COMPUTATIONAL SCIENCE AND TECHNIQUES
Volume 2, Number 1, 2014, 238-252
© Klaipėda University, 2014

Publisher: Klaipėda University
http://journals.ku.lt/index.php/CST
Online ISSN: 2029-9966
DOI: http://dx.doi.org/10.15181/csat.v2i1.77

AN APPROACH TO SOA DEVELOPMENT METHODOLOGY: SOUP COMPARISON WITH RUP AND XP

Sandra Svanidzaitė
Vilnius University Institute of Mathematics and Informatics, Lithuania
sandra.svanidzaite@gmail.com

Abstract. Service oriented architecture (SOA) is an architecture for distributed applications composed of distributed services with weak coupling that are designed to meet business requirements. One of the research priorities in the field of SOA is creating such software design and development methodology (SDDM) that takes into account all principles of this architecture and allows for effective and efficient application development. A lot of investigation has been carried out to find out whether can one of popular SDDM, such as agile methodologies or RUP suits, be adapted for SOA or there is a need to create some new Service-oriented SDDM. This paper compares one of Service-oriented SDDM – SOUP – with RUP and XP methodologies. The aim is to find out whether the SOUP methodology is already mature enough to assure successful development of SOA applications. This aim is accomplished by comparing activities, artifacts of SOUP and RUP and emphasizing which XP practices are used in SOUP.

Keywords: SOA, RUP for SOA, SOUP, SDDM.

Introduction

Service-Oriented Architecture is an architecture comprising loosely coupled services, described by platform-agnostic interfaces that can be discovered and invoked dynamically. Loosely coupled refers to defining interfaces in such a way that they are independent of each other's implementation. In a loosely coupled system, it is possible to swap-out one of the components and replace it with another and cause no effect to the system (SOA, 2004), (Lewis et al., 2010)

According to (Mittal, 2010) SOA projects potentially suffer from one or more of the following problems:

- SOA projects are significantly more complex than typical software projects, because they require a larger, cross-functional team along with correspondingly more complex inter-team communication and logistics.
- Usually it is hard to define the scope and boundaries of a SOA project. As a result, the vision for the final result is often not clear at the project's inception.
- SOA can have a very positive impact on an organization, but, on the other hand, SOA development and replacement of legacy systems can be very expensive.
- SOA project has a higher risk of failure than other traditional software development projects.

To overcome these problems one of the research priorities in the field of SOA is creating a proper software design and development methodology that considers all SOA principles and causes effective and efficient application development of this architecture (Mamaghani et al., 2010). Lots of investigations have been carried out to find out which of nowadays used software design and development methodologies, such as *agile* methodologies or RUP, suits SOA principles the best, or whether there is a need of a new SOA design and development methodology.

This paper contributes to finding whether SOUP methodology is already mature enough to assure successful development of SOA applications by comparing SOUP methodology (Mittal, 2010), *"a methodology which has taken the best elements from RUP and XP and is targeted specifically at SOA projects"* with RUP and XP. The paper focuses on the comparison of SOUP and RUP activities and artifacts and highlights the practices of XP that are included in SOUP with the aim to reveal the differences between SOUP, RUP, XP and to find out whether SOUP methodology covers all the best features of RUP and XP that are necessary for SOA development.

The remainder of the paper is organized as follows. Section 1 briefly discusses the main characterization of RUP, SOUP and XP methodologies. Section 2 compares SOUP, RUP and XP and provides analysis of each SOUP phase. Finally, Conclusion section discusses and concludes the paper.

1. The main characterization of RUP, SOUP and XP Methodologies

The Rational Unified Process (RUP) is an iterative software design and development methodology, targeted at object-oriented paradigm solutions. It was created by Rational Software Corporation. The most important features of RUP are as follows (Kruchten, 2000):

1. RUP is a well-defined and organized software engineering process framework with defined *Roles*, *Work products* and *Activities*. *Role* defines a set of related skills, competencies and responsibilities; *Work product* represents something resulting from a task, including all the documents and models produced while working through the process; *Activity* describes a unit of work assigned to a role that provides a meaningful result.
2. RUP has two dimensions: *static* and *dynamic*. Dynamic dimension is described in terms of phases (*inception, elaboration, construction* and *transition*), iterations and milestones. Static dimension is described in terms of disciplines (*business modeling, requirements, analysis and design, implementation, test, deployment, configuration and change management, project management, environment*), activities, work products and roles. Main RUP activities and artifacts are described in details in section 2 of this paper.
3. RUP is an adaptable process framework, intended to be tailored by the development organizations and software project teams that will select the elements of the process that are appropriate for their needs.

