Towards the Online Testing of Distributed and Heterogeneous Systems with Extended Petri Nets

Bruno Lima*† and João Pascoal Faria*†
*INESC TEC,
FEUP campus, Rua Dr. Roberto Frias, s/n 4200-465 Porto, Portugal
†Faculty of Engineering, University of Porto,
Rua Dr. Roberto Frias, s/n 4200-465 Porto, Portugal

{bruno.lima, jpf}@fe.up.pt

***Abstract*—The growing dependence of our society on increasingly complex software systems makes software testing ever more important and challenging. In many domains, such as healthcare and transportation, several independent systems, forming a heterogeneous and distributed system of systems, are involved in the provisioning of end-to-end services to users. However, existing testing techniques, namely in the model-based testing field, provide little support for properly testing such systems.**

To bridge the gaps identified in the state of the art we intend to develop a research work where the main goal is to significantly reduce the cost of testing distributed and heterogeneous systems, from the standpoint of time, resources and expertise required, as compared to existing approaches. For that, we propose a preliminary approach and a toolset architecture for automating the testing of end-to-end services in distributed and heterogeneous systems. The tester interacts with a visual modeling frontend to describe key behavioral scenarios, invoke test generation and execution, and visualize test results and coverage information back in the model. The visual modeling notation is converted to a formal notation amenable for runtime interpretation in the backend. A distributed test monitoring and control infrastructure is responsible for interacting with the components of the system under test, as test driver, monitor and stub. At the core of the toolset, a test execution engine coordinates test execution and checks the conformance of the observed execution trace with the expectations derived from the visual model. A real world example from the Ambient Assisted Living domain is presented to illustrate the approach.

As future work we intend to develop distributed and incremental algorithms for online testing of distributed and heterogeneous systems based on Extended Petri Nets at runtime and validate them in real world case studies.

***Index Terms*—Software Testing; Distributed algorithms; UML; Petri nets;**

I. Introduction

Due to the increasing ubiquity, complexity, criticality and need for assurance of software based systems [1], testing is a fundamental lifecycle activity, with a huge economic impact if not performed adequately [2]. Such trends, combined with the needs for shorter delivery times and reduced costs, demand for the continuous improvement of software testing methods and tools, in order to make testing activities more effective and efficient.

Nowadays software is not more like simple applications but has evolved to large and complex system of systems [3]. A system of systems consists of a set of small independent systems that together form a new system. The system of systems can be a combination of hardware components (sensors, actuators, etc.) and software systems used to create big systems or ecosystems that can offer multiple different services. Currently, systems of systems capture a great interest from the software engineering research community. Testing these distributed and heterogeneous software systems or systems of systems, running over interconnected mobile and cloud based platforms, is particularly important and challenging. Some of the challenges are: the difficulty to test the system as a whole due to the number and diversity of individual components; the difficulty to coordinate and synchronize the test participants and interactions, due to the distributed nature of the system; the difficulty to test the components individually, because of the dependencies on other components.

An example of a distributed and heterogeneous system is the Ambient Assisted Living (AAL) ecosystem that was prototyped in the context of the nationwide AAL4ALL project [4]. The AAL4ALL ecosystem comprises a set of interoperable AAL products and services (sensors, actuators, mobile and web based applications and services, middleware components, etc.), produced by different manufacturers using different technologies and communication protocols (web services, message queues, etc.). To assure interoperability and the integrity of the ecosystem, it was developed and piloted a testing and certification methodology [5], encompassing the specification of standard interfaces and component categories, the specification of unit (component) and integration test scenarios, and the test implementation and execution on candidate components by independent test laboratories. A major problem faced during test implementation and execution was related with test automation, due to the diversity of component types and communication interfaces, the distributed nature of the system, and the lack of support tools. Similar difficulties have been reported in other domains, such as the railway domain [6]. In fact, we found in the literature limited tool support for automating the whole process of specification-based (or model-based) testing of distributed and heterogeneous systems, as will be explained in Section II.

II. State-of-the-art

A. Model-based testing

Model-based testing (MBT) techniques and tools have attracted increasing interest from academia and industry, because of their potential to increase the effectiveness and efficiency of the test process, by means of the automatic generation of test cases (test sequences, input test data, and expected outputs) from behavioral models of the system under test (SUT) [7].

However, MBT approaches found in the literature suffer from several limitations [8]. The most common practical limitation is the lack of integrated support for the whole test process. This is a big obstacle for the adoption of these approaches in practice by industry, because of the effort required to create or adapt tools to implement some parts of the test process. Other limitations are the difficulty to bridge the gap between the model and the implementation, and the difficulty to control the test case explosion problem (i.e., the combinatorial explosion in the number of test cases generated from models).

MBT can be done *offline* or *online* (also called *adaptive* or *on-the-fly*). Offline testing means that test cases are first generated and subsequently executed [9], while in online testing test generation and execution are performed together so that the test generator can react to how the SUT behaves [10]. The use of online testing is necessary if the SUT is non-deterministic, because the test generator can see which path the SUT has taken, and follow the same path in the model [11].

We focus our research work on the *online* testing of distributed and heterogeneous systems (instead of offline testing), to cope with non-determinism in the implementation or in the specification. We also propose a toolset architecture to support the whole test process in an integrated fashion.

