Framework for Automated Functional Testing of P2P-based M2M Applications

Besfort Shala, Patrick Wacht, Ulrich Trick, Armin Lehmann
Research Group for Telecommunication Networks
Frankfurt University of Applied Sciences
Frankfurt/M., Germany
shala@e-technik.org

Besfort Shala, Bogdan Ghita, Stavros Shiaeles
Centre for Security
Communications and Network Research
University of Plymouth
Plymouth, UK

***Abstract*—This publication presents a novel concept for automated testing of decentralised services and applications in Peer-to-Peer (P2P) connected Machine-to-Machine (M2M) networks. Different challenges and requirements for testing are defined and a novel testing framework with a special testing architecture for functional testing is introduced. Furthermore, this publication describes a novel concept for deriving and generating test cases in M2M applications composed by several services.**

Keywords— M2M; P2P; Service and Application; Functional automated testing

I. INTRODUCTION

Building surveillance, energy management, traffic management, electro mobility and ambient assisted living are only a few Machine-to-Machine (M2M) application fields which are present nowadays. According to the European Telecommunications Standards Institute (ETSI), M2M applications are defined as "applications that run the service logic and use Service Capabilities accessible via open interfaces" [1]. Previous papers have defined requirements and concepts to realise service and application provision in M2M. The work and investigations of this research paper are based on the P2P-based M2M application (P2P4M2M) framework which offers new possibilities for applications, realised by several peers, independent of central instances or corporations [2].

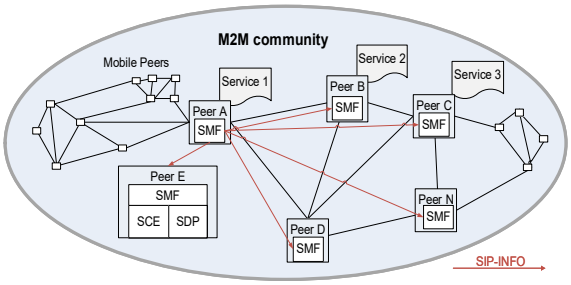

Fig. 1. P2P connected peers within a M2M community [2]

Reference [2] defines a framework that realises service and application provisioning using P2P networking in M2M application field. An application consists of one or more underlying services that are combined (i.e. aggregated or composed). Also, the use of the community concept described in [2] helps to avoid legal restrictions, adjust different interests among the peers and ensure optimisation and forming P2P networks. Fig. 1 shows the structure of the P2P connected peers within an M2M community based on [2]. Besides many advantages of the service provisioning concept, [2] does not consider approaches for testing P2P-based services/ applications in M2M and does not provide strategies to handle security risks. Therefore, a novel testing framework is required to enable testing of heterogeneous and decentralised services and applications in the P2P4M2M framework.

The aim of this paper is to illustrate the challenges and requirements of testing services and applications in P2P4M2M and to define a novel concept for functional automated testing. The testing methodology in this concept is based on model-based testing because of its advantages described in [3] when compared to other methodologies. For dealing with the distributed nature of services and applications in [2], this paper introduces a novel testing framework with a special testing architecture. The proposed testing architecture integrates a Test Generation Environment.

In order to show the importance of this research work, the following paper is structured into seven sections. After the introduction, section II presents an overview about the concept of service and application provisioning based on the P2P4M2M framework. Section III illustrates related work on testing approaches. Testing challenges and requirements are presented in section IV. Section V gives an overview about the principles of the proposed testing framework and describes the testing architecture and its elements. Section VI shows the test derivation and generation concept. At the end, section VII concludes with an application example related to the testing concept presented in this research.