Extreme programming (XP) is an iterative agile software design and development methodology created by Kent Beck and described firstly in his book (Beck, 2000). XP is a lightweight methodology for small-to-medium-sized teams developing software in the face of vague or rapidly changing requirements. XP life cycle contains *exploration*, *planning*, *iteration to release*, *production*, *maintenance* and *death* phases, includes 12 practices, grouped into four areas, derived from the best practices of software engineering (Beck, 2000):

1. Group: ***Fine scale feedback***
   1. **Practice:** *Pair Programming.* Software is built by two programmers, sitting side by side, at the same machine. Code is reviewed by at least one other programmer.
   2. **Practice:** *Planning Game.* A meeting that occurs once per software development iteration. The planning process is divided into two parts: *1) Release Planning* - is focused on determining what requirements are included in near-term releases, and when they should be delivered. *2) Iteration Planning* – is focused on planning the activities and tasks for the developers.
   3. **Practice:** *Test Driven Development.* Unit tests (automated tests that test the functionality of pieces of the code e.g. classes, methods) are written before the eventual code is produced. This approach is intended to stimulate the programmer to think about conditions in which his or her code could fail and allows producing fully tested code with 100 percent coverage during project iteration.
   4. **Practice:** *Whole Team.* All contributors to an XP project are one team. Customer should be on hand at all times and available for questions.
2. Group. ***Continuous process***
   5. **Practice:** *Continuous Integration.* The development team should always keep the system fully integrated and work with the latest version of the software.
   6. **Practice:** *Design Improvement.* XP promotes to develop only what is needed for today and keep it as simple as possible. After a few iterations code needs to be refactored. Code refactoring increases cohesion, reduces coupling and removes code duplications.
   7. **Practice:** *Small Releases.* The delivery of the software is done via frequent, functional and tested releases. The small releases help the customer to gain confidence in the progress of the project. Customer can evaluate software release and in turn provide feedback.
3. Group. ***Shared understanding***
   8. **Practice:** *Coding Standards.* Coding standard is an agreed set of coding rules that the entire development team must adhere to throughout the project. The standard specifies a consistent style and format for source code, within the chosen programming language, as well as various programming constructs and patterns that should be avoided in order to reduce the probability of defects (Kolawa, 2007)
   9. **Practice:** *Collective Code Ownership.* All project programmers are responsible for all the code. This, in turn, means that everybody is allowed to change any part of the code.
   10. **Practice:** *Simple Design.* Build software in a simple design. Keep the software simple and suited to current functionality.
   11. **Practice:** *System Metaphor.* It is a naming concept for classes and methods that should make it easy for a team member to guess the functionality of a particular class/method, from its name only. XP teams should develop a common vision of the system and everyone should understand how the system works, where to look for functionality, or where to add functionality
4. Group: ***Programmer welfare***
   12. **Practice:** *Sustainable Pace.* Programmers should not work more than 40 hour in a week. A key enabler to achieve sustainable pace is frequent code-merge and always executable, test covered high quality code.

Service-oriented Unified Process (SOUP) is a hybrid software engineering methodology that is targeted at SOA projects. It is proposed by Kunal Mittal from Sony Pictures Entertainment (Mittal, 2010) and is based on RUP and XP methodologies.

During the initial SOA project in organization there's a need for some formal software methodology analogous to RUP that addresses all risks of the project. An agile methodology like XP might not be formal enough and the most important drawback of it is the lack of documentation and any up-front design of the system or of its requirements. However, after initial SOA project is successfully established, continuing to use this formal methodology for SOA maintenance or development of new SOAs on the top of existing ones can make the process too complex.

SOUP is a six-phase methodology for SOA application development. SOUP phases represent a set of activities and artifacts that are critical to the success of a SOA project. SOUP processes are divided into two categories: 1) for initial SOA development, 2) for ongoing SOA management and new SOAs development on the top of existing ones.

SOUP includes six processes: *incept*, *define*, *design*, *construct*, *deploy* and *support* that comes into two slightly different variations: one for initial SOA development and one for ongoing SOA development. SOUP process model for initial SOA development is provided in Figure 1.

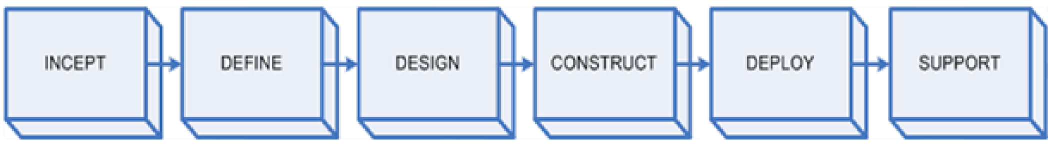

Figure 1. SOUP Process Model for Initial SOA Development (Mittal, 2010)

Figure 2 shows how SOUP phases map into RUP phases.