B. Models

MBT approaches use a high variety of models. In general, one can distinguish state based and scenario based approaches [12]. State based approaches use abstract state machines [13], UML state machines [14], input-output automata [15] or similar behavioral models for describing *all possible behaviors* of the system or its components. Scenario-based approaches use UML sequence diagrams (SDs), message sequence charts (MSC) [16] or similar behavioral models for describing interactions between components of the system or interactions between the system and the environment that occur in specific contexts, representing *key system behaviors*. State-based models are best suited for capturing system design decisions and are usually more detailed, whilst scenario-based models are best suited for capturing system requirements [12] and are usually less detailed.

We focus our research work on the *scenario-based* testing of distributed and heterogeneous systems (instead of state-based testing), because scenario-based models are more convenient for describing and visualizing the interactions that occur between the components and actors of a distributed system [17] in key scenarios. Scenario-based models also help partially avoiding the test case explosion problem. To facilitate industrial adoption, we opted for using UML sequence diagrams (SDs) [18][19], with a minimum number of restrictions and extensions, as the input behavioral models. However, UML SDs are not well suited for incremental execution at runtime (as required by an online testing strategy); for that, we opted to use extended Petri Nets [20], for efficient incremental execution and conformance checking, as in our previous work for object-oriented systems [21]. However, in that work only the interaction between objects in a standalone java program were tested, with limited support for parallelism and concurrency. In the current work, we intend to extend that approach for distributed and heterogeneous systems. The extended Petri nets used in our previous work were Event-Driven Colored Petri Nets. In the current work, we intend to add time constraints, so the extended Petri nets will be Timed Event-Driven Colored Petri Nets (TEDCPN). Translation rules from UML SDs (namely time constraints) to TEDCPNs will have to be defined.

C. Distributed testing architectures

One difficulty in testing distributed systems is that their distributed nature imposes theoretical limitations on the conformance faults that can be detected by the test components, depending on the test architecture used [22], [23]. Two basic test architectures have been proposed in the past to test distributed systems: a purely distributed test architecture with independent local testers communicating synchronously with the components of the SUT [24]; a purely centralized test architecture, in which a single centralized tester interacts asynchronously with the components of the SUT. More recently, Hierons [23] proposed a hybrid framework that combine local testers and a centralized tester. He proved that this architecture is more powerful than the distributed and centralized approaches, i.e., it has a higher fault detection capability. However, his work is only concerned with conformance relations between execution traces and specifications based on input-output automata, without addressing algorithms for distributed and incremental online test generation and execution.

Given its advantages, we base our research work on the *hybrid test architecture* proposed by Hierons [23].

III. Research objectives and methodological approach

A. Research hypothesis

Our research hypothesis (that we expect to prove at the end of the research work) is:

> Using an online model-based testing approach, with UML SDs as input models and Extended Petri Nets at runtime, on top of a hybrid test execution architecture, it is possible to fully automate the testing of distributed and heterogeneous systems, at the integration, unit and system level, in an effective, efficient and accessible way.

By 'fully automate' we mean that the only manual activity needed should be the creation of the model required as input for automatic test case generation and execution, together with mapping information between the model and the implementation (e.g., location of system components), without the need to develop test components specific for each SUT. Besides that, all phases of the test process should be supported in an integrated fashion. This will be achieved by:

- the automatic translation of the input behavioral models (UML SDs) to the formal models needed at runtime (Extended Petri Nets, or more specifically, Timed Event-Driven Colored Petri Nets);
- the automatic test input generation (generation of test inputs to the SUT) and conformance checking (checking actual SUT responses against expected ones) based on executable models and mapping information at runtime;
- the provision of 'local testers' (responsible for interacting directly with the SUT components) specific for each platform;
- the automatic reporting of test results (conformance errors and coverage information) back in the input model (with colors and annotations).

By 'effective' we mean an approach with a high fault detection and localization capability. This will be achieved as follows:

- by adopting a hybrid test architecture allowing the detection of a higher number of errors as compared to purely distributed or centralized architectures [23];
- by monitoring and checking (against the specification) the interactions between components in the SUT, besides the interactions between the SUT and the environment;
- by using an incremental conformance checking algorithm, that allows capturing the execution state of the SUT as soon as a failure occurs, and hence facilitate subsequent fault diagnosis;
- by following an online, adaptive, test generation strategy, that allows testing non-deterministic SUT behaviors;
- by testing temporal constraints.

By 'efficient' we mean efficient test execution. This will be achieved as follows:

- by using a distributed conformance checking algorithm, that minimizes communication overheads during test execution;
- by using a runtime model that allows a more efficient model execution and conformance checking.

By 'accessible' we mean an approach accessible for testers in industry. This will be achieved as follows:

- by using a user-friendly input notation, following industry standards;
- by following a scenario-based approach, that usually requires less detailed input models and closer to the requirements than state-based approaches.

.

Regarding the test levels, we intend to focus on integration testing of end-to-end services, but supporting also system testing (in which internal interactions between components of the system need not be monitored) and unit testing (in which system components are tested in isolation). In our envisioned approach, the same input model can be used to perform tests at different levels (unit, integration, and system testing), simply by changing the selection of observable and controllable events in the input model. For unit testing purposes, all the messages in the model will be considered controllable (i.e., to be generated by the test harnessed) except for the messages departing from the component under test.

B. Sub-objectives and research questions

To prove the research hypothesis, the following sub-objectives will be pursued:

- define translation rules from the input model (UML SDs) to the runtime model (Extended Petri Nets, e.g., Timed Event-Driven Colored Petri Nets);
- define a distributed and incremental algorithm for conformance checking of the observed execution trace against the expectations set by the runtime model;
- define a distributed and incremental algorithm for test input generation based on the current execution status and coverage level of the runtime model;
- define rules for translating the test results (conformance errors and coverage information) back to annotations the input model;
- develop a toolset prototype and conduct experiments in real world case studies.