II. P2P-BASED M2M APPLICATIONS

Reference [2] presents a concept of service and application provision in M2M. A service, as well as an application, can be realised by peers using technical or non-technical principles

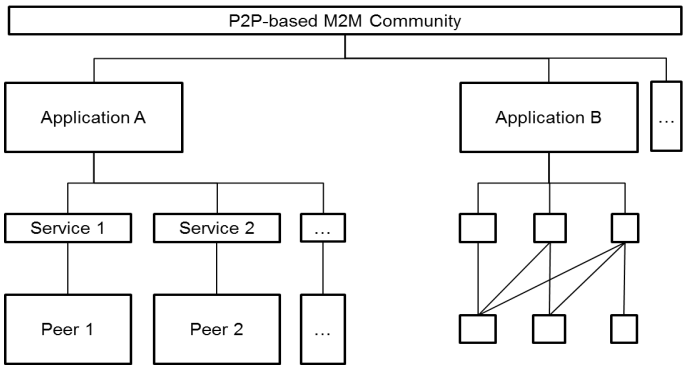

Fig. 2. Generalised structure of P2P4M2M framework [2]

(i.e. it can be provided using technical devices, e.g. computers, or by a human, e.g. personal assistance services). The services are realised by one or more service components which form the building blocks of services. The service components themselves are realised via several software applications executed on several execution environments. Peers are also represented by technical devices or humans (if applicable supported by technical devices) which are networked using P2P mechanism. The M2M community described in [2] forms a social network of peers where different sub communities are also used to address different application fields, interests and geographical locations. The networking enables the participating peers to provide a service that can be consumed by others [2].

Reference [4] introduces a Service Management Framework (SMF) installed in the local households, consisting of Service Delivery Platform (SDP) and Service Creation Environment (SCE), and uses the concept of P2P networked energy-community. The SCE brings the functionality to design and configure value-added services graphically, according to the personal needs of the users [4]. The SMF described in [4] is also the main component for service and application provisioning in M2M based on the P2P4M2M framework [2].

According to [5], the information exchange between the peers for the service utilisation and the signalling to generate the application is enabled by using various M2M communication protocols (e.g., CoAP, HTTP, SIP, MQTT) based on SUBSCRIBE/ NOTIFY principle. Fig. 2 shows a generalised overview of the P2P4M2M framework mentioned in [2].

III. Related work

To the author`s knowledge, there are no known studies about automated testing and securing services and applications within the P2P4M2M framework. Furthermore, only a very limited amount of related work exists on functional testing of P2P systems and M2M applications. In relation to the different aspects of testing some relevant publications are presented later in this paper.

It is crucial to define a suitable testing architecture for testing different services and applications based on the P2P4M2M framework. Our survey of the related literature shows some centralised approaches for testing distributed systems. Several publications [6-9] present testing architectures based on a coordinator and testers with focus on testing P2P functions and distributed systems. The coordinator inserts and controls several testers which run on different logical nodes. Also, the coordinator collects centrally information of the distributed System under Test (SUT), derives the test verdicts, observes and controls external and internal actions of SUT and has a global view on distributed SUT. The testers execute the test instructions received from the coordinator and control the volatility of single peers. The problems of these approaches are the single points of failure of the global tester and the low level of usability in large scale systems as they do not scale up to large numbers of peers (typical P2P system may have a high number of peers). Another problem is the non-efficient generation of test cases and the missing environment for test generation. There are also decentralised approaches for testing distributed systems such as in [7] who introduces distributed testers with the following functions: several operating tester components which process a global test case together. The behaviour of the testers is controlled by a test coordination procedure. Reference [10] also proposes a distributed test architecture without the use of a central coordinator and ensuring the coordination between the testers by introducing a distribution procedure of test sequences among the testers. The disadvantage of this approach is that the testers do not have a global view on the SUT, so they must synchronise each other by means of a test coordination which leads to a high effort for synchronisation events for coordination between the testers. Also, an effective controllability of the participating testing peers is missing due to the lack of a central authority.

The aim of model-based testing (MBT) based on [11] is the creation of one or more formal models from which test cases can automatically be generated and executed according to predetermined test criteria. According to [11], model-based testing includes at least one of the following aspects: test modelling and test generation from models. There are different models for the purpose of model-based testing introduced in literature such as: Statecharts [12], Finite State Machines (FSM) [13], Petri nets [14] and UML [15]. Reference [16] presents an automated functional testing approach which follows a model-based strategy using Statecharts notation for modelling the potential behaviour of a service. The approach used in [16] leads to an enormous amount of generated test cases and is also not applicable for distributed systems such as the P2P4M2M framework. A model-based and test-driven testing methodology in the IoT domain is introduced by [17] but the main focus is the semantical description of IoT services which are running centralised on an application server and this approach lacks the possibility for deriving tests for applications composed by fully distributed services.

