SPECIFYING TESTS FOR AD-HOC MOBILE SYSTEMS

Zoltán MICSKEI
Advisors: István MAJZIK (BUTE), Hélène WAESELYNCK (LAAS)

I. Introduction

Mobile ad-hoc networks propose new challenges for software development and verification and validation (V&V) activities. Apart from the issues found in fixed distributed systems, fresh ones are presented in the new environment: high dynamicity or context awareness. New nodes are constantly joining and leaving, the application running on the host has to be aware of these changes. Nodes are moving out of each other's communication range frequently, hence the failure of sending a message is not a rare event any more. The state of an application depends not only on the messages it receives from the others, it should also take into account its context, e.g., its current location coordinates supplied by a GPS unit or other information from the environment. Thus the testing methodology of these systems should take into account these specificities. The current modeling languages for specifying test cases have to be adapted to these requirements.

This paper presents the typical problems of mobile systems through a case study. Section II presents the related work in modeling mobile systems, Section III describes the case study and the results of its analysis, while Section IV illustrates with examples why UML 2.0 Sequence Diagrams need extensions when used for specifying tests for mobile systems.

II. Related work

According to our research, currently there is no standard for modeling mobile systems yet, but in the recent years several approaches have emerged. A number of publications focus on mobile agents from the broad area of mobile systems. An agent is a software component that executes specific tasks on behalf of someone with some autonomy [1]. Mobile Agent Modeling with UML is a UML profile recommended by Belloni and Marcos in [1]. The stereotypes and tagged values of the profile are organized into views that describe the different aspects of the mobile agent. In [4] mobile computing (MC) environments were investigated. Because Objectcharts (which are variants of Harel's Statecharts) turned out to be inadequate to model such environments, an extension called Mobicharts was proposed. The extension contained specific states to model, e.g., the situation when the mobile host is disconnected and mechanisms to express task migration.

Grassi et al. proposed an UML profile to support physical mobility of the computing nodes and the logical mobility of software elements [2]. The behavior of mobility was expressed on so-called mobility manager statecharts. The paper included examples to show how the profile can be applied to describe basic mobile code paradigms (e.g. Code on Demand, Mobile Agent). In [3] a UML extension called Mobile UML was proposed to model mobile systems in global computing. The extensions consisted of (i) a UML profile to express mobility concepts (*location, mobile, mobile location*) and (ii) new diagram types. According to the authors, the problem with UML Sequence diagrams when modeling mobile scenarios is that movement of an entity can be expressed only indirectly by adding a new object box. Thus, to overcome the complexity of this approach, a new diagram type Sequence Diagram for Mobility (SDM) was recommended.

Several approaches have been proposed, that contain many similar elements, however, each of them are specialized for a specific aspect of mobile systems, and no general standard is available at the moment. Moreover, these extensions mainly consider logical mobility or physical mobility from one infrastructure point to the other, and they do not offer a solution to ad-hoc networks.

III. GMP Case Study

This chapter includes the insights gained from the detailed analysis of a Group Membership Protocol (GMP) [5]. We choose this particular application because it is a good example of a non-trivial, mobile-based service. It addresses a very complex problem, i.e., to maintain a consistent membership information in a mobile setting, where besides the challenges raised by traditional distributed systems (e.g. atomicity, asynchronous behavior) problems from the mobile environment (e.g. frequent topology changes, network delays) also arise. The protocol has a specification [5] which contains (i) general properties the protocol should satisfy (e.g., the successor of a group shall be either a proper superset or a proper subset of the group), (ii) textual description and (iii) pseudo-codes describing the important methods. Moreover, the authors created a Java language implementation as part of the LIME [6] open source middleware for mobile applications. The implementation is not just a small example program: it consists of 4 KLOC of Java code, contains 22 Java classes and after all the components are started there are 6 concurrent threads.

The functionality of the protocol is divided into two parts: (i) group discovery manages the discovery and reporting of newly arrived hosts, (ii) group reconfiguration performs the merging and splitting of groups when needed. The protocol uses a centralized approach, every group has one leader. The leader collects the location and discovery information of the hosts. Using this information, it checks the group merging and splitting criterion, and starts a group change operation if necessary. The criterion used is the *safe distance* criterion, i.e., if two nodes are within this distance, the protocol guarantees that, regardless of their moving pattern, they will have enough time to finish their current communication.