To achieve the sub-objectives we aim at responding to the following research questions:

- **RQ1**: Which extensions of Petri Nets should be used?
- **RQ2**: How to translate temporal constraints from UML SD to such extended Petri Nets?
- **RQ3**: How to partition the Petri Net for distributed execution?
- **RQ4**: How to choose the next test action in a distributed environment?
- **RQ5**: How to translate the results back to the UML SD?

C. Methodology

This work will follow the Engineering method[25]:

> Engineers develop and test a solution to a hypothesis. Based upon the results of the test, they improve the solution until it requires no further improvement.

Validation will be performed by case studies.

IV. Past work and preliminary results

Until this moment it was conducted research work regarding the state of the art analysis, the state of the practice analysis, and the conception of the overall solution.

Regarding the state of the art analysis, it was conducted a systematic analysis of works on model based testing and distributed systems testing.

Regarding the state of the practice analysis, we conducted a survey on "Testing Distributed and Heterogeneous Systems − State of the practice" (available at https://goo.gl/GExS2w),

that was distributed to the participants of two industry-oriented conferences in the software testing area (TESTING Portugal 2015 and UCAAT 2015 - ETSI User Conference on Advanced Automated Testing) and was responded by 147 persons. The main goal of this work is to explore the testing of DHS from the point of view of industry practitioners, in order to assess the current state of the practice and identify opportunities and priorities for research and innovation initiatives. More precisely, we aim at responding to the following research questions:

- **SRQ1**: How relevant are DHS in the software testing practice?
- **SRQ2**: What are the most important features to be tested in DHS?
- **SRQ3**: What is the current status of test automation and tool sourcing for testing DHS?
- **SRQ4**: What are the most desired features in test automation solutions for DHS?

The survey allowed us to confirm the high relevance of DHS in software testing practice, confirm and prioritize the relevance of testing features characteristics of DHS, confirm the existence of a significant gap between the current and the desired status of test automation for DHS, and confirm and prioritize the relevance of test automation features for DHS. A complete analysis of the survey results can be found in [26].

Regarding the conception of the overall solution, we conceived a preliminary approach and a toolset architecture (see Figure 1) for automating the testing of end-to-end services in distributed and heterogeneous systems.

In the envisioned approach, the tester interacts with a visual modeling frontend to describe key behavioral scenarios based on UML sequence diagrams, invoke test generation and execution, and visualize test results and coverage information back in the model (see Figure 2). The visual modeling notation is converted to a formal notation amenable for runtime interpretation in the backend based on extended Petri Nets (see Figure 3). A distributed test monitoring and control infrastructure is responsible for interacting with the components of the system under test, as test driver, monitor and stub. At the core of the toolset, a test execution engine coordinates test execution and checks the conformance of the observed execution trace with the expectations derived from the visual model. This approach was fully described in [17].

V. Future Work and Expected Results

As future work we intend to perform the following activities:

- **Define translation rules**: Define the translation rules for conversion between the visual modeling notation (UML SDs) and the formal modeling notation (Extended Petri Nets);
- **Algorithms design**: Define and formally specify distributed and incremental algorithms for online testing of distributed and heterogeneous systems (namely, for conformance checking and test input generation);

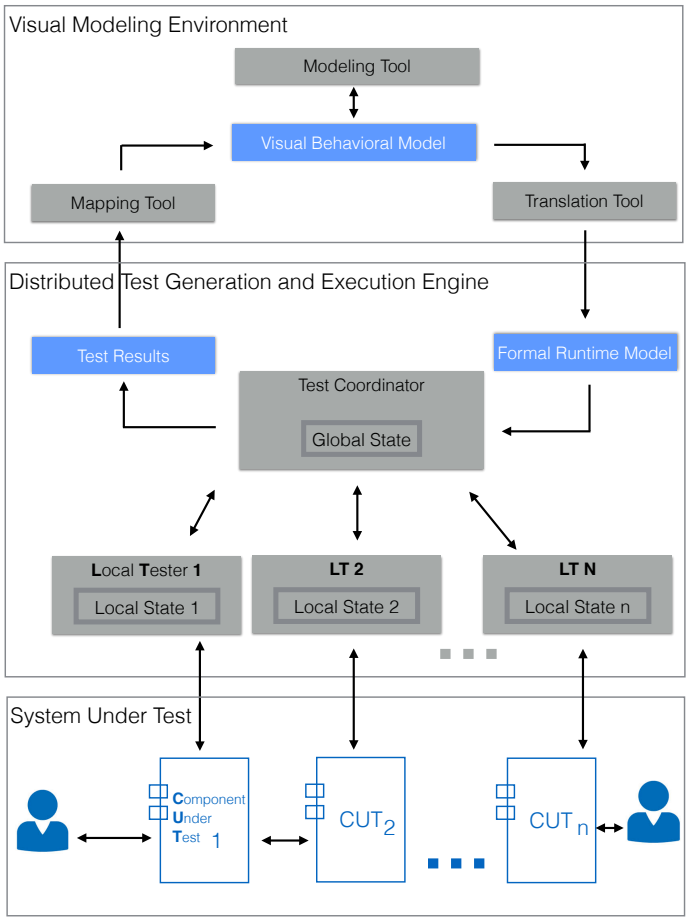


Fig. 1. Proposed toolset architecture

- **Algorithms validation**: Verify and validate the above algorithms using formal verification and validation and testing techniques;
- **Practial implementation**: Implement a prototype necessary for automating the proposed approach, using the algorithms previously defined;
- **Final validation**: Validate the solution approach and toolset prototype with respect to the research hypothesis previously described (section III-A) in real world case studies, one of which in the Ambient Assisted Living domain.