IV. Challenges and requirements for testing applications in P2P4M2M

According to [18], testing is defined as “the process of analysing a software item to detect the differences between existing and required conditions and to evaluate the features of the software items”. The aim of this research is the testing of services and applications based on the P2P4M2M framework.

The process of creating M2M applications based on [5] makes functional testing very complex and can be described as follows: The application creator creates and configures an M2M application using his SCE. The M2M application consists of several services which are part of an M2M community. The services are described by their Service Interface Description. The Service Interface Description includes service ID, service functions, input, output and further configuration parameters of a service. Services are provided by different peers participating in a P2P network without the use of a central authority. The creation of an application will generate an SCXML (State Chart XML) description which precisely describes the potential functionality the application should deliver in a formal manner. In principle, such an SCXML description includes the involved services, the connection of services as well as conditions and definitions of input/ output parameters.

A special testing framework is required for testing the M2M application. First of all, the P2P4M2M framework is a distributed system and based on [19] distributed systems are "heterogeneous in terms of communication networks, operating systems, hardware platforms and also the programming language used to develop individual components". Reference [19] states that the size and complexity of distributed systems is growing and the system should be able to run over a wide variety of different platforms and access different kinds of interfaces. Considering the distributed characteristics of the P2P4M2M framework and the need for exchanging relevant information for testing it is important to ensure collaboration between the services, applications and test elements. The decentralisation of the peers and also their volatility (nodes leaving and entering suddenly) in the P2P-based M2M application community has to be considered in the test environment. Especially in the P2P4M2M concept, the application creator could be a user who has no technical background and who is not able to prepare the testing. Also, based on [11] the application creator should not be the test creator. If the application creator has already interpreted a specification incorrectly during the development, he will also misinterpret it for the test. For this reason and also for the advantages of test automation [11] the testing needs to be automated using a mechanism which utilises the information provided by the application and the services. Another problem is the procedure for defining test cases. Testing distributed services and applications in M2M networks requires different methods for deriving and generating test suites and for running the test. Reference [20] presents several problems for testing distributed systems including the test data generation and the specific execution behaviour. The testing framework must have the ability to derive test cases from the information gathered by the application and the distributed services. Based on the characteristics of the P2P4M2M framework presented in [5] the execution of the test cases on the participating services and the composed application should also be considered. Furthermore, there are several security issues based on [21] related to the P2P4M2M framework. Additionally, reference [21] introduces the concept of trust for P2P-based M2M applications and the integration of a Trust Management System (TMS) within the testing framework. Due to these different challenges, the general requirements for the testing framework can be summarised as follows:

- **Collaboration** – It is necessary to have collaboration between the application creator, the test environment and the peers, which are part of the application, and are providing or consuming services.
- **Deployment** – The testing framework needs to have the ability to deal with high number of peers and also the volatility of nodes in P2P network should be considered by the framework.
- **Test Automation** – Based on the complexity of the P2P4M2M framework the whole testing process needs to be automated considering the distributed architecture of the system.
- **Test Derivation** – Test suites need to be derived and generated from the gathered information about the composed M2M application and the participating services.
- **Test Execution** – The generated test cases need to be executed on different services in a timely manner. Also the test cases for the whole application should be executed after its creation.
- **Verification** – The testing process should deliver results about the functionality of the considered SUT, which could be a service or an application.
- **Tool support** – The framework should provide tools to generate, execute and manage tests.
- **P2P and M2M capability** – The framework should consider the included P2P mechanism and its characteristics within the application framework. M2M communication protocols should also be supported.
- **Trust Management System support** – The framework should provide the possibility to integrate a trust management system in its architecture.

V. PRINCIPLES OF PROPOSED TESTING FRAMEWORK

The challenges of testing (see section IV) and the complexity of the P2P-based M2M application framework leads to the necessity to define a suitable testing framework. The focus within this research work is the functional testing of services/ applications based on the P2P4M2M framework. Functional testing is the process of verifying the functions in a system to assure that they meet the specified requirement. Reference [22] defines that "every software system can be seen as a black box, where a tester selects valid and invalid inputs and determines the correct output" and in functional testing "a tester does not need to know the internals of the SUT as the focus is to evaluate the functional correctness of a given system, independently of its internal implementation".