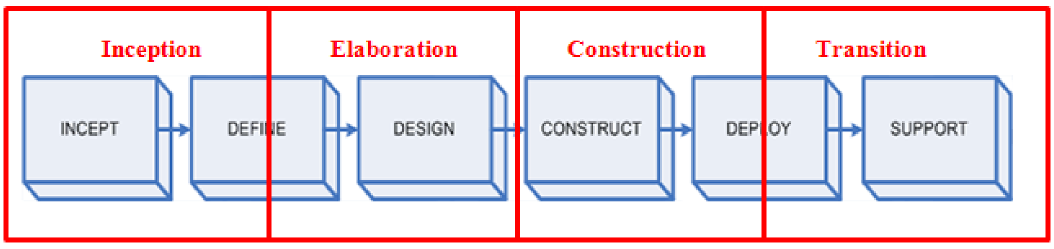

Figure 2. SOUP and RUP Model (Mittal, 2010)

SOUP process model for ongoing SOA Development is provided in Figure 3. Here (Mittal, 2010) shows how SOUP and XP can be overlaid on each other.

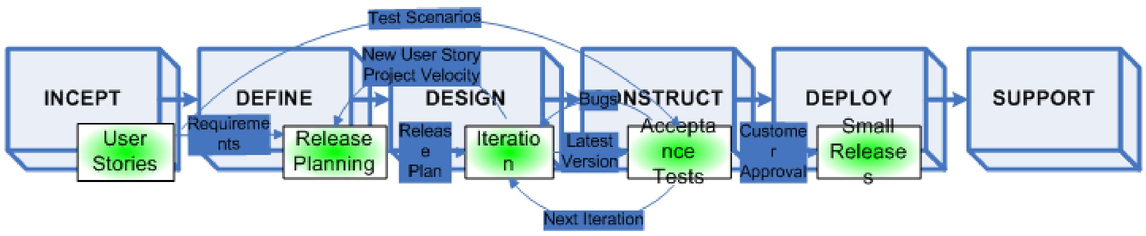

Figure 3. SOUP Process Model for Ongoing SOA Development (Mittal, 2010)

As we see from the Figure 2. and Figure 3. the mapping of SOUP to RUP and XP is made at a very high abstraction level without providing any details about which RUP activities and artifacts and XP practices are included in SOUP and which ones are skipped. In the next section of this paper we will make a deeper analysis and explore how the elements of SOUP methodology maps into RUP and XP.

2. Comparison of SOUP, RUP and XP Phases

A comparison of SOUP, RUP and XP is carried out in a following way:

1. Each phase of the three methodologies is described:
   1) by outlining the purpose and objectives,
   2) mapping SOUP and RUP activities to artifacts. SOUP phase description includes all activities and artifacts of the methodology. RUP phase description includes only the most important activities and artifacts.
   3) defining XP practices that are used in each XP phase. XP is a highly adaptable methodology that does not provide extensive lists of activities and artifacts and enforces project team to self-organize by leaving room to decide which activities should be accomplished and which artifacts should be produced, taking phase objectives and XP practices into account;
2. The most similar SOUP, RUP and XP phases are grouped together;
3. Each SOUP phase is thoroughly analyzed by defining features taken (and omitted) from RUP and XP methodologies.

SOUP: Incept phase for Initial and Ongoing SOA Development. The purpose of this phase for initial SOA development is to understand the business needs for SOA adoption and how SOA fits within the organization. The objective of this phase is to decide whether SOA project is profitable (or not) by evaluating project scope and risks.

SOUP *Incept* phase for ongoing development concerns on building new SOA projects that consume existing services and expose new ones.

A number of activities and artifacts for initial and ongoing SOUP Incept phase are described in table 1 (Mittal, 2010).

Table 1. Incept Phase of SOUP: Activity to Artifact Mapping

Activity	Artifact
Formulate the vision and the scope of the project	*Vision and scope document.* This document outlines the overall vision of the SOA project. It also provides some boundaries that establish the project's scope.
Define SOA project strategy	*Strategy document.* It is a high-level plan determining how a project can be implemented: business analysts define business high-level requirements in order to determine the advantages of service-oriented solutions and propose appropriate strategy.
Accomplish ROI analysis	*Return on Investment (ROI) document.* The document outlines costs and savings of the project. It should determine short-term and long-term benefits.
*Create communications plan** *Only for Initial SOA Development	*Communications plan.* It explains how SOA implementation team should collaborate with project stakeholders.