The expected results of our work are:

- Translation rules from UML SDs to extended Petri Nets
- Distributed and incremental algorithms for conformance checking and test input generation
- Toolset prototype
- Experimental results

VI. Conclusions

It was presented a research plan and initial research results for automated scenario-based testing of distributed and heterogeneous systems to overcome the limitations that exist in the state of the art for testing these systems.

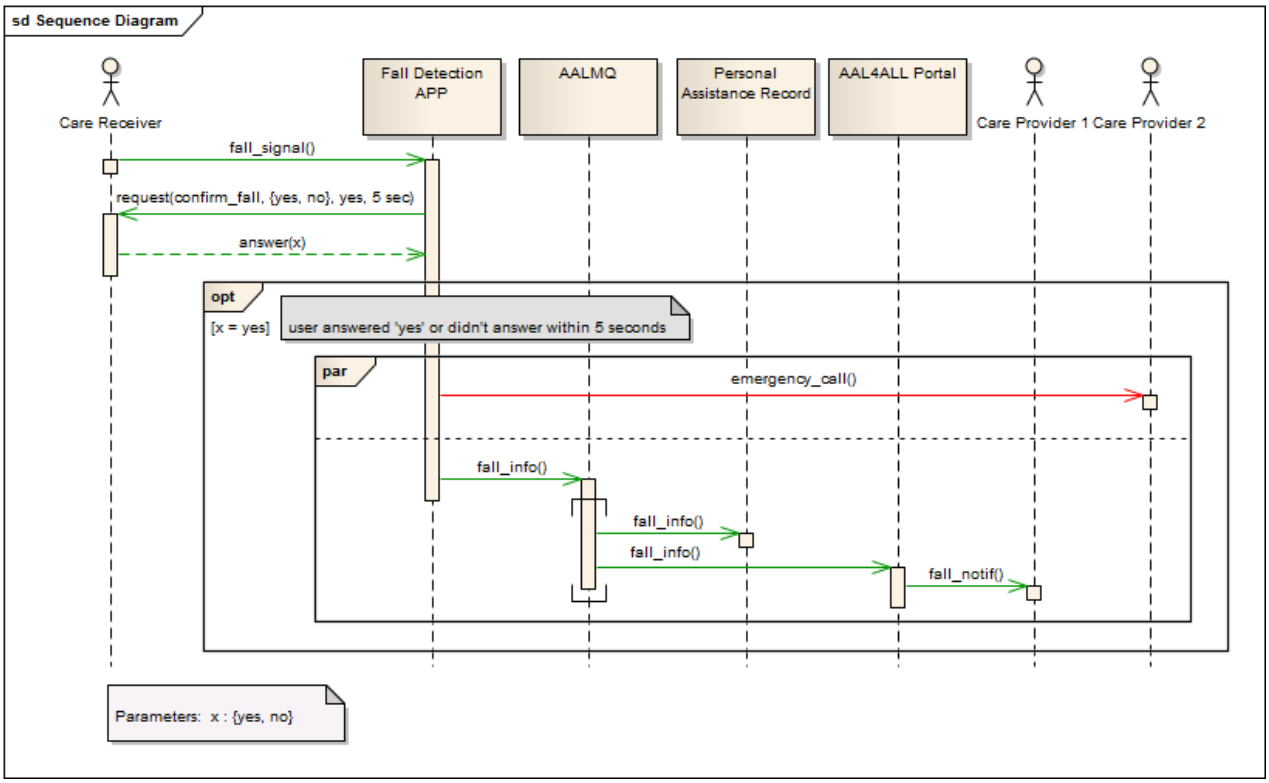


Fig. 2. UML sequence diagram representing the interactions in a fall detection scenario. The diagram is already painted after a failed test execution in which the fall detection application did not send an emergency call [17].