Three black-box testing scenarios can be derived based on the application creation process described in [5]. The first scenario deals with the testing of a service after it enters the M2M community. This happens to ensure the availability of the correctly working services in the community and should be done after predefined time intervals. The second and third

testing scenarios will happen after the application creator builds an application using several services participating in the M2M community. The composed and created application needs to be tested based on its configuration and according to the special conditions of each participating service in the composition. An example for testing an application is provided later in this research paper. The services, which are part of the composed application, need to be tested according to their special configurations within the application.

Based on the related work [6-10] [15-16], the requirements for the testing framework for P2P-based M2M applications and the need for a load balanced and effective testing mechanism, a test architecture with a combination of a global tester called Test Master and distributed testers called Test Agents is presented in this work. An additional test component for load balancing is required due to the increasing number of participating peers and provided services in the P2P-based M2M community and the inability of the Test Master to scale up with the increasing number of distributed services. Therefore, a test generation environment is included in the testing framework which derives and generates test cases and also interacts with the Test Master, the services and the application creator. Fig. 3 shows the conceptual test architecture consisting of a Test Master, Test Agents and Test Generation Environment (TGE).

The TGE gets an SCXML description of a composed application and also the service interface descriptions of each participating service and generates test cases for this application respectively each of the participating services. Afterwards, the TGE will send the relevant test instructions to the Test Master who is the coordinator of the overall testing framework. The Test Master will send test instructions to the Test Agents, who will afterwards execute the test cases on the SUT. Systems under Tests are all services which are part of the community and the composed application. Table I gives an overview about the different functions of the test components.

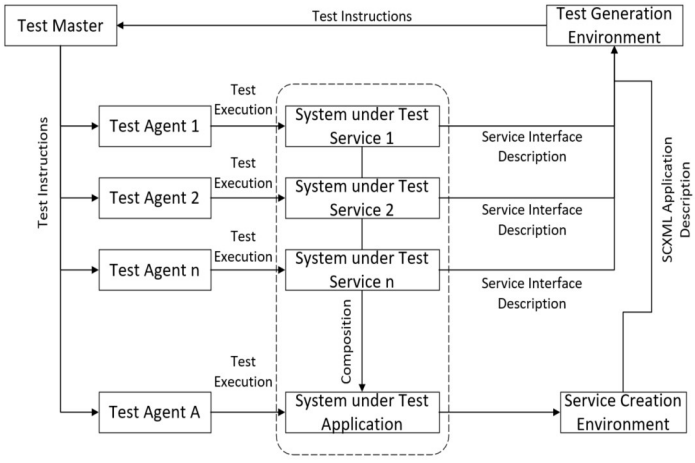

Fig. 3. Conceptual Test Architecture of the P2P4M2M framework

TABLE I. TEST ELEMENTS OF THE TESTING FRAMEWORK

Test Element	Functionality
Test Master	• controlling test processes • managing test processes • receiving test instructions from Test Derivation Environment • receiving Test Results from the Test Agent • evaluating Test Results • providing the Application Creator with information about the test results • providing the service providers with information about the test results • sending test instructions to the test agents • interacting with all test elements • maintaining list of test agents
Test Generation Environment	• receiving Service Interface Descriptions from the services • receiving SCXML Descriptions from the Application Creator • deriving and generating test cases • sending test instructions to the Test Master
Test Agent	• receiving instructions from the Test Master • executing test instructions on SUT • sending test results to the Test Master • exchanging test related information with the Test Master

VI. PROPOSED TEST GENERATION ENVIRONMENT

As mentioned in the previous chapters, the behaviour of the services and applications based on the P2P4M2M concept are described by their SCXML descriptions. Each service and application in the P2P4M2M framework is considered as a system and the behaviour modelling using SCXML will build a so-called system model. Reference [23] defines system models as a tool for describing the composition and interaction of components in the system. System models are useful for testing because they provide general information about the functionality of a system. Another aim of this research is to generate a test model from the system model and the information it provides. Then, this test model generates test cases for executing them on the system. Furthermore, the challenge of this publication is to develop a mechanism for transforming the system model to a test model.

