Test-Enhanced Life Cycle for Composed IoT-Based Services

Daniel Kuemper, Eike Steffen Reetz, Daniel Hölker, Ralf Tönjes

Test-Enhanced Life Cycle for Composed IoT-Based Services

Daniel Kuemper, Eike Reetz, Daniel Hölker, and Ralf Tönjes

University of Applied Sciences Osnabrück, P.O Box 1940, 49009 Osnabrück, Germany
{d.kuemper,e.reetz,d.hoelker,r.toenjes}@hs-osnabrueck.de

Abstract. Major challenges in developing services for the Internet of Things (IoT) are based on heterogeneous interfaces and radio technologies. This paper proposes a knowledge driven service life cycle, which enables a structured utilisation of semantic descriptions for re-usability and testing. Furthermore the approach facilitates the process of encapsulating IoT resources into services.

Keywords: SOA, IoT, Test, Life Cycle, Composition, Deployment

1 Introduction

Scalability and interoperability are key challenges in the composition of Internet of Things (IoT) applications by reusing existing IoT resources like sensors and actuators. The IoT.est project aims at surmounting available silo architectures with varying, partly proprietary interfaces by re-using Service-Oriented Architecture (SOA) approaches to ensure compliance, scalability and the ability to test and monitor IoT services. The main approach is to encapsulate the communication with heterogeneous IoT resources into Representational State Transfer based (RESTful)[1] services that are linked to an extensive, formal description of their functional and non-functional capabilities. The encapsulation prevents the need of re-implementing proprietary IoT resource interfaces for every invocation and enables the usage of widespread clients, proxies and analysis tools. The service descriptions, stored in a semantic knowledge management, ensure re-usability and the ability to find services for composition by providing extensive information about their in- and outputs as well as their Quality of Service (QoS) and Quality of Information (QoI). The service descriptions are also used to facilitate the (semi-) automated derivation of test cases[2] to ensure service reliability.

This work assumes that a common understanding of the service life cycle is crucial in order to build successful IoT-based services [3][4]. To ensure a knowledge driven service composition and testing approach the annotation process of the service becomes eminent. While previous life cycle approaches like the classical V-Model or agile programming such as Extreme Programming have already considered techniques like test-first [5] and test-driven development they did not explicitly describe the process of knowledge annotation. The proposed life cycle approach overcomes these limitations by adding a clear view of knowledge representation, annotation and the process of test derivation from this knowledge.

In this project the creation and composition of new services is guided by a service life cycle framework that is aligned to support a consistent workflow[6]. The service life cycle employed in the IoT.est project is outlined in the following sections towards a brief definition of the utilised service model.

1.1 IoT.est Service Model

This work utilises RESTful interfaces to encapsulate IoT services for enhanced reusability. It defines two types of services to ensure direct usage and composition of IoT services without dealing with heterogeneous interfaces:

The **Atomic Service (AS)** is a RESTful web service, accessing $0 - n$ IoT resources via their own individual interfaces and radio technologies. It enables access to these resources via standardised `Get, Post, Put, Delete` request methods, whose invocation is defined in a dedicated Web Application Description Language (WADL) document[7]. Input parameters as well as response documents of these methods are extensively described in the semantic knowledge management, to e.g. identify a specific service parameter not just as a *double* but rather as a *temperature/Celsius* value. The implemented AS can be deployed to a run time environment for web services and is registered in the knowledge management.

The **Composite Service (CS)** enables a business process-based composition of various AS and CS. It also provides a RESTful interface for service invocation and does not directly connect to IoT resources using their proprietary interfaces. It only uses AS and CS interfaces to acquire sensor information and to control actuators. The interfaces are also described by WADL and a semantic description to enable reusability.

2 Life Cycle Phases

The documentation of life cycle iterations is stored in the projects knowledge management component. The following sections describe the different phases and stages of the life cycle that can be followed in Fig. 1. Stages are represented by circles. Short arrows show typical stage succession. Larger curved arrows show shortcuts, which occur if needed amendments are identified.

2.1 Modelling Phase

Stage A: Identification / Adaptation of the Business Process

Description: Initial stage to identify the goal of a new service from a business perspective. Adaptation during further iterations of the life cycle due to enhanced requirements or technical restrictions. Business process adaptation leads to a new version of the service where older versions can independently be developed further.

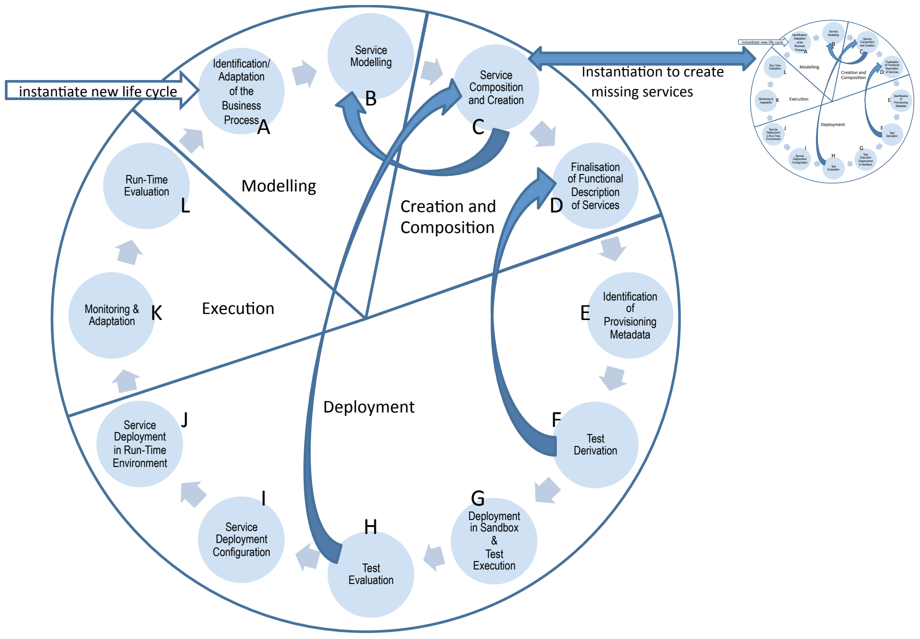
instantiate new life cycle
Identification/ Adaptation of the Business Process A
Service Modelling B
Service Composition and Creation C
Instantiation to create missing services
Finalisation of Functional Description of Services D
Identification of Provisioning Metadata E
Test Derivation F
Deployment in Sandbox & Test Execution G
Test Evaluation H
Service Deployment Configuration I
Service Deployment in Run-Time Environment J
Monitoring & Adaptation K
Run-Time Evaluation L
Modelling
Creation and Composition
Deployment
Execution

Fig. 1. Phases and Stages of the Service Life Cycle

Outcomes: A human readable description of the functionality/goal of the service (e.g. for an atomic service: providing the heart rate of patients; for a composite service: providing monitoring and alarm functionality for patients with cardio-vascular issues). This description also includes non-functional capabilities like QoS requirements derived from the desired user experience and estimated usage.

Stage B: Service Modelling

Description: Identification of a technical realisation to achieve the goal of stage A.
Outcomes: Abstract view of an engineer on how to solve the problem using services (Block diagram, data flow, event-based behaviour, process description).

2.2 Creation and Composition Phase

Stage C: Service Composition and Creation

Description: Lookup for services that provide the required functionality to achieve the goal defined in Stage A/B. Composition of available services to create a composite service (described in Business Process Model and Notation (BPMN)). Initiation of the creation of a non-available atomic or composed service by instancing a new life cycle.

Decision: Is the proposed service model feasible with available resources and can missing services be created or composed? Not possible: stage B; If new atomic services are needed and new life cycle instance for them is created: stage A (of new life cycle). After new creation or if the creation/composition is possible: stage C.
Outcomes: If an AS is created: an implemented service that can be deployed to a run time environment. If a CS is composed: the available services required for the composition and the BPMN implementation of the service logic exists an is executable for testing.

Stage D: Finalisation of Functional Description of Services

Description: After the description of raw interfaces and data types, included during development via annotations, a semantic description of in- and output parameters and the stateful description of the service behaviour has to be added. The semantic description enables the developer to explicitly specify used parameters that can be reused for composition and test derivation.
Outcomes: The full semantic description of the service (functional features and expected QoS) and the service interface description (WADL).

2.3 Deployment Phase

Stage E: Identification of Provisioning Metadata

Description: Based on the service description the resources for deployment ensuring functionality, connection to the IoT resources and QoS have to be determined to select a sufficiently equipped runtime environment.
Outcomes: Provisioning metadata - agreed by the service developer and the service provider. Platform independent deployment descriptor.

Stage F: Test Derivation

Description: The derivation of a System Under Test (SUT) and test cases is generated. Semantic information is used to narrow down the variety of test cases. E.g. the parameters of a service accepting latitude/longitude coordinates as input to deliver data are not simply modelled as two `float`-values. They will be described with an upper ontology as geo-coordinates each with possible value range $\{-90.0°, 90.0°\}/\{-180.0°, 180.0°\}$. With this information valid values for tests are clearly specified.
Decision: If service description is conflicting or insufficient the service life cycle will return to stage D.
Outcomes: Set of all executable test cases as Testing and Test Control Notation Version 3 (TTCN-3)[8] code.

Stage G: Deployment in Sandbox & Test Execution

Description: Based on stage E a sandbox environment is selected to execute a set of test cases created in stage F. The IoT resources are emulated during testing[9].
Outcomes: Test results (TTCN-3 log files) are stored in the knowledge management to ensure detailed evaluation of failed test cases.

Stage H: Test Evaluation

Description: A test developer analyses the results of the test case execution.
Decision: If errors occur: report to stage C to redevelop the service. If service behaves like expected, go on to stage I.
Outcomes: Go/No-go decision made by the test developer based on the resulting log files.

Stage I: Service Deployment Configuration

Description: After the service has been tested, one or more run time platforms are selected for service provisioning.
Outcomes: Final form of the provisioning metadata and deployment descriptor (platform dependent - manifest file).

Stage J: Service Deployment in Runtime Environment

Description: The deployment on the final run time platform is made.
Outcomes: Deployable package (platform dependent e.g. `war`) deployed in runtime. Configured and instantiated service. The service is running and available for monitoring.

2.4 Run Time Phase

Stage K: Monitoring & Adaptation

Description: While the service is running it is an on-going process to monitor service availability and functionality. In the IoT.est Framework CSs are not implicitly bound to specific service endpoints of ASs. If an alternative AS exists, which can deliver the same information, the CS is able to utilize this AS as service endpoint. Due to altering infrastructure conditions this stage monitors service behaviour and performs necessary actions to adapt the composite service infrastructure. Furthermore the monitoring delivers non functional service quality and load information that can be utilized during next iterations of the service life cycle.
Outcomes: Report of service usage and availability, adaptation, QoS adherence, occurred malfunctions and alerts.

Stage L: Run Time Evaluation

Description: The evaluation of stage K reports is used for further life cycle iterations. Furthermore the enquiry of user experience is used to advance the next service versions.
Outcomes: Documentation of suggestions for service or runtime changes. E.g. the replacement of runtime or redevelopment of service if QoS issues occur.

3 Conclusion

The proposed SOA approach lowers the obstacle for integrating IoT resources into existing service architectures and shows a far-reaching applicability in the IoT domain. In combination with a structured development using a service life cycle, which is utilising semantic descriptions, it ensures re-usability of services and facilitates (semi-) automatic test derivation and composition. Next steps will be the automated monitoring and Service Composition Environment (SCE) based tracking of the service life cycle.

Acknowledgments. The research leading to these results has received funding from the European Union Seventh Framework Programme for the IoT.est project under grant agreement n° 257521. `http://www.ict-iotest.eu`

References

1. Roy Thomas Fielding, R. T.: Architectural Styles and the Design of Network-based Software Architectures, University of California (2000)
2. Kuemper, D., Reetz, E.S., Toenjes, R.: Test Derivation for Semantically Described IoT Services. In 22nd Future Network and Mobile Summit, Lisbon (2013)
3. Cabral Pinto, F., Chainho, P., Pssaro, N., Santiago, F., Corujo, D., Gomes, D.: The business of things architecture. Transactions on Emerging Telecommunications Technologies, Wiley (2013)
4. Raverdy, P. G., Autili, M., Bertolino, A., Cordier, C., Bhushan, B., Ords, I., Devlic, A.: Service Lifecycle Management, Mobile Service Platforms (MSP) cluster of IST FP6 projects, White Paper, (2008)
5. Andrea, J.: Envisioning the next-generation of functional testing tools. IEEE Software (2007)
6. Reetz, E. S. , Kuemper, D. , Lehmann, A., Toenjes, R.: Test Driven Life Cycle Management for Internet of Things based Services: a Semantic Approach. In VALID 2012, The Fourth International Conference on Advances in System Testing and Validation Lifecycle (pp. 21-27), Lisbon (2012)
7. Hadley, M.J.: Web application description language (WADL), Technical report, Sun Microsystems (2006)
8. ETSI, EG 201 873-1: Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part1: TTCN-3 Core Language (2012)
9. Reetz, E., Kuemper, D., Moessner, K., Toenjes, R.: How to Test IoT Services before Deploying them into Real World. In 19th European Wireless Conference (EW2013), Guildford (2013)