RUP: Inception phase. The purpose of this phase is to assure that project is feasible by developing vision and scope of the project, identifying stakeholders and assessing project risks. The objectives of this phase are: 1) to establish the project's scope and boundaries, 2) to define use cases of the system that will drive critical functionality, 3) to demonstrate at least one systems' candidate architecture against the primary use cases, 4) to estimate the overall cost and schedule for the entire project, and 5) evaluate risks.

A number of activities and artifacts for RUP Inception phase are described in table 2 (Kruchten, 2000).

Table 2. Inception Phase of RUP: Activity to Artifact Mapping

Activity	Artifact
Formulate the vision and the scope of the project	*Vision and scope document.* This document includes core project's requirements, key features and project boundaries.
Prepare business cases and project glossary	*Initial business case.* This document includes: business context, success criteria, financial forecast. *Business use case model.* It describes business functions and is used as an input to identify user roles their functions and deliverables in the organization. *Initial project glossary.* This document defines a common terminology that is used consistently across the project and organization.
Prepare initial use cases	*Use case model survey.* This document lists all use cases and actors that can be identified at this early stage. *Initial use-case model.* It is a communication medium that serves as a contract among the customer, the users and the system developers. It allows validating that the system will do what it is expected to do and that system developers will implement what they are expected to implement. An initial use-case model should be no more than 10% to 20% complete during the initial phase.
Evaluate risks	*Initial risk assessment.* This document describes risks by providing: project phase and activity in which risk may happen, current control measures and actions to be taken.
Plan the project	*Project plan.* The document shows the phases, activities, iterations of the project.
Plan the development process	*Preliminary case of software development process.* This document includes life-cycle model, development team structure description, activities to be performed, practices to be followed and the artifacts to be produced.
Develop prototypes	*Prototypes.* RUP highlights such kinds of prototypes as: behavioral, structural, exploratory, evolutionary prototypes that can be used to reduce risk.

XP: Exploration phase. The purposes of this phase are to prepare initial requirements for the system in the form of story cards and/or user stories and to familiarize with tools, technology and practices that will be used throughout the project. The objectives of this phase are: 1) to get enough requirements that will allow formulating system metaphor and building the whole system architecture skeleton and 2) explore all technologies and tools that will be used in the project by creating system prototype. XP practices used in this phase are: *System metaphor, Whole Team* (Beck, 2000).

XP: Planning phase. The purpose of this initial project planning phase is to agree with the customer on a date by which the smallest, most valuable set of user stories will be done. The objectives of this phase are: 1) to prioritize user stories and assign them to first iteration (and to any other iteration if more user stories are prepared), 2) to estimate the initial set of user stories, including every design, programming and testing task that is needed to perform. XP practices used in this phase are: *Planning Game* (Beck, 2000).

The analysis of SOUP Incept phase for Initial and Ongoing SOA Development. This phase makes only initial steps to the SOA project and includes only project preparatory activities, as a result, it differs a lot from RUP *Inception* phase, as it not only includes project preparatory activities but contains requirements analysis, risk management, prototyping, development process planning activities as well (more details can be found in table 2).

In addition to this, SOUP *Incept* phase contains *Define SOA project strategy activity* that is a SOA specific one and cannot be taken neither from RUP, nor from XP methodology.

Furthermore, first XP phase - *Exploration* is more alike to RUP *Inception* than to SOUP *Incept* phase as it intends to start the project by preparing initial set of requirements, formulating system metaphor and creating system prototype. On the other hand, XP *System metaphor* principle is partly realized in SOUP *Inception* phase as the boundaries of SOA and initial high-level requirements are defined allowing project team to formulate system metaphor.

One of the biggest drawbacks of SOUP *Incept* phase and the whole SOUP as well, is that it does not include any SOA project iteration planning activities likes the ones described in XP *Planning* phase. As a result, XP *Planning Game* practice is skipped in SOUP.

To sum up, SOUP *Incept* phase differs a lot from RUP *Inception*, XP *Exploration* and *Planning* phases and most of RUP *Inception*, XP *Exploration* phase activities are accomplished and artifacts are produced in SOUP *Define* and *Design* phases where SOA analysis and design begins.

SOUP: Define phase. The purpose of this the most critical phase for SOA project during initial SOA development is to define the requirements and develop use cases. The objectives of this phase are: 1) to fully understand business processes affected, 2) to collect, define and analyze functional and non-functional requirements by using a formal requirements-gathering and management