In this work the task of test derivation and generation is carried out by the Test Generation Environment (TGE) which must know how the different services work together. The TGE must know the logic of the services/ application in order to derive and generate tests accordingly. This logic has to be derived from the SCXML descriptions of the application and the service interface descriptions of the participating services. The workflow of the service and application creation and testing is shown in Fig. 3. The special challenges for deriving test cases in the P2P-based M2M application framework are: Different services; services providing different functions; application built by the composition of different services; created SCXML description of an application that does not provides the full information about the full functionality of each service; filtering the relevant information from the SCXML descriptions.

Reference [24] presents a Test Creation Framework which allows test developers to automatically create and execute test cases. There are some major drawbacks in the framework presented in [24]. First, there is no possibility for filtering the relevant information of the distributed system models and building a relevant test model. Second, [24] introduces the role of the test developer whereas the testing framework presented in this research aims to be fully automated without the use of a test developer. In order to fill the gaps of [24] and to fulfil our requirements of test derivation and test automation, we integrate the special component of the Specification Collector which will be explained later. With respect to the above described challenges, a novel Test Generation Environment concept is introduced and shown in Fig. 4. This environment consists of four different components which is based on model based testing (MBT) and is explained below.

- **Specification Collector Unit (SCU)** – Collects the information received from the application creator and the services participating in the composed application. After receiving this information the SCU has to filter the relevant information and to build a so called Test Application Description (TAD) which afterwards will be sent to the Behaviour Model Generator.
- **Behaviour Model Generation Unit (BMGU)** – From the received TAD the BMGU has to generate extended behaviour models based on behaviour notation languages such as: EFSM, Statechart, BPEL, UML etc. The models describe the behaviour of the composited application and each of the participating services.
- **Test Suite Derivation Unit (TSDU)** – After getting the behaviour models the TSDU will derive possible test cases respectively test suits for the created P2P-based M2M application and will send the derived abstract tests suits to the Test Suite Generation Unit.
- **Test Suite Generation Unit (TSGU)** – Depending on the test execution environment the TSGU will generate from the received abstract tests suits the executable test suits and will forward these information to the Test Master.