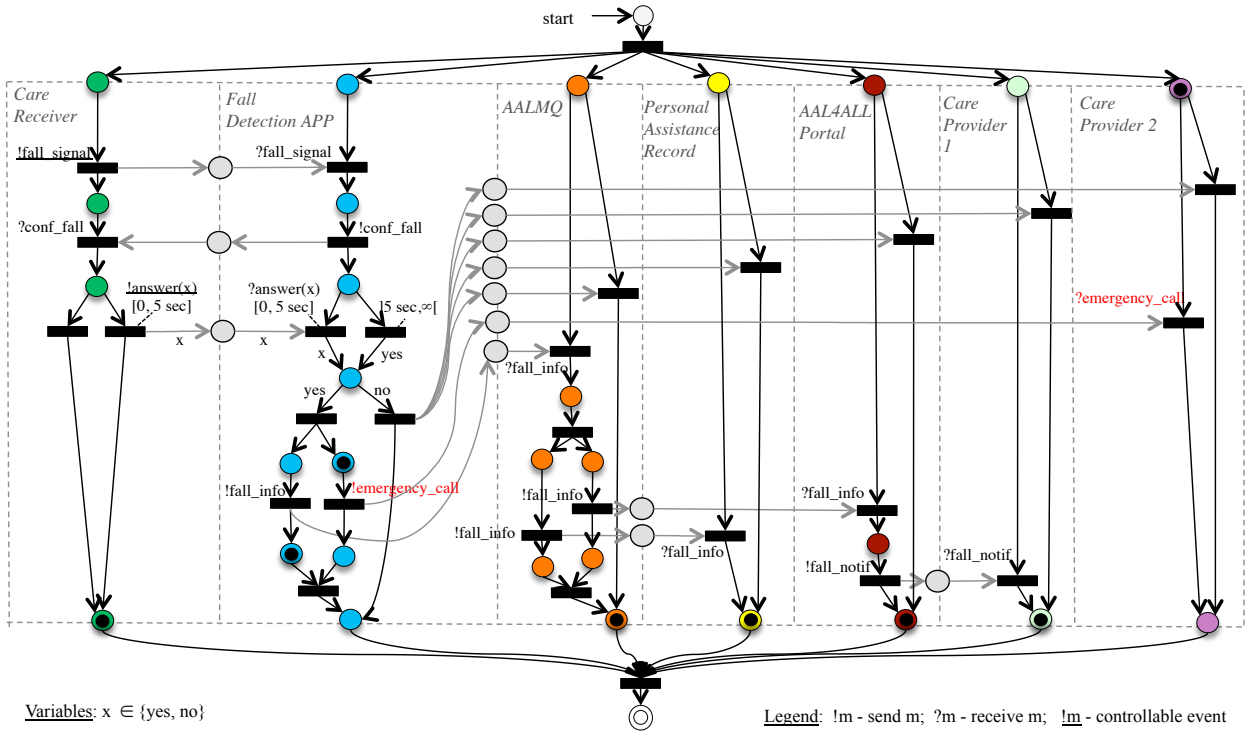


Fig. 3. TEDCPN derived from the SD of Figure 2. The net is marked in a final state of a failed test execution in which the fall detection application did not send an emergency call [17].

ACKNOWLEDGMENT

This research work was performed in scope of the project NanoSTIMA. Project "NanoSTIMA: Macro-to-Nano Human Sensing: Towards Integrated Multimodal Health Monitoring and Analytics/NORTE-01-0145-FEDER-000016" is financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

REFERENCES

[1] B. Boehm, "Some Future Software Engineering Opportunities and Challenges," in *The Future of Software Engineering*, S. Nanz, Ed. Springer Berlin Heidelberg, 2011, pp. 1–32. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15187-3_1

[2] G. Tassey, "The Economic Impacts of Inadequate Infrastructure for Software Testing," National Institute of Standards and Technology, Tech. Rep., 2002.

[3] DoD, "Systems Engineering Guide for Systems of Systems," Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering Version 1.0, Tech. Rep., 2008.

[4] AAL4ALL, "Ambient Assisted Living For All," http://www.aal4all.org, 2015.

[5] J. P. Faria, B. Lima, T. B. Sousa, and A. Martins, "A Testing and Certification Methodology for an Open Ambient-Assisted Living Ecosystem," *International Journal of E-Health and Medical Communications (IJEHMC)*, vol. 5, no. 4, pp. 90–107, 2014.

[6] C. Torens and L. Ebrecht, "RemoteTest: A Framework for Testing Distributed Systems," in *Software Engineering Advances (ICSEA), 2010 Fifth International Conference on*, Aug 2010, pp. 441–446.

[7] M. Utting and B. Legeard, *Practical Model-Based Testing: A Tools Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[8] A. C. Dias Neto, R. Subramanyan, M. Vieira, and G. H. Travassos, "A Survey on Model-based Testing Approaches: A Systematic Review," in *Proceedings of the 1st ACM International Workshop on Empirical Assessment of Software Engineering Languages and Technologies: Held in Conjunction with the 22Nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007*, ser. WEASELTech '07. New York, NY, USA: ACM, 2007, pp. 31–36. [Online]. Available: http://doi.acm.org/10.1145/1353673.1353681

[9] S. Schulz, J. Honkola, and A. Huima, "Towards model-based testing with architecture models," in *Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the*. IEEE, 2007, pp. 495–502.

[10] M. Mikucionis, K. G. Larsen, and B. Nielsen, "T-uppaal: Online model-based testing of real-time systems," in *Automated Software Engineering, 2004. Proceedings. 19th International Conference on*. IEEE, 2004, pp. 396–397.

[11] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," *Software Testing, Verification and Reliability*, vol. 22, no. 5, pp. 297–312, 2012. [Online]. Available: http://dx.doi.org/10.1002/stvr.456

[12] W. Grieskamp, *Formal Approaches to Software Testing and Runtime Verification: First Combined International Workshops, FATES 2006 and RV 2006, Seattle, WA, USA, August 15-16, 2006, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, ch. Multi-paradigmatic Model-Based Testing, pp. 1–19. [Online]. Available: http://dx.doi.org/10.1007/11940197_1

[13] Microsoft, "Spec Explorer," http://research.microsoft.com/en-us/projects/specexplorer/, May 2016.

[14] A. Huima, "Implementing conformiq qtronic," in *Testing of Software and Communicating Systems*. Springer, 2007, pp. 1–12.

[15] Q. Tani and A. Petrenko, "Input/output automata'," in *Testing of Communicating Systems: Proceedings of the IFIP TC6 11th International Workshop on Testing of Communicating Systems (IWTCS'98) August 31-September 2, 1998, Tomsk, Russia*, vol. 3. Springer, 2013, p. 83.

