De Lara & Vangheluwe, 2002b) to build our framework. Hence, we have firstly proposed two meta-models for mobile statecharts and flowgraphs to allow their visual modeling in the AToM3 tool. Then, we have developed a twofold graph grammar which simulates the execution of mobile statechart diagrams using flowgraphs and at the same time derives their corresponding \( \pi \)-calculus code. The generated specifications are then used to verify systems using \( \pi \)-calculus analytical tools such as the MWB tool (Victor & Moller, 1994) used here.

The rest of the paper is organized as follows. In Section 2, we present a background concerning M-UML, \( \pi \)-calculus and AToM3. In Section 3, we present the proposed approach to build the framework. In Section 4, an example is presented. In Section 5, some related works are invoked. Section 6 concludes the paper and gives some perspectives.

BACKGROUND

M-UML Statechart Diagrams (MSDs)

M-UML Statechart Diagrams (MSDs) (Saleh & El-Morr, 2004) extends UML statecharts (OMG, 2015) to model the behavior of mobile agents by exposing their different states. We present in Figure 1 structural elements of MSDs, and we focus on the new introduced elements to deal with mobility in statechart diagrams.

A platform is a physical node on which the agents of an application can run. The initial pseudo-state represents the execution starting point of an agent. The agent, at a given time, could be found in a distinct state at a particular platform. A normal state represents a simple state of the agent at its platform. A mobile state is a state reachable by the agent out of its base platform; it is designed by a box (M). A transition is a unidirectional link which emanating from one state (i.e. the source state) to another state (i.e. the target state). A transition is simple, if the both states at its two ends are situated on the same platform. A mobile transition is a transition between two states reachable by the agent while at two different platforms; it is designed by a box (M) and it has several forms as illustrated in Figure 1. All states reached by an agent either at its base or away will be depicted with a dashed box (M). A remote transition occurs if a mobile agent reaches a state while interacting with another agent remotely; it is designed by a box (R). A transition with \( \langle\text{agentreturn}\rangle \) stereotype represents the return of the mobile agent to its original platform by reaching a state in it. If the agent finishes the execution, it will attain the final pseudo-state.

\( \pi \)-Calculus

The \( \pi \)-calculus (Milner, 1993, 1999) is a process algebra that is used for specifying concurrent systems with mobile communication. The \( \pi \)-calculus uses two concepts to model mobile systems; a process (also called agent) which is an active communicating entity in the system, and a name which is anything else, e.g. a communication link, variable, data, etc. This formalism differs from other calculus in authorizing the passage of channels between processes and consequently augmenting
Related Content

Intelligent Transportation Systems: The State of the Art in Railways
[www.igi-global.com/chapter/intelligent-transportation-systems/205855?camid=4v1a](www.igi-global.com/chapter/intelligent-transportation-systems/205855?camid=4v1a)

Recognition of Chemical Entities using Pattern Matching and Functional Group Classification
Hierarchical Fuzzy Rule Interpolation and its Application for Hotels Location Selection
www.igi-global.com/chapter/hierarchical-fuzzy-rule-interpolation-and-its-application-for-hotels-location-selection/178388?camid=4v1a

Big Data Analytics With Machine Learning and Deep Learning Methods for Detection of Anomalies in Network Traffic