The analysis was conducted by (1) reviewing the specification, (2) creating a UML model for the implementation of the protocol, (3) comparing the specification to the implementation, and (4) testing the implementation. The general properties of the protocol were analyzed to determine whether they are testable or not. The description of the protocol was reviewed; worst-case scenarios were investigated to see whether the calculation of safe distance and the atomicity of the group changes are always valid. UML class diagrams were created to model the static structure of the implementation, this helped to check conformance to the specification. Sequence diagrams were drawn for each of the important scenarios, which revealed design failures and possibly invalid scenarios. Finally, simple random testing was carried out on the implementation which found several scenarios violating the properties of the protocol. Several of these scenarios were anticipated previously by the review. In summary, the following main issues were found during the review (the detailed description of the analysis and the results can be found in [7]):

- Some of the general properties of the protocol are incomplete or not testable.
- The English language specification of the protocol is sometimes ambiguous, and the pseudo code definition of the key functions is also not sufficient, the control flow is missing.
- The atomicity of the protocol is not guaranteed in worst-case scenarios.
- The implementation lacks key features that are essential to the correct behavior.

The analysis showed general problems that are relevant for any mobile application dealing with mobility and cooperation of hosts. The same analysis (e.g., testable properties, atomicity of operations, identifying unclear parts in the specification and modeling static and dynamic structure) could be performed for any mobile application. Moreover, the case study highlighted the following general challenges:

- It is not easy to model mobile system instances. Without a suitable notation and modeling methodology serious design defects could be introduced.
- The definition of properties containing spatial and temporal information is a complex task, but the correct formulation is essential to the later verification steps.

IV. Expressing Tests using UML 2.0 Sequence Diagrams

UML 2.0 introduced a major change in the sequence diagrams (SD), many new elements were imported from other scenario languages like Message Sequence Charts (MSC). Several operators were introduced to combine diagram fragments, e.g. *parallel* and *alternate* execution. *Negation* and *assert* operators could be used to specify modalities, *ignore* and *consider* operators can express that the diagram shows only a subset of messages. Using the previously presented GMP case study we investigated how these new elements can be used in testing by trying to specify test cases.

Using the UML 2.0 SD language a test case for a split scenario in the GMP can be expressed like the following. The *goal* of the test case is to check that if a node moves out of safe distance the leader detects it and sends the new group membership to everyone before the node leaves the communication range. This behavior can be expressed with the sequence diagram on Figure 1 showing the messages exchanged during the split operation.

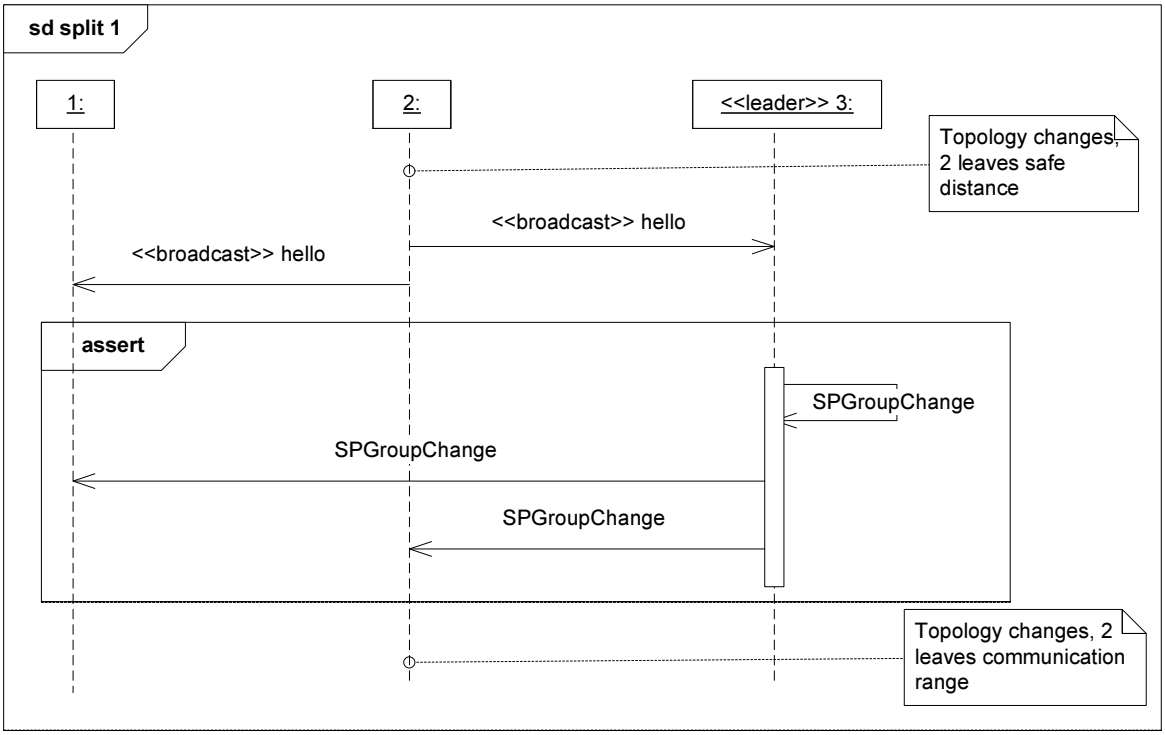

Figure 1: Test scenario for a group split

The above diagram uses only standard UML concepts. Using comments and stereotypes the scenario can be partially described, although it can be seen that UML sequence diagrams lack the element to express two important constructs frequent in mobile environments in a precise way:

- broadcasting messages in local vicinity,
- context changes, like two nodes moving out of each other's range.

