Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 57 (2016) 219 – 223

www.elsevier.com/locate/procedia

49th CIRP Conference on Manufacturing Systems (CIRP-CMS 2016)

Agile Service-oriented Analysis and Design of Industrial Internet Applications

Thomas Usländer[a]*

[a]*Fraunhofer IOSB, Fraunhoferstr. 1, 76131 Karlsruhe, Germany*

* Corresponding author. Tel.: +49-721-6091-480; fax: +49-721-6091-413. *E-mail address:* Thomas.uslaender@iosb.fraunhofer.de

Abstract

This paper highlights the importance of systematic agile service engineering for Industrial Internet and Industrie 4.0 software applications. Here, the functional and non-functional requirements of IT users (mostly from the disciplines of mechanical engineering and electrical engineering) need to be mapped to the capabilities of emerging service platforms. The capabilities of service platforms are usually described, structured and formalized by software architects. However, very often, their description does not fit to the language of the user. This complicates the transition from requirements analysis to system design. The paper describes how this 'gap' may be closed with help of a service-oriented analysis and design (SOAD) methodology entitled SERVUS and a corresponding Web-based collaborative tool that supports the documentation according to the SERVUS design methodology. SERVUS denotes a Design Methodology for Information Systems based upon Service-oriented Architectures and the Modelling of Use Cases and Capabilities as Resources. It describes individual design activities interconnected by a common modelling environment (based upon the REST principles) combining several viewpoints of architectural reference model, e.g. the RAMI4.0 Business, Information and Functional Layers. SERVUS was successfully used in numerous collaborative and inter-disciplinary software projects.

Peer-review under responsibility of the scientific committee of the 49th CIRP Conference on Manufacturing Systems

Keywords: Service-oriented analysis and design; SOAD; Industrie 4.0; service engineering; SERVUS; use cases

1. Motivation

According to the Industrial Internet Reference Architecture (IIRA) [1], the Industrial Internet is an "Internet of things, machines, computers and people, enabling intelligent industrial operations using advanced data analytics for transformational business outcomes". Industrial Internet systems cover multiple applications domains, e.g. energy, healthcare, manufacturing, public sector, transportation and related industrial systems. The IIRA requests that they "must be easily understandable and supported by widely applicable, standard-based, open and horizontal architecture frameworks and reference architectures that can be implemented with interoperable and interchangeable building blocks". The German initiative Industrie 4.0 basically complies with these objectives but focuses on industrial production. As a starting point, a Reference Architecture Model Industrie 4.0 (RAMI4.0) was published [4]. RAMI4.0 aims at providing a kind of 'coordinate system' for all element types in an Industrie 4.0 system architecture. However, it does not encompass the process how to engineer software applications in an Industrie 4.0 environment. This aspect is also missing in the IIRA up to now. This paper aims at contributing to these architecture models by adding the service engineering aspects to them.

Today, software and service engineering is performed in agile manner in order to increase flexibility during the software design and development and to have milestones to re-focus the process. However, agility needs to encompass the requirements analysis phase, too. The reason is that, only in rare cases, the requirements are fully available and fixed when software design and developments starts. In contrary, especially in the domain of factory automation systems, requirements analysis is an agile process including a multi-step dialogue between the user(s), the stakeholders and the software architects who know about the technological capabilities and constraints, and who may also estimate the effort to realize the expectations of the user. The resulting discussion often leads to reconsiderations and/or refinements of the user requirements. If multiple users of one

Peer-review under responsibility of the scientific committee of the 49th CIRP Conference on Manufacturing Systems
doi:10.1016/j.procir.2016.11.038

or even multiple organizations are involved, such dialogues are usually carried out during requirements analysis workshops facilitated by experienced systems analysts or architects. The crucial aspects are a solid methodology underpinning this process as well as an associated flexible documentation of the requirements during this process, also taking into account capabilities and constraints of underlying production and manufacturing IT architectures.

This paper presents a Web-based collaborative tool that supports the documentation according to the SERVUS design methodology [3].

The paper is structured as follows: Section 2 provides an overview about service-oriented analysis and design and the SERVUS Design Methodology, before the structure of the use case model, i.e. its meta-model, is presented in section 3. The architecture of the tool supporting the methodology in form of a so-called use case server is described in section 4. As an illustrative example, section 5 explains how it is used in Industrie 4.0 requirements analysis, before the paper concludes in section 6 with references to other tool applications and ideas about future developments in the RAMI4.0 context.

2. Service-oriented analysis and design (SOAD)

Numerous methodologies for service-oriented analysis and design were described in the literature, and were partly embedded in software development tools. The deficiency today is, that there is no well-established methodology that brings together the requirements and the expert knowledge of thematic users with the services and information offerings of service platforms, and, in addition, explicitly obeys the guidelines and constraints of architectural frameworks such as RAMI4.0 as side-conditions (see fig. 1). The SERVUS design methodology described in this paper proposes to fill this gap.

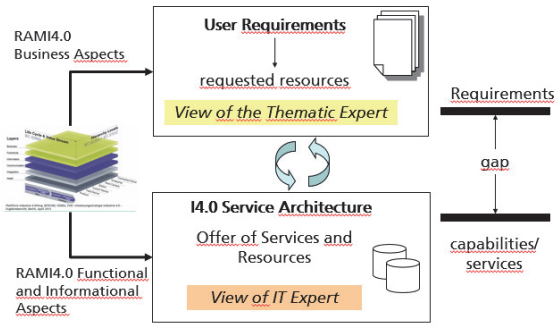

Fig. 1: Agile Service Engineering forIndustrial Internet Applications

2.1. SERVUS Design Methodology

The SERVUS design methodology [1] describes individual design activities interconnected by a common modelling environment that is common to the Enterprise, Information and Service Viewpoint of the ISO Reference Model of Open Distributed Processing (RM-ODP) [5], or the RAMI4.0 Business, Information and Functional Layer [4], respectively. The common modelling environment follows a resource-oriented approach according to the Representational State Transfer (REST) architectural style [6]. Hereby, a resource is considered to be an information object that is uniquely identified, may be represented in one or more representational forms (e.g. as a diagram, XML document or a map layer) and support resource methods that are taken from a limited set of operations whose semantics are well-known (uniform interface). A resource has own characteristics (attributes) and is linked to other resources forming a resource network. Furthermore, resource descriptions may refer to concepts of the domain model (design ontology) using the principle of semantic annotation, yielding so-called semantic resources

SERVUS considers requirement analysis to be part of the ISO RM-ODP Enterprise Viewpoint which largely corresponds to the IIRA Business Viewpoint or the RAMI4.0 Business Layer, respectively.

2.2. Use Cases

Requirements analysis results in use case models that describe the behavior of a system [7]. According to [7] “a use case is a sequence of actions performed by the system to yield an observable result that is typically of value for one or more actors or other stakeholders of the system”.

When designing an IIoT system application, a use case expresses the functional, informational and qualitative requirements of a user (i.e. an actor or a stakeholder) with respect to the system. Usually, use cases do not describe the user interactions themselves. The requirements for the user interface, i.e. a description of how the system functions are accessed by and presented to the user depending on his end-user device, may be captured in separate task models and then mapped to use cases [8]. It is essential for the use case description that the level of abstraction, the type of formalism as well as the language should be such that it is adequate to the domain of expertise of the users. In order to serve as a kind of contract with the user, a use case shall be both understandable to the user but also precise enough. Very often this means that use cases shall be specified in a non-technical way, normally achieved using plain text in natural language. However, in order to reduce the ambiguities and impreciseness of descriptions in natural language, structured textual descriptions are preferred. This means that use case descriptions are structured according to a given template, e.g. an application form comprising identifier and description fields or thematic domain references associated with code lists.

The SERVUS design methodology recommends a semi-formal description of use cases, e.g. according to the template proposed by Cockburn [9], but extends it in order to include references to requested resources, e.g. a time-series of water gauge values represented as a diagram. Optionally, it may be accompanied by a UML use case diagram.