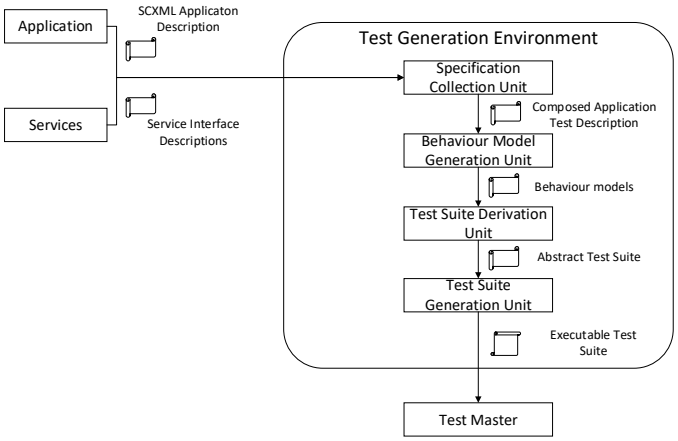

Fig. 4. Detailed view of the Test Generation Environment

VII. Application Example

To prove the concept of testing P2P-based M2M applications provided by the P2P4M2M framework, an example application is introduced in the following. The major idea behind the application called "Temperature Surveillance" is to allow consumers to continuously get informed about the temperature in certain rooms, e.g. via their smartphones. The application requires the involvement of three different services which exchange information by using SIP SUBSCRIBE and NOTIFY messages. Service 1 delivers the temperature values by accessing diverse temperature sensors in the environment whereas service 2 evaluates the received values from service 1 and determines the consumers (via SIP URIs) who should receive the values. The role of service 3 is to forward the received temperature values via SIP instant messages to the consumers. Based on this example, the proposed Test Generation Environment generates a test execution architecture composed of one so-called Master Test Component (MTC) and several Parallel Test Components (PTC). Both the terms MTC and PTC are derived from typical Testing and Test Control Notation 3 (TTCN-3)-based environments [25]. The TTCN-3 concept also allows to define a distributed test execution environment where all test components (MTC, PTC) are running on different machines. MTC and PTCs are the corresponding Test Master and Test Agents presented in section V. This aspect is very relevant for testing distributed applications running in the P2P4M2M framework. Specifically for the "Temperature Surveillance" application, four PTCs are used (see Fig. 5). The number of PTCs depends on the functionality provided by the services that are involved to provide the application functionality. For both service 1 and 2, it is sufficient to let only one PTC (PTC 1 for service 1, PTC 2 for service 2) participate in the test execution. For service 3, a random number of PTCs is required as the number of consumers is depending on the number of registered SIP URIs. An example test case verifying the main functionality of the application would be executed as follows: First, PTC 1 subscribes service 1 to continuously receive temperature values via NOTIFY messages. As soon as PTC 1 receives new values, they are forwarded to all the other PTCs via the MTC to compare them later in the test execution. Second, PTC 2 subscribes to service 2 to receive the temperature values and the SIP URIs of the consumers. PTC 2 will also verify if the temperature values received by PTC 1 match with the ones PTC 2 received from service 2. If the values are not equal, the test case will be directly declared as "fail". If the values are equal, the test case execution continues with service 2 forwarding the SIP URIs and the values to PTC 3 and PTC 4. Both PTCs will then realise a registration process with the SIP URIs to be able to receive the SIP instant messages with the temperature values by service 3. As soon as PTC 3 and PTC 4 received the messages, the values included will also be compared with the values received from PTC 2. After the test case execution is finished, an overall verdict (such as "pass", "fail" or "inconclusive") is assigned. Based on the executable test suite (see Figure 4) for this specific application example, further test cases can then be executed using the identical test architecture.

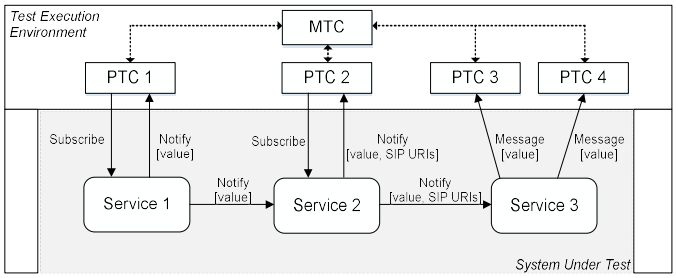

Fig. 5. Testing the M2M Application "Temperature Surveillance"

VIII. CONCLUSION

This publication presents a novel concept for automated functional testing of P2P-based services and applications in M2M. The presented concept aims to deal with the complexity of testing applications which are composed by several heterogenic services with different service functionalities and configuration parameters. The missing role of a test creator in the P2P-based M2M application framework and the automation of test case generation using the integration of the Test Generation Environment are solved by the presented testing framework.

The next step is to develop a Test Application Description which includes relevant information for automated testing derived from the composed application and the participating services in the composition based on the P2P4M2M framework. Furthermore, the possibilities to distribute the test elements using P2P mechanism and to build a P2P test community are part of future steps which have to be evaluated. We are simultaneously working on a concept of integrating a trust management system inside the presented testing framework for ensuring security and trustworthiness in P2P-based M2M applications using trust.

ACKNOWLEDGEMENTS

The research project P2P4M2M providing the basis for this publication was partially funded by the Federal Ministry of Education and Research (BMBF) of the Federal Republic of Germany under grant number 03FH022IX5. The authors of this publication are in charge of its content.

REFERENCES

[1] ETSI TR 102 725, V1.1.1, 2013-06: Technical Report, "Machine-to-Machine communications (M2M); Definitions", ETSI TISPAN

[2] M. Steinheimer, U. Trick, W. Fuhrmann and B. Ghita, "P2P-based community concept for M2M Applications," Proc. of Second International Conference on Future Generation Communication Technologies (FGCT 2013), London, UK, December 2013

[3] P. Wacht, "Framework for Automated Functional Tests within Value-Added Service Environments", PhD Thesis, School of Computing and Mathematics, University of Plymouth, UK, December 2015

[4] M. Steinheimer, U. Trick, P. Ruhrig, R. Tönjes, M. Fischer and D. Hölker, „SIP-basierte P2P-Vernetzung in einer Energie-Community", ITG-Fachbericht 242: Mobilkommunikation, pp. 64, Mai 2013

[5] M. Steinheimer, U. Trick, B. Ghita and W. Fuhrmann, "Decentralised System Architecture for autonomous and cooperative M2M Application Service Provision", International Conference on Smart Grid and Smart Cities (ICSGSC), in press, 2017

[6] E. Almeida, G. Sunye, Y. Traon and P. Valduriez, "A framework for testing peer-to-peer systems," Dans 19th International Symposium on Software Reliability Engineering (ISSRE 2008), Redmond, Seattle, USA, IEEE Computer Society, 2008

[7] A. Ulrich and H. Konig, "Architectures for testing distributed systems," Dans Proceedings of the IFIP TC6 12th International Workshop on Testing Communicating Systems, pp. 93–108, Deventer, The Netherlands. Kluwer, B.V., 1999

[8] P. Rosenkranz, M. Wählisch, E. Baccelli and L. Ortmann, "A Distributed Test System Architecture for Open-source IoT Software," IoT-Sys´15 Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems, pp. 43-48, ACM, New York. 2015

[9] C. Torens and L. Ebrecht, "RemoteTest: A Framework for Testing Distributed Systems," 2010 Fifth International Conference on Software Engineering Advances (ICSEA 2010), pp. 441-446, Nice, France, 2010

[10] A. Khoumsi, "Testing distributed real time systems using a distributed test architecture", Sixth IEEE Symposium on Computers and Communications, 2001

[11] M. Winter, T. Roßner, C. Brandes and H. Götz, "Basiswissen modellbasierter Test", dpunkt Verlag Heidelberg, Germany, ISBN: 978-3-86490-297-0, 2016

[12] D. Harel, "Statecharts: A Visual Formalism for Complex Systems," Science of Computer Programming, 8(3):231–274, June 1987

[13] G. Booch, J. Rumbaugh, and I. Jacobson, "The Unified Modeling Language user guide," Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1999

[14] W. Reisig, "Petri Nets: An Introduction," Springer-Verlag New York, Inc., New York, NY, USA, 1985

[15] The Unified Modeling Language 2.0: Superstructure FTF convenience document (ptc/04-10-02). Object Management Group, 2004. Available at:www.uml.org

[16] P. Wacht, U. Trick, W. Fuhrmann and B. Ghita, "Efficient Test Case Derivation from Statecharts-Based Models", Proceedings of the Eleventh International Network Conference, Frankfurt, Germany, 2016

[17] D. Kuemper, E. Reetz and R. Tönjes, "Test Derivation for Semantically Described IoT Services", Future Network and Mobile Summit (FutureNetworkSummit), pp. 1-10, IEEE, 2013

[18] IEEE Std 610.12, (1990), IEEE Standard, "IEEE Standard Glossary of Software Engineering Terminology", IEEE

[19] A. Saifan and J. Dingel, "Model-based testing of distributed systems," Technichal report, vol. 548, 2008

[20] S. Ghosh and A. Mathur, "Issues in testing distributed component-based systems," In Proceedings of the First International Conference on Softaware Engineering Workshop on Testing Distributed Component-Based Systems, Los Angeles, CA, May 1999

[21] B. Shala, P. Wacht, U. Trick, A. Lehmann, B. Ghita and S. Shiaeles, "Ensuring Trustworthiness for P2P-based M2M applications," 7th International Conference on Internet Technologies & Applications (ITA), in press, 2017

[22] M. Pezzè and M. Young, "Software testen und analysieren" (translated title: "Testing and analysing software"), Oldenbourg, Munich, Germany, ISBN: 3-486-58521-6. 2009

[23] I. Schieferdecker, "Modellbasiertes Testen," OBJEKTspektrum 3/07, pp. 39-45. 2007

[24] P. Wacht, U. Trick, W. Fuhrmann and B. Ghita, "A Novel Test Creation Framework for Value-Added Services," 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, 2016

[25] EG 201 873-1: Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part1: TTCN-3 Core Language. ETSI, September 2008