Apart from the problem of missing elements to describe the mobile environment, the other challenge with Sequence Diagrams is the lack of well-defined semantics. Semantics problem appear e.g., in the assignment of the verdict. The test is passed if node 1 and node 2 receive the SPGroupChange message with the correct group membership. The *assert* operator could be used to show explicitly what are the required messages, if the messages in the assert fragment do not appear, that particular trace is considered as not valid. However, as reported earlier in literature, e.g., in [8], the definition of assert's semantics is quite problematic.

The next example illustrates a more general semantics issue. Figure 2 is valid according to the UML specification; however it raises serious causal issues. For example sending message *z* must

occur after receiving *y*, because they are on the same lifeline, thus sending *z* is after the sending of *x*. But if *y* is not received (it is contained in an optional fragment), this causal ordering is not valid as sending *x* and sending *z* become concurrent activities.

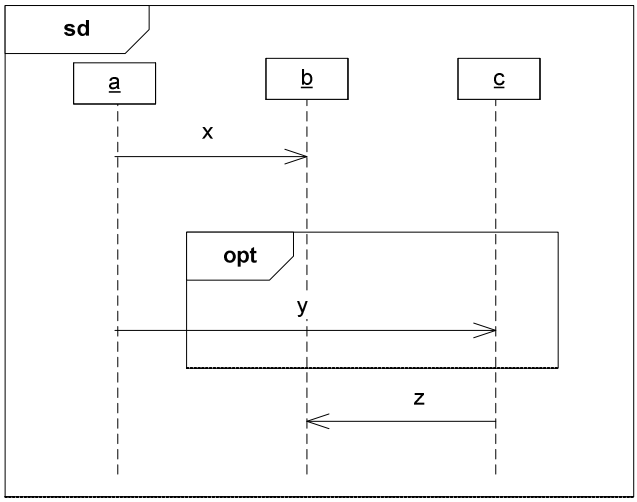

Figure 2: Example for semantic problems in UML Sequence Diagrams

It can be seen even from these examples, that there are several challenges with UML 2.0 Sequence Diagrams if used for test case specification. The UML 2.0 version contains too much complex language elements with ambiguous semantics. Possibly a narrower set of elements, but with a well-defined semantics would be much more useful for specifying precise test cases.

V. Conclusion

In this paper we presented a case study, an analysis of a Group Membership Protocol, which revealed the typical challenges in modeling and testing mobile systems. Moreover, it turned out that UML Sequence Diagrams, which are frequently used for describing test cases, lack the concepts to express the high dynamicity and context awareness of mobile systems. Based on the experiences gained in the case study our future work is to specify extensions, with a well-defined semantics, that adapt Sequence Diagrams to this environment.

References

[1] E. Belloni and C. Marcos, "MAM-UML: An UML Profile for the Modelling of Mobile-Agent Applications," in *Proceedings of the XXIV International Conference of the Chilean Computer Science Society (SCCC'04)*, 2004

[2] V. Grassi, R. Mirandola, and A. Sabetta, "A UML Profile to Model Mobile Systems," in *The Unified Modelling Language, LNCS 3273*, Springer, 2004

[3] H. Baumeister, N. Koch, P. Kosiuczenko, P. Stevens, and M. Wirsing, „UML for Global Computing," In *Proc. of IST/FET Int. Workshop on Global Computing (GC'03)*, Revised Selected Papers, LNCS 2874, Springer, 2003

[4] S. Acharya, R.K. Shyamasundar, "MOBICHARTS: A Notation to Specify Mobile Computing Applications," in *Proc. of 36th Hawaii International Conference on System Sciences (HICSS-36 2003)*, January 6-9, 2003, Big Island, HI, USA. IEEE Computer Society, 2003, ISBN 0-7695-1874-5

[5] Q. Huang, C. Julien, and G. Roman, "Relying on Safe Distance to Achieve Strong Partitionable Group Membership in Ad Hoc Networks," *IEEE Transactions on Mobile Computing* 3, 2 (Apr. 2004).

[6] Lime, Middleware for mobile applications, URL: http://lime.sourceforge.net/

[7] Z. Micskei, H. Waeselynck, M. D. Nguyen, and N. Riviere, "Analysis of a group membership protocol for Ad-hoc networks," LAAS Technical Report *no. 06797*, November 2006

[8] D. Harel and S. Maoz, "Assert and negate revisited: modal semantics for UML sequence diagrams," in *Proc. of SCESM '06*, Shanghai, China, 2006.