[16] W. Damm and D. Harel, "Lscs: Breathing life into message sequence charts," *Formal methods in system design*, vol. 19, no. 1, pp. 45–80, 2001.

[17] B. Lima and J. P. Faria, *Software Technologies: 10th International Joint Conference, ICSOFT 2015, Colmar, France, July 20-22, 2015, Revised Selected Papers*. Cham: Springer International Publishing, 2016, ch. Automated Testing of Distributed and Heterogeneous Systems Based on UML Sequence Diagrams, pp. 380–396. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-30142-6_21

[18] OMG, "OMG Unified Modeling Language ^{TM} (OMG UML), Superstructure," Object Management Group, Tech. Rep., 2011.

[19] H.-G. Gross, *Component-Based Software Testing with UML*. Springer Berlin Heidelberg, 2005.

[20] K. Jensen, *Coloured Petri nets: basic concepts, analysis methods and practical use*. Springer Science & Business Media, 2013, vol. 1.

[21] J. P. Faria and A. C. R. Paiva, "A toolset for conformance testing against UML sequence diagrams based on event-driven colored Petri nets," *International Journal on Software Tools for Technology Transfer*, pp. 1–20, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10009-014-0354-x

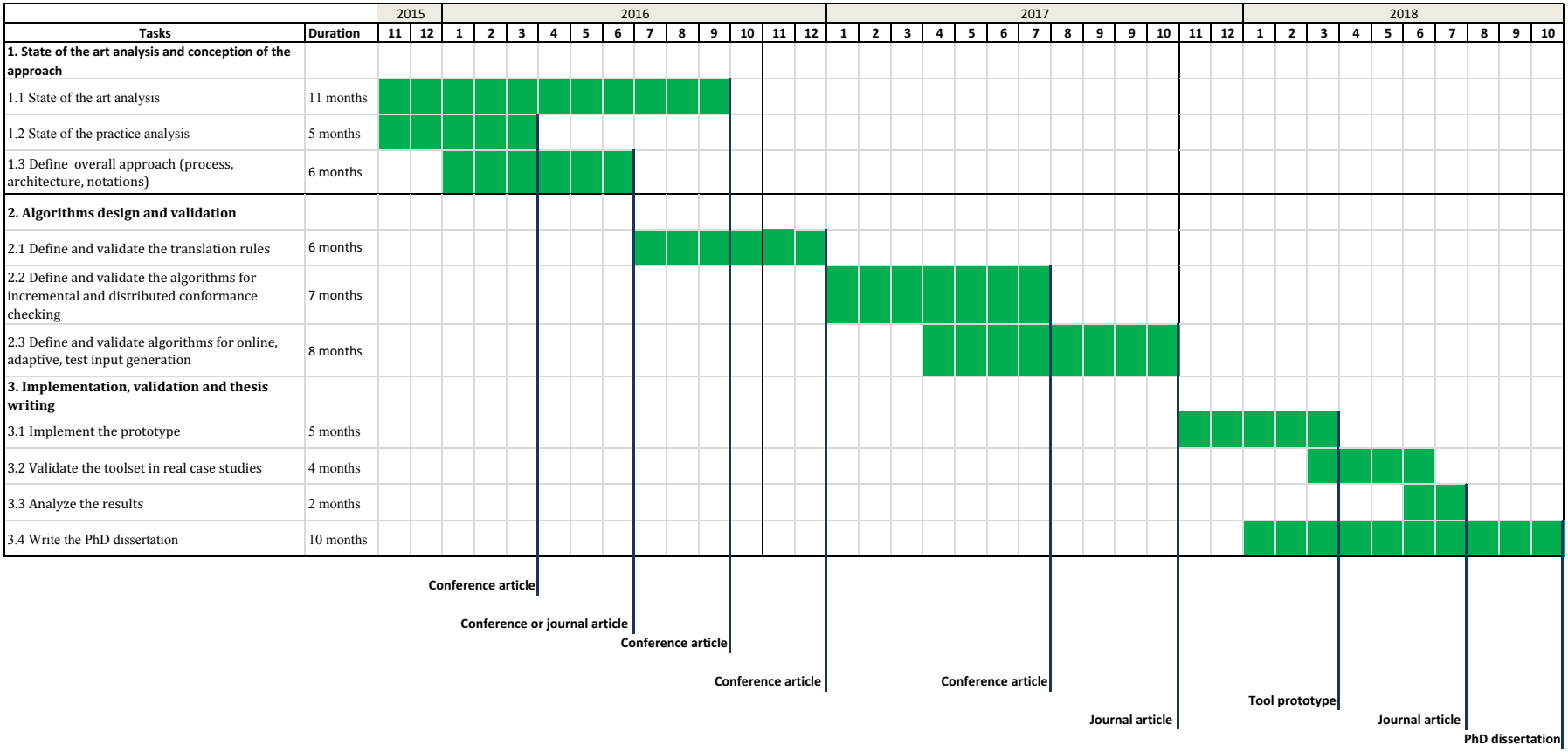
[22] R. M. Hierons, M. G. Merayo, and M. Núñez, "Scenarios-based testing of systems with distributed ports," *Software: Practice and Experience*, vol. 41, no. 10, pp. 999–1026, 2011. [Online]. Available: http://dx.doi.org/10.1002/spe.1062

[23] R. M. Hierons, "Combining Centralised and Distributed Testing," *ACM Trans. Softw. Eng. Methodol.*, vol. 24, no. 1, pp. 5:1–5:29, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2661296

[24] A. Ulrich and H. König, "Architectures for Testing Distributed Systems," in *Testing of Communicating Systems*, ser. IFIP — The International Federation for Information Processing, G. Csopaki, S. Dibuz, and K. Tarnay, Eds. Springer US, 1999, vol. 21, pp. 93–108. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-35567-2_7

[25] M. V. Zelkowitz and D. R. Wallace, "Experimental models for validating technology," *Computer*, vol. 31, no. 5, pp. 23–31, 1998.

[26] B. Lima and J. P. Faria, "Testing distributed and heterogeneous systems: State of the practice," in *Proceedings of the 10th International Conference on Software Engineering and Applications*, 2016.


2015
2016
2017
2018
Tasks
Duration
11 12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10
1. State of the art analysis and conception of the approach
1.1 State of the art analysis
11 months
1.2 State of the practice analysis
5 months
1.3 Define overall approach (process, architecture, notations)
6 months
2. Algorithms design and validation
2.1 Define and validate the translation rules
6 months
2.2 Define and validate the algorithms for incremental and distributed conformance checking
7 months
2.3 Define and validate algorithms for online, adaptive, test input generation
8 months
3. Implementation, validation and thesis writing
3.1 Implement the prototype
5 months
3.2 Validate the toolset in real case studies
4 months
3.3 Analyze the results
2 months
3.4 Write the PhD dissertation
10 months
Conference article
Conference or journal article
Conference article
Conference article
Conference article
Journal article
Tool prototype
Journal article
PhD dissertation

Towards the Online Testing of Distributed and Heterogeneous Systems Based on Extended Petri Nets

INESCTEC | ASSOCIATE LABORATORY PORTUGAL

Bruno Lima and João Pascoal Faria

{bruno.lima, jpf}@fe.up.pt

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Hypothesis

Using a scenario-oriented model-based adaptive (online) testing approach, with extended Petri Nets at runtime, on top of a hybrid test execution architecture, it is possible to fully automate the testing of distributed and heterogeneous systems, at the integration, unit and system level, in an effective, efficient and accessible way.

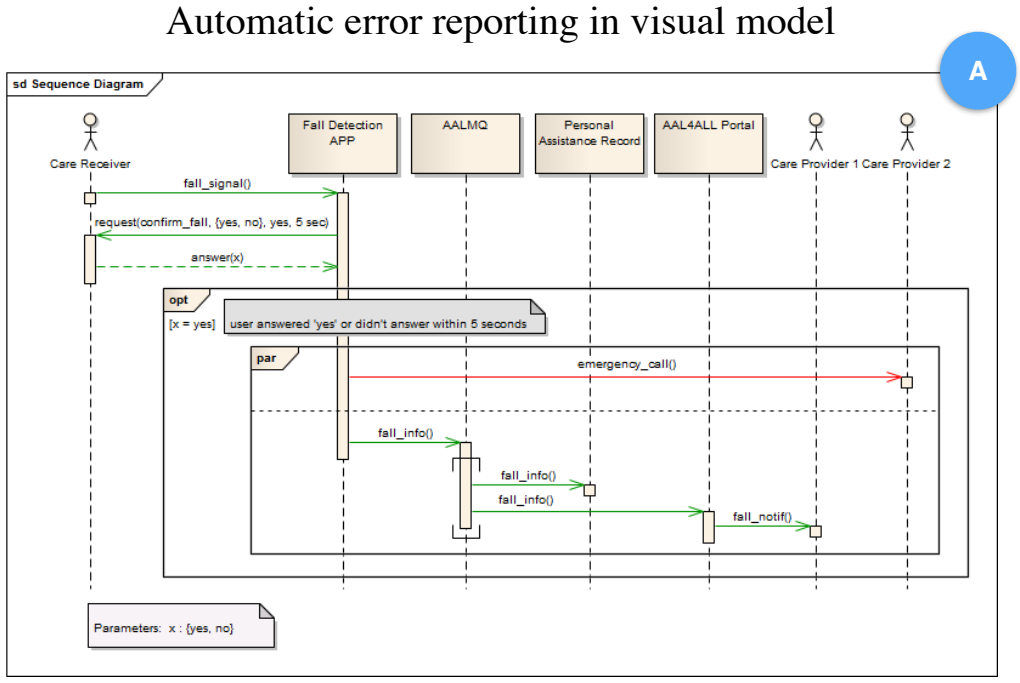
Main Ideas

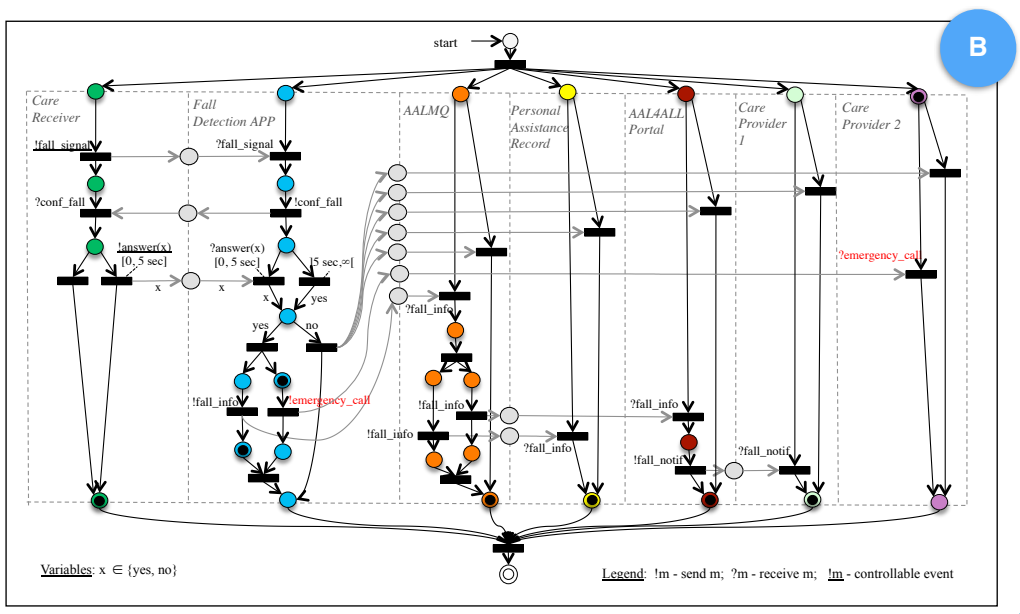
Two different modeling notations (front-end/back-end)

Automatic translation between both notations

Usage of an online, adaptive, test generation strategy

Automatic error reporting in visual model





Research Questions

RQ1 - Which extensions of Petri Nets should be used?

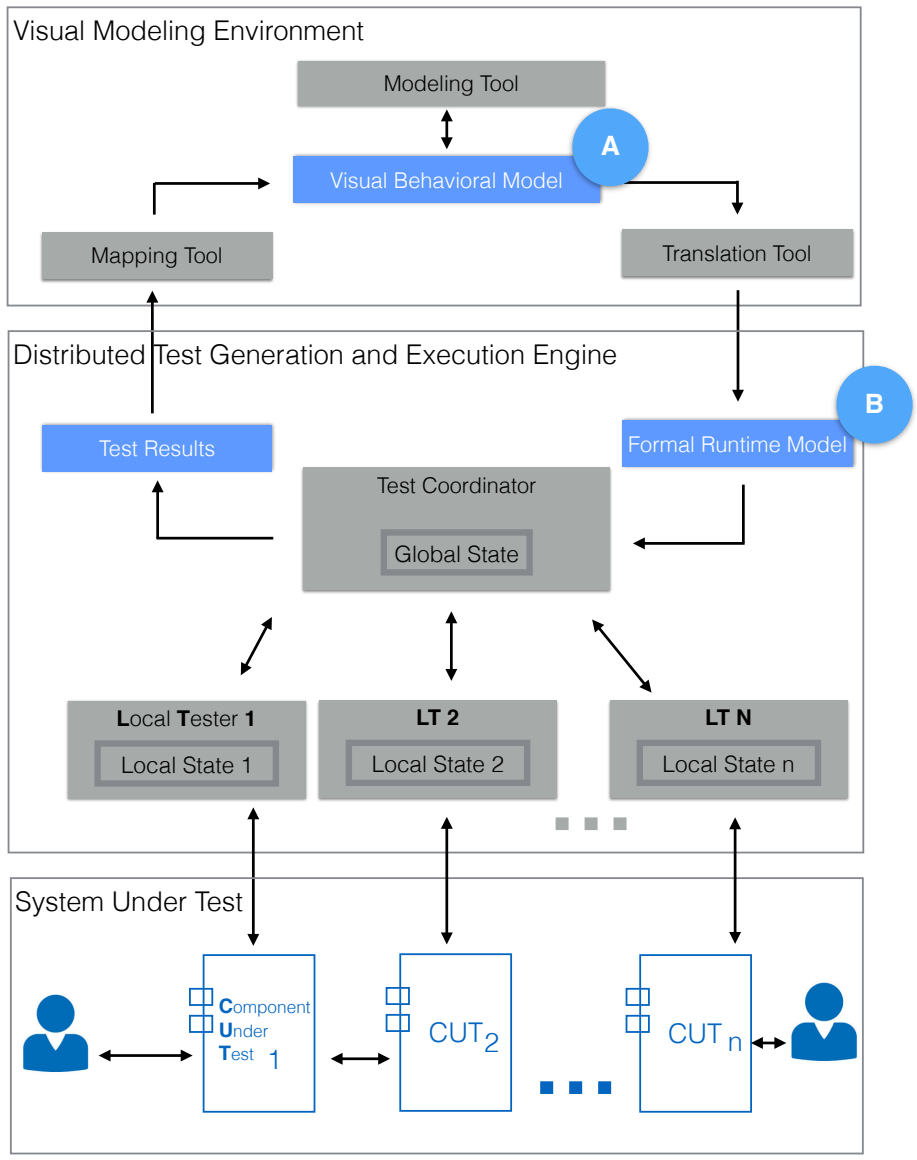
RQ2 - How to translate temporal constraints from UML SDs to Petri Nets?

RQ3 - How to partition the Petri Net for distributed execution?

RQ4 - How to choose the next test action in a distributed environment?

RQ5 - How to translate the results back to the UML SD?

Prototype Architecture



Usage of a hybrid test architecture

Validation

Validation will be performed by case studies, namely in the Ambient Assisted Living and Health domains

Expected Results

Translation Rules between UML Sequence Diagrams and Petri Nets

Distributed Test Generation and Execution Algorithms

Tool Prototype

Project "NanoSTIMA: Macro-to-Nano Human Sensing: Towards Integrated Multimodal Health Monitoring and Analytics/NORTE-01-0145-FEDER-000016" is financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

NORTE2020 PROGRAMA OPERACIONAL REGIONAL DO NORTE

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Europeu de Desenvolvimento Regional