A Process Framework for Designing Software Reference Architectures for providing Tools as a Service

Muhammad Aufeef Chauhan[±1,3], Muhammad Ali Babar[±1,2], Christian W. Probst[3]

[±]CREST-Centre for Research on Engineering Software Technologies[1,2]
[1]Software and Systems Section, IT University of Copenhagen, Denmark
[2]The University of Adelaide, Australia
[3]Formal Methods Section, Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark

muac@itu.dk, ali.babar@adelaide.edu.au, cwpr@dtu.dk

Abstract. Software Reference Architecture (SRA), which is a generic architecture solution for a specific type of software systems, provides foundation for the design of concrete architectures in terms of architecture design guidelines and architecture elements. The complexity and size of certain types of software systems need customized and systematic SRA design and evaluation methods. In this paper, we present a software Reference Architecture Design process Framework (RADeF) that can be used for analysis, design and evaluation of the SRA for provisioning of Tools as a Service as part of a cloud-enabled workSPACE (TSPACE). The framework is based on the state of the art results from literature and our experiences with designing software architectures for cloud-based systems. We have applied RADeF SRA design two types of TSPACE: software architecting TSPACE and software implementation TSPACE. The presented framework emphasizes on keeping the conceptual meta-model of the domain under investigation at the core of SRA design strategy and use it as a guiding tool for design, evaluation, implementation and evolution of the SRA. The framework also emphasizes to consider the nature of the tools to be provisioned and underlying cloud platforms to be used while designing SRA. The framework recommends adoption of the multi-faceted approach for evaluation of SRA and quantifiable measurement scheme to evaluate quality of the SRA. We foresee that RADeF can facilitate software architects and researchers during design, application and evaluation of a SRA and its instantiations into concrete software systems.

Keywords. Cloud Computing, Software Reference Architecture (SRA), Tools as a Service (TaaS), Architecture Design, Architecture evaluation.

1 Introduction

A Software Reference Architecture (SRA) provides an abstraction for designing and reasoning about a concrete software architecture of a specific system domain [1][2].

Whilst a concrete architecture is designed for a specific project according to well-defined business goals and requirements, a SRA usually aims to address generic business goals and domain requirements. A SRA consists of not only details on architecture components and its view, but also encompasses best practices for describing the architecture and the process guidelines for analysis, design and development of the architecture [3]. Though describing stakeholders concerns in terms of architecture view points and presenting the details of a SRA using multiple views [4] is important, it is equally important to describe the design-time and run-time quality characteristics of a SRA and the use of appropriate architecture styles and patterns [5]. A SRA is primarily designed for two main reasons: (i) to standardize existing available concrete architectures or (ii) to propose a preliminary SRA that can facilitate concrete architecture design for a specific domain. Whilst a SRA standardization effort focuses on extracting reusable architecture elements from a number of concrete architectures, a SRA preliminary proposition focuses on recommendations for SRA documentation, guidelines for SRA design and evaluation as well as SRA adoption and evolution.

In this paper, we present a software Reference Architecture Design process Framework (RADeF) for designing cloud-based systems in general and cloud-based Tools as a service workSPACE (TSPACE) in particular. RADeF reports a set of key specifications and SRA design guidelines. Whilst cloud-based systems provision on-demand computing as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [6], a TSPACE is characterized by as an activity or a task specific selection and on-demand provisioning of Tools as a Service (TaaS) as part of an integrated cloud-enabled workspace [6]. We assert that designing and evaluating a TSPACE SRA is more challenging than SRAs of general cloud-based systems because of the involvement of diversified tools and tenants with varying functional requirements and quality concerns. For example, performance and scalability can be more important for tenants and users of software development and testing TSPACE, whereas integration can be more important for tenant and users of architecture analysis and design TSPACE. Furthermore, instantiation of a TSPACE SRA for different domains can require customization (e.g. addition of new components or only selecting a subset of a SRA), which requires a mechanism that can be used to analyze quality and completeness of the instantiated architectures. Although there have been attempts to provide a systematic approaches for reference architecture design, documentation and evaluation [2, 3, 7], to the best of our knowledge, there has been a little work done on providing a process framework for SRA design given the specific needs of SRA design and evaluation of the TSPACE. Our work reported in this paper aims to address the following research objectives:

- Provide a systematic approach that can lead to a SRA's design elements identification, requirements analysis and detailed design.
- Provide insight to the specific needs of TSPACE SRA evaluation and instantiation into concrete architectures.
- Demonstrate application of RADeF on SRAs of software architecture design and implementation TSPACEs.

The organization of the paper is as follows. Section 2 provides the details of RADeF. Section 3 describes the results of the case studies of applying RADeF for

describing and implementing TSPACE. Section 4 provides an overview of the related work and Section 5 concludes the paper by sharing lessons learned and experiences.

2 Reference Architecture Design process Framework (RADeF)

A SRA is expected to provide guidance for designing and evaluating a concrete architecture. A SRA description usually includes reusable solutions in terms of architectural goals, architectural styles, design patterns, design principles and decision and guidelines for initiating a SRA. That is why it is important that a SRA description includes as much details as possible. It is also important to have a clearly described process that can be used to design and evaluate a SRA [2]. In this section, we describe RADeF, the process that can lead to TSPACE SRA development, evaluation and implementation. We also discuss important factors that should be considered at each stage of TSPACE SRA design. A pictorial representation of RADeF is presented in Fig. 1. RADeF is an iterative process framework and information produced in preceding stages is used as input for the proceeding stages of the process and as shown in Fig. 1.

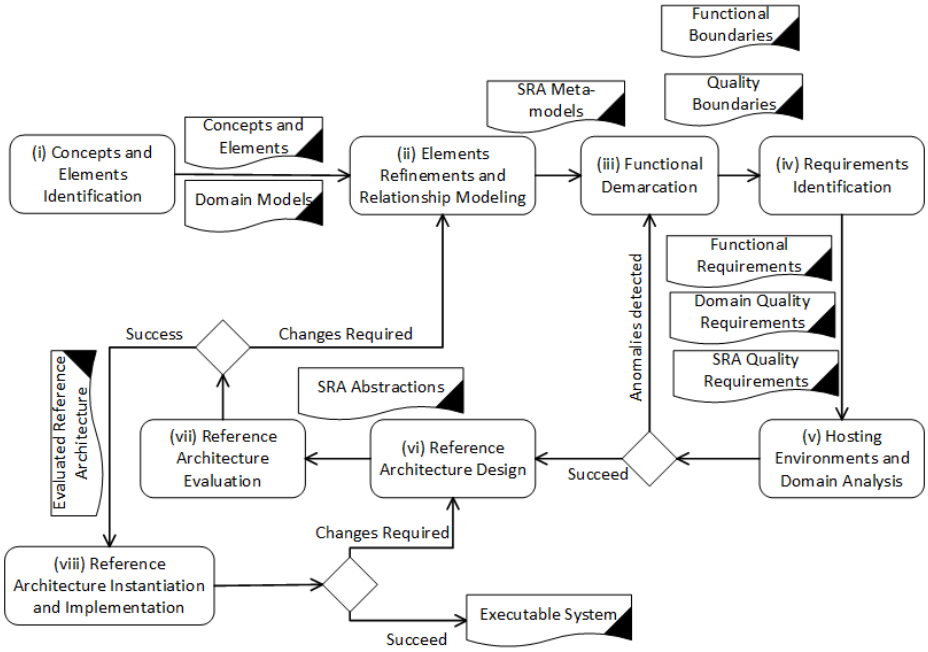

Fig. 1. A Process Framework for Designing a Software Reference Architecture

2.1 Identification of a Reference Architecture's Concepts and Elements

First step in designing a SRA of a cloud-based TSPACE is to identify the concepts and elements that constitute TSPACE. A SRA consists of not only SRA requirements

and SRA views, but it also provides guidelines for SRA evaluation. A generic view of the SRA elements is depicted in Fig. 2(a). The required concepts an elements are identified through a high-level analysis of a particular domain. TSPACE SRA elements can be classified into: Tenants, Tools, Provisioning Infrastructure, Artifacts, Context and Integration Methods. Each of the elements is tailored and extended with respect to the domain requirements for which the SRA is to be enacted.

Participants' Roles: End users, Requirements Analysts and Software Architects.

Artifact(s) Consumed: Business Requirements.

Artifact(s) Produced: High-level relationship models for TSPACE concepts and elements.

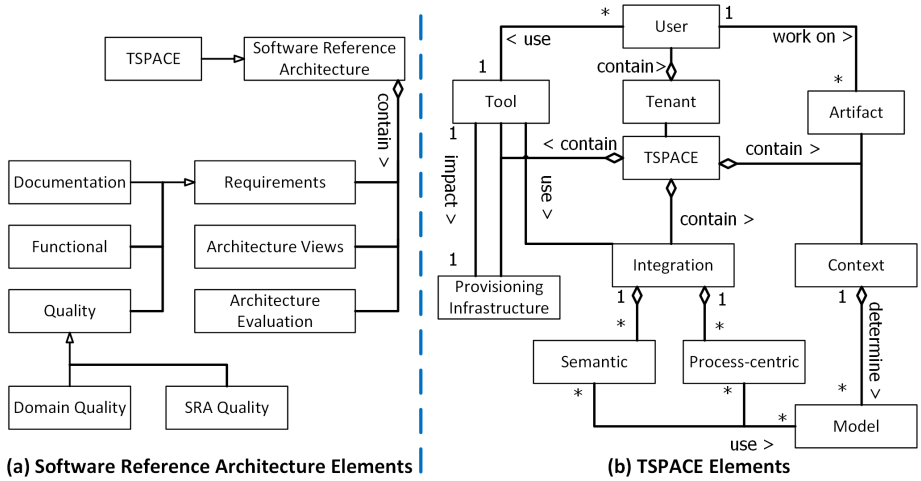

Fig. 2. Software Reference Architecture Elements

2.2 Refinement of Domain Element and Relationships Modeling

The activities identified for this stage are aimed at refining the identified elements in previous stage, establishing the hierarchical structure of TSPACE elements and modeling relations among the elements. Domain models are considered the main sources of the information for this stage. The domain models can provide standardizations for elements, their hierarchical structures and the relationships among the elements. However the domain models need to be extended in order to cover all the dimensions of TSPACE including the tools, the development processes which govern the provisioning and usage of the tools, data integration and exchange formats among the tools, and any additional functional aspects that are required by TSPACE in a specific domain. Fig. 2(b) shows TSPACE elements and relationships among the elements. The artifacts that are produced at this stage, serve as a foundation for the detailed requirements analysis and architecture design of the components that are responsible for tools bundling and integration in the TSPACE.

Participants' Roles: Business Analyst and Software Architect.

Artifact(s) Consumed: Documentation approaches, documentation templates and architecture design abstractions.
Artifact(s) Produced: TSPACE conceptual models that consists of concepts and elements that encompass TSPACE and relationship among the concepts and models.

2.3 Functional Demarcation between the Reference Architecture Elements and the Tools to be Provisioned

This stage of the activities deals with demarcation of functional requirements to be handled by a SRA and functional requirements for which TSPACE can rely on the tools (that can be provisioned by TSPACE). The artifacts that are produced at this stage provide a foundation for TSPACE functional requirements. The high-level architecture design with specific focus on the identification of components responsible for the TSPACE features.
Participants' Roles: Requirements Analyst, Business Analyst and Software Architect.
Artifact(s) Consumed: Domain models.
Artifact(s) Produced: Documents describing functional demarcation of TSPACE and encompassing tools.

2.4 Requirements Identification and Classification

The TSPACE SRA requirements can be classified into service model, integration and quality requirements as discussed below.

Service Model Requirements: This task aims at identifying the requirements for tools bundling, provisioning and enactment. For example, one of the primary objectives for providing a software architecting TSPACE is to provide the bundled suite of tools on demand as part of a TSPACE. It is critical to determine bundling and provisioning constraints and parameters. The tools bundling mechanism should be flexible enough to cater integration needs of different types of the tools to be used in a particular domain. In certain cases, there can also be some constraints with respect to the underlying virtualized infrastructure (e.g., IaaS cloud virtual machines) that can host the tools to enable their operations within acceptable runtime quality parameters (e.g., performance, scalability and reliability). The artifacts that are produced in this activity provide guidelines to identify integration needs of the tools in a TSPACE and guide the SRA analysis and design process.

Integration Requirements: Integration requirements focus on integration needs of the tools that can be provisioned in TSPACE. With reference to software architecting domain, the integration mechanism should be flexible enough to accommodate different proprietary and standardized formats as well as support integration among heterogeneous types of tools (e.g. desktop-based, web-based and cloud-based tools). The tools that are provisioned in a TSPACE instance can vary and the integration mechanism should be flexible enough to adapt to the tools' integration requirements of the provisioned tools. The integration mechanism should also support workspace requirements, such as awareness of the operations that are performed on the artifacts as

a result of the users' activities [8]. The artifacts that are produced at this stage guide the reference architecture design and analysis process of integration.

SRA Quality Requirements: **The** TSPACE is aimed at providing a bundled suite of tools following a service model. As a result, the TSPACE SRA needs to incorporate architecture quality requirements of cloud-enables services based system such as scalability [9], multi-tenancy [10] and dynamic provisioning [11]. The activities that are performed at this stage aim to identify important quality characteristics with reference to the domain in which the TSPACE is to be used. For the software architecting domain, scalability, multi-tenancy and dynamic provisioning are important. For another domain such as software testing, elasticity [12] and reliability [13] can be important. The artifacts that are produced as a result of this activity provide a foundation for runtime architecture quality requirements of TSPACE.
Participants' Roles: Business Analyst and Software Architect.
Artifact(s) Consumed: Design time constrains and tools bundling constraints, TSPACE functional boundaries, required activities and tasks, and tools enactment/provisioning parameters and constraints, Collaboration and integration models.
Artifact(s) Produced: Integration and collaboration models. Design time constrains, tools bundling constraints and tools' provisioning/enactment parameters. TSPACE runtime architecture quality requirements.

2.5 Impact of Potential Cloud Hosting Environments on the Domain

The suitability of the underlying IaaS or PaaS platform can impact the way a reference architecture is designed. E.g. PaaS environments can be a suitable choice for testing domains in which autonomous scalability of the resources is more important. Whilst IaaS environments can be suitable for hosting tools implemented using different technologies as IaaS clouds provide customizable hosting environments.
Participants' Roles: Software Architect.
Artifact(s) Consumed: List of potential cloud hosting environments.
Artifact(s) Produced: Selected cloud hosting environments.

2.6 Reference Architecture Documentation, Analysis and Design

This stage of the activities focus on analyzing architecture documentation approaches and preliminary analysis of the maturity of the domain for which a SRA is designed. The analysis of the documentation approaches determines the most appropriate strategies for capturing the architecture of the domain for which TSPACE is designed. A comprehensive analysis of the SRA documentation approaches is reported in [2, 3]. Angelov et al. have recommended that a reference architecture documentation include information about the context, goals and design decisions. The context dimension covers the purpose, the organization(s) that is (are) developing a reference architecture and its maturity stage (e.g., preliminary or classic) [2]. The goal dimension encompasses business goals and quality attributes as well as the purpose of defining a reference architecture (e.g., to standardize concrete architecture or to facilitate design of concrete architecture). The design dimension elaborates whether a SRA is concrete

or abstract and whether the SRA has been described using formal, semiformal or informal approaches. Avgeriou et al. [3] propose that a SRA description should have three main elements: (i) description of the approach used to document a SRA, (ii) guidelines on instantiation of a SRA and (iii) evaluation of a SRA corresponding to desired functional requirements and quality attributes. The outcome of this activity determines the approach used for describing a SRA, the level of abstractions to be covered in the SRA documentation, the objectives and the selection of the approaches for evaluation and instantiation of a SRA. Outcome of this activity has impact on all the proceeding stages of the reference architecture design process. A summary of a SRA design dimensions is shown in Fig. 2(a).

A SRA design should be based on reference models and architecture styles and patterns [14, 15]. If a TSPACE SRA is to be used for mission-critical and safety-critical tools, then it is also important to have metrics that can be used to measure runtime quality parameters of an architecture. An empirical investigation of the SRAs have revealed the absence of important views [4] in a SRA and the details of the supporting algorithms and formalization to achieve the required functionality of the reference architecture [4] impact a SRA's adoption and applicability. Hence, a SRA should encompass all the important views according to some well-known approaches such as 4+1 view model [4].

Participants' Roles: Software Architect.

Artifact(s) Consumed: Architecture documentation templates and models.

Artifact(s) Produced: SRA documentation approaches used, filled templates, details of the abstractions to be used, evaluation and initialization approaches and views.

2.7 Evaluating a Reference Architecture

Evaluation of a SRA is an important step for analyzing its feasibility and applicability. Different considerations for evaluating a reference architecture have been proposed [3, 7, 16]. Avgeriou et al. [3] have proposed to evaluate a SRA using scenarios and prototyping. Scenarios based approaches enable an implementation-independent evaluation. The evaluation scenarios need to be focused on important design time and runtime qualities of the architecture. The prototyping helps analyze the suitability of the implementation decisions such as platform choices and programming languages for the design decisions incorporated in a SRA. Angelov et al. [7, 16] have argued that straightforward adoption of architecture evaluation methods such as Architecture Tradeoff Analysis Methods (ATAM) [17] and Software Architecture Analysis Methods (SAAM) [18] is not feasible because: gathering all the stakeholders and generating scenarios for a SRA evaluation may not be possible, there can be a significantly large diversity of stakeholders and the levels of abstractions in the designed components can be quite high. Hence, it is important to identify the most relevant architecture requirements by involving domain experts or domain models and then preparing scenarios by involving a SRA's potential users [7, 16].

Other than the above-mentioned challenges, a TSPACE SRA evaluation activity has some additional complexities. For example, a TSPACE provision the tools for performing the different activities; hence there is a need for tools integration and

workspace specific functions in a *aaS model. An evaluation activity focuses on the parts of a SRA that are embodied by TSPACE boundaries rather than by the tools to be provisioned. Some of the key quality characteristics are inherited from *aaS model for evaluating a TSPACE SRA's abilities of on-demand provisioning of tools in a particular domain, whose quality attributes should drive the evaluation activities. Hence, the evaluation activity should focus on identifying and analyzing the relevant quality attributes for the given domain. Moreover, as the SRA's elements (i.e. components or services) and design decisions collectively constitute to SRA quality, traditional architecture analysis and evaluation methods such as utility tree [17] from ATAM are not sufficient because these are unable to quantify architecture quality. We advocate for leveraging an new approach inspired from attack-defense trees [19] to enhance the utility tree for analysis of the completeness of a SRA. Fig. 3(a) shows the structure of the enhances utility tree. Sub-nodes of the utility tree corresponding to each quality can be assigned with three types of operators: logical OR operator which identifies that opting any of the branch can achieve a quality attribute, logical AND operator that indicates that opting all of the branches will be essential to meet a quality criteria, and a Seq-AND (sequential AND) operator indicates that the design decisions corresponding to the branches need to be executed in a specific sequence in order to achieve the corresponding quality characteristic. In some cases, it might be required to analyze overall quality and completeness of the SRA. For this purpose, the probability values for the effectiveness of the design decisions can be assigned to each branch of the quality attribute nodes (such that maximum probability of all design decisions corresponding to each quality attribute do not exceed one). When probability values are used, OR operator takes minimum, AND takes sum and SeqAND takes sum of the probability values of all the branches of a quality attribute sub-tree. Finally, to evaluate the tools bundling and integration approaches, a prototype based evaluation is considered more effective. That means a TSPACE SRA prototype can play a critical role for the SRA evaluation and the tools that are selected for provisioning using the prototype can help to cover the most critical evaluation scenarios. The outcome of evaluation activity can trigger modification in the artifacts that were generated in previous stages as depicted in Fig. 1.

Participants' Roles: User, Requirement Analyst, Business Analyst and Software Architect.

Artifact(s) Consumed: TSPACE Software Reference Architecture.

Artifact(s) Produced: Evaluation results.

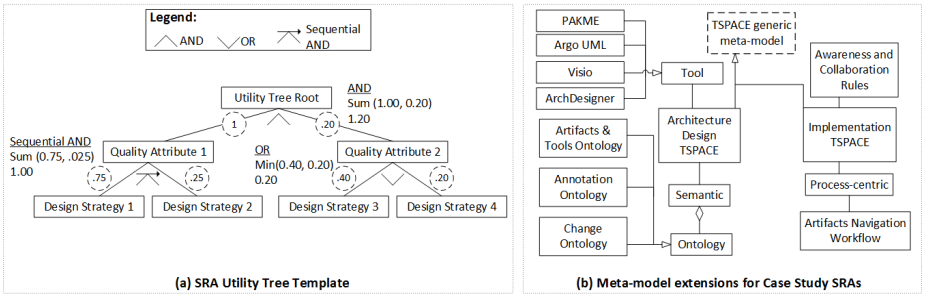

Fig. 3. SRA Utility Tree Template and Models

2.8 Reference Architecture Instantiation and Implementation

As a SRA provides a generic architecture solution for a specific domain, its instantiations can require appropriate tailoring, sometimes significant. As a result, some of the components can be excluded from the instantiated architecture and some additional components can be incorporated. The enhanced utility tree (Fig. 3(a)) presented in Section 2.7 can facilitate the analysis and quantification of the concrete architecture.
Participants' Roles: Business Analyst, Software Architect and Developers.
Artifact(s) Consumed: Evaluated TSPACE Software Reference Architecture.
Artifact(s) Produced: Instantiated system.

3 Two Cases of Applying RADeF

We have followed RADeF to support the design of a SRA for two types of TSPACE: software architecting tools and software implementation tools. The two case studies of applying RADeF for analysis, design, evaluation and implementation of the TSPACE aimed at provisioning integrated suite of tools for software architecting and implementation tools. The tools commonly used for software architecture design and software implementation were selected for the case studies and TSPACE was designed by following RADeF steps. In this section, we provide the insight gained from our experiences from applying RADeF.

First and second stage of RADeF is to ***identify concepts and elements of a SRA*** and ***establish relationships between the elements***. The generic model presented in Fig. 2(b) provides a foundation for TSPACE elements identification and relationship modeling. Though the generic model needs to be extended to cater the needs of a specific type of tools and the operations that can be performed using the tools. Fig. 3(b) shows the extensions to the generic model for software architecting and software implementation domain. The tools used for software architecting have different types of the artifacts, e.g., architecture knowledge artifacts, design decision artifacts and architecture design diagrams. Since software architecture artifacts can be at different levels of abstractions, and there is no need to exchange complete artifacts (although selected information exchange is required) among architecting tools, which can be integrated through semantic integration technologies. We have leveraged IEEE 1471-

2000 [20] and ISO/IEC/IEEE 42010:2011 [21] to build the semantic integration model for SRA of the architecting tools. Fig. 3(b) shows a high-level view of the elements of the semantic model (the details can be found in [22]). The software implementation need to exchange the artifacts for collaborative work. For example, in a scenario where a UML modeling tool is used to design class diagrams, the code skeleton generated using the UML modeling tool (forward engineering) has to be used as input by Integrated Development Environments (IDEs). For example, process-oriented tools bundling requires process-centric integration. At this stage the SRA integration models are produced that provide foundations for the detailed architecture design.

Functional demarcation between the requirements to be incorporated by a SRA and the requirements to be incorporated by the provisioned tools is an important step for the ***requirements identification***. As in the case studies, our focus was on providing software architecting and development tools, the SRAs focused on tools provisioning, tools integration and awareness of the operations that are performed on the artifacts using the tools. Whereas, individual tools were responsible for providing support for specific activities such as architecture knowledge management, architecture design decision management, architecture design and software implementation. Table 1 shows details of the SRA and the tools' requirements classification. The details of the requirements can be found in [22, 23]. Multi-tenancy and scalability are domain specific quality requirements to support a large number of tenants [24]. ***Analysis and identification of cloud hosting environments*** for software architecting and implementation domains requires using IaaS cloud because of heterogeneity of the tools. A combination of desktop and web-based tools are used for software architecting and implementation. The IaaS provides flexibility to host the existing tools by configuring the virtual machine templates.

Table 1. Functional Demarcation and Requirements

Functional Demarcation	Tools Requirements	Architecting	Knowledge management, design decision management, architecture modeling.
		Implementation	Software development, unit testing.
	SRA Requirements	Functional	Autonomous provisioning, semantic integration, process centric integration, awareness of the operations.
		Quality	Flexibility, interoperability, completeness and adaptability.
	Domain	Quality	Multi-tenancy, scalability

For the TSPACE ***SRAs detailed design,*** we have used a layered architecture [5] and a view-based approach [4] to represent different parts of the SRA. A layered architecture can facilitate easy modifiability of a TSPACE SRA, whose different dimensions can be represented using a view-based approach. The TSPACE meta-model (Fig. 2(b)) and the detailed models (Fig. 3(b)) produced in the second stage of RADeF are used as a foundation for the detailed design. Table 2 shows the key architecture

design decisions for software architecting and software implementation of a TSPACE SRA design. We have reported the details on the architecture views and design decisions in [6, 23].

Table 2. Decisions for software architecting (Arc.) and implementation (Impl.) case studies

Architecture Design Decisions	Case Study	
	Arc.	Impl.
Service Oriented and REST Architecture	✓	✓
Centralized Repository to have common semantic integration models	✓	×
Use of pipes and filter patterns to support multi-tenancy and easy scalability	✓	✓
Tenant specific integration, information discovery and awareness rules	✓	✓
Process-centric integration	✓	✓
IaaS cloud for hosting tools	✓	✓

As discussed earlier, the inclusion of heterogeneous tools producing and consuming artifacts at different levels of abstractions makes the evaluation of a TSPACE SRA a challenging activity. We have adopted multi-faceted approach to ***evaluate the TSPACE SRAs*** for the reported case studies. (i) We evaluated the TSPACE SRAs and their respective implementations for functional completeness corresponding to the functional and quality requirements. (ii) We implemented the prototype systems for TSPACE SRAs using Amazon IaaS cloud[1]. Interface modules of TSPACE have been implemented using Service Oriented Architecture (SOA) [25] and REST [26] architecture styles using JavaEE service technologies (JAX-RS[2], JAX-WS[3]) for enabling easy interoperability of different types of tools with the systems. The semantic integration has been implemented using Apache Jena Framework[4]. The process-centric integration has been implemented using jBPM[5] process workflow engine. (iii) We used quantitative architecture evaluation approach that is presented in Section 2.7, which is based upon utility tree of ATAM, but can quantifiably measures the TSPACE SRA's quality. The evaluation was carried out by six potential stakeholders, who had experiences (of architecting and implementation) of software development tools, process-based applications, cloud-based systems and collaborative software development systems.

A subset of the enhanced utility tree (described in Section 2.7) constructed in the evaluation session is presented in Fig. 4. The participants of the evaluation session were asked to assign each of the design decisions with values 0, 0.25, 0.50, 0.75 or 1.00. Then the average of the value score was taken for each of the design decisions

[1] http://aws.amazon.com/
[2] http://jax-rs-spec.java.net/
[3] https://jax-ws.java.net/
[4] https://jena.apache.org/
[5] http://www.jbpm.org/

to be assigned to a specific quality attribute on a utility tree branch. In case, if there were more than one design decisions corresponding to a specific quality attribute, an average was divided by the total number of design decisions to keep the maximum probability value under 1 corresponding to each of the quality attributes. If some of the design decisions are important than others, then weighted averages can be used. Whilst we considered all of the design decisions of the equal importance, the enhanced utility tree branches corresponding to each of the quality attributes (and sub attributes) had either AND, OR and SeqAND operators (as discussed in Section 2.7). The evaluation participants found the proposed operators (that were assigned to the enhanced utility tree) helpful to quantify the architectural quality of the TSPACE SRA. Fig. 4 shows the evaluation results corresponding to four key quality attributes of the TSPACE SRAs for software architecting and implementation TSPACE. An average of the quality score (average of the score given by the six evaluators) is shown in the figure corresponding to each of the design decisions of the quality attributes. Sum and Min functions (as described in Section 2.7) are used to calculate the aggregated quality score of the reference architectures.

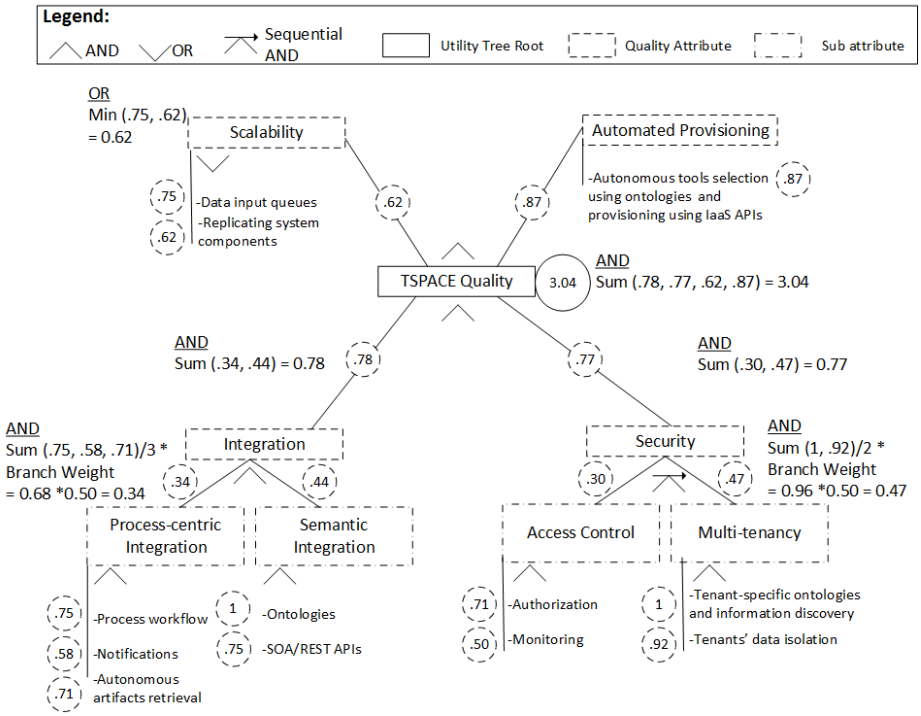

Fig. 4. SRA Evaluation Utility Tree

4 Related Work

Given the increasing importance of SRAs for guiding the designing and evaluating of concrete architectures in different domains, several researchers have attempted to

provide a set of standardized activities and frameworks for designing and documenting reference architectures. One of the most comprehensive and detailed guidelines have been reported by Angelov et al. [1, 2, 7, 16]. Their work provides a classification technique of the reference architectures based upon the domain-specific maturity and how the reference architectures are designed. For the mature domains, the aim of a reference architecture is to provide the standardization of the architectures, whereas, for the emerging domains, the purpose is to facilitate the design of concrete architectures in multiple organizations. Some of the problems associated with designing a reference architecture are missing design methods, challenges in defining non-functional requirements, problems with selecting appropriate views, lack of suitable architecture documentation methods and relatively little support for evaluating the reference architectures [1]. In our TSPACE SRA design process, we have explicitly catered all of the above-mentioned challenges to support the process of designing the reference architecture and have explicit stages for design and documentation methods, define non-functional requirements, select appropriate views and choose appropriate evaluation strategies.

Avgeriou [3] suggests representing a reference architecture using multiple viewpoints of Rational Unified Process (RUP) including logical viewpoint, deployment viewpoint, implementation viewpoint and data viewpoint. Avgeriou has emphasized that the reference architecture should be evaluated using both scenario-based and architecture prototype-based evaluation with respect to development-time and run-time qualities [3]. Nakagawa et al. [27] have proposed the use of ontologies to identify different components of the reference architecture. Fernandez et al. [28] have described the key documentation elements of a software reference architecture. The documentation elements include technical design, architecture knowledge and experiences and management documentation. For TSPACE SRA, we have described the details about the technical design and architecture knowledge. However, the management of the documentation (during applications of the software reference architecture in different setting) is out of the scope of this work.

5 Discussion and Conclusions

The cloud-enabled tools not only need to be compliant with specific quality requirements but also need to provide support for different activities, just like desktop-based tools. Whilst tools in every domain have their specific challenges, there are some generic characteristics that tools in every domain need to address. We share our experience from different activities of designing the TSPACE SRA.

Adoption of Appropriate Methodology to Formalize relations among TSPACE Elements: To establish relationship among the artifacts that are maintained by cloud-based tools with other tools is a critical characteristic and can play a significant role in cloud-based tools adoption. Hence, it is important to identify the integration needs for the tools to be provisioned from a cloud-enabled workspace. Our experience has shown that an ontology driven semantic model can provide support for tools selection, relating different artifacts with each other even though the artifacts are maintained by

using different proprietary data structures, and awareness needs in a cloud based workspace. As different tools have different requirements for integration, there is a need to have appropriate semantic integration models corresponding to the artifacts' formats used by the tools.

Incorporating Workflows with Tools Provisioning: In some cases, the tools that are provisioned as part of a tools suite need to exchange information according to project specific development processes (e.g., to manage collaboration in distributed architecture evaluation processes [29]). In such cases, the integration support for the tools needs to be complemented by a workflow based process on the cloud so that artifacts among the tools can be exchanged according to the specific software development processes.

Quality of Individual Tools in TSPACE: In our proposed TSPACE SRAs, we have considered each of the provisioned tools as a black box and have not considered the management of quality characteristics of each individually provisioned tool during the lifecycle of a TSPACE instance. However, for certain tools that produce executable artifacts, e.g. model driven tools used to generate code, may require extra computing, memory or other resources during their life cycle depending on the tasks to be executed. In such cases, a TSPACE for the tools needs to incorporate the metrics and corresponding prediction models so that additional resource needs can be predicted and resources can be acquired according to the needs of a specific task.

Impact of software reference models: Availability of standardization models for respective domain impact the reference architecture design process. Whist designing TSPACE software reference architecture for software architecting domain, we have leveraged IEEE 1471-2000 [20] and ISO/IEC/IEEE 42010:2011 [21] architecture documentation models as a baseline for the identification of the TSPACE architecture elements and the TSPACE ontology meta-model design. The meta-model has been further enhanced by analyzing architecting TSPACE requirements. The incorporation of standardized domain model in the reference architecture design ensures the applicability of reference architecture for broader range of tools. Unavailability of the standardization models for the respective domain or not using them during the reference architecture design can negatively impact the applicability of a reference architecture.

Selection of Appropriate underlying IaaS Clouds and Cloud Deployment Models: As tools in a TSPACE SRA are considered as black box, the tenant specific constrains on artifacts' storage location are applied onto the tools that are provisioned on the location that is compliant with the constraints (in our prototype implementation, we have used Amazon EC2 location specific provisioning features). However, for more complex use cases, where location constraints on the artifacts can change during their lifecycle, Virtual Machines (VMs) hosting the tools might need to be migrated from one location to another. In such cases, the capability of underlying IaaS to support VMs migration would play a critical role. Hence, IaaS cloud selection and selection of cloud deployment model (e.g., public, private or hybrid) should be carefully made. A cloud environment that supports the desired features should be selected.

Multi-facet approach for TSPACE SRA Evaluation: Considering a generic nature of TSPACE SRA and a broad range of potential stakeholders, multiple architecture evaluation techniques need to be adopted for evaluating a reference architecture from different perspectives. We have evaluated the TSPACE software reference architecture using scenario-based evaluation approaches [18], architecture tradeoff analysis method [17] and a prototype implementation of the reference architecture. Scenario-based evaluation approaches can help evaluate completeness of a SRA with respect to refer-

ence architecture objectives and requirements. Architecture tradeoff analysis method enables the identification of strong and week points of a SRA. A prototype is a viable way to demonstrate the feasibility of a SRA. The proposed TSPACE SRA evaluation methodology facilitates the quality score of not only the SRAs but also their concrete representations. For example, if a concrete implementation of the SRA corresponding to evaluation tree presented in Fig. 4 adopts different parts of the design decisions and corresponding components for different tenants, the quality of the instantiated architecture and corresponding system can be computed on the fly, especially for SaaS based systems.

In future, we intend to apply RADeF on software reference architecture design and analysis of other types of cloud-based systems. We also intend to carry out empirical evaluations on our proposed quantification mechanism for SRA evaluation utility tree to analyze its impact on long-term management of the software reference architectures.

References

1. Angelov, S., J. Trienekens, and R. Kusters, *Software reference architectures-exploring their usage and design in practice*, in *Software Architecture*. 2013, Springer. p. 17-24.
2. Angelov, S., P. Grefen, and D. Greefhorst, *A framework for analysis and design of software reference architectures.* Information and Software Technology, 2012. **54**(4): p. 417-431.
3. Avgeriou, P., *Describing, instantiating and evaluating a reference architecture: A case study.* Enterprise Architect Journal, 2003: p. 24.
4. Kruchten, P.B., *The 4+1 View Model of architecture.* Software, IEEE, 1995. **12**(6): p. 42-50.
5. Buschmann, F., et al., *Pattern-oriented software architecture: a system of patterns*. 1996: John Wiley & Sons, Inc. 457.
6. Chauhan, M.A., M. Ali Babar, and Q.Z. Sheng, *A Reference Architecture for a Cloud-Based Tools as a Service Workspace*, in *2015 IEEE Conference on Service Computing (SCC)*. 2015, IEEE: New York, USA.
7. Angelov, S., J.J. Trienekens, and P. Grefen, *Towards a method for the evaluation of reference architectures: Experiences from a case*, in *Software Architecture*. 2008, Springer. p. 225-240.
8. Dourish, P. and V. Bellotti, *Awareness and coordination in shared workspaces*, in *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*. 1992, ACM: Toronto, Ontario, Canada. p. 107-114.
9. Sodhi, B. and T.V. Prabhakar, *Application architecture considerations for cloud platforms*, in *2011 Third International Conference on Communication Systems and Networks (COMSNETS)*. 2011, IEEE. p. 1-4.
10. Domingo, E.J., et al., *CLOUDIO: A Cloud Computing-Oriented Multi-tenant Architecture for Business Information Systems*, in *2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*. 2010, IEEE. p. 532-533.
11. Calheiros, R.N., et al., *The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds.* Future Generation Computer Systems, 2012. **28**(6): p. 861-870.
12. Han, R., et al., *Enabling cost-aware and adaptive elasticity of multi-tier cloud applications.* Future Generation Computer Systems, 2014. **32**: p. 82-98.
13. Brandic, I., D. Music, and S. Dustdar, *Service mediation and negotiation bootstrapping as first achievements towards self-adaptable grid and cloud services*, in *Proceedings of the 6th international*

conference industry session on Grids meets autonomic computing. 2009, ACM: Barcelona, Spain. p. 1,Äì8.

14. Bass, L., P. Clements, and R. Kazman, *Software Architecture in Practice*. 2012: Addison-Wesley Professional. 640.
15. Avgeriou, P. and U. Zdun, *Architectural patterns revisited–a pattern.* 2005.
16. Angelov, S. and P. Grefen, *An e-contracting reference architecture.* Journal of Systems and Software, 2008. **81**(11): p. 1816-1844.
17. Kazman, R., et al. *The architecture tradeoff analysis method*. in *Engineering of Complex Computer Systems, 1998. ICECCS '98. Proceedings. Fourth IEEE International Conference on*. 1998.
18. Kazman, R., et al. *SAAM: a method for analyzing the properties of software architectures*. in *Software Engineering, 1994. Proceedings. ICSE-16., 16th International Conference on*. 1994.
19. Kordy, B., et al., *Attack–defense trees.* Journal of Logic and Computation, 2012: p. exs029.
20. *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.* IEEE Std 1471-2000, 2000: p. i-23.
21. *ISO/IEC/IEEE Systems and software engineering -- Architecture description.* ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000), 2011: p. 1-46.
22. Chauhan, M.A., *Foundations for Tools as a Service Workspace: A Reference Architecture.* PhD Dissertation, IT University of Copenhagen, Denmark, 2016(ITU-DS; No. 118).
23. Chauhan, M.A. and M.A. Babar, *PTaaS: Platform for Providing Software Developing Applications and Tools as a Service.* Technical Report TR-2014-176, 2014, URI: https://pure.itu.dk/ws/files/74130379/TR_2014_176.pdf.
24. Azeez, A., et al., *Multi-tenant SOA Middleware for Cloud Computing*, in *2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*. 2010, IEEE. p. 458-465.
25. Huhns, M.N. and M.P. Singh, *Service-oriented computing: Key concepts and principles.* Internet Computing, IEEE, 2005. **9**(1): p. 75-81.
26. Fielding, R.T., *Architectural styles and the design of network-based software architectures*. 2000, University of California, Irvine. p. 162.
27. Nakagawa, E.Y., E.F. Barbosa, and J.C. Maldonado. *Exploring ontologies to support the establishment of reference architectures: An example on software testing*. in *Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on*. 2009. IEEE.
28. Martínez-Fernández, S., et al. *Artifacts of software reference architectures: a case study*. in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. 2014. ACM.
29. Ali Babar, M., *A framework for groupware-supported software architecture evaluation process in global software development.* Journal of Software: Evolution and Process, 2012. **24**(2): p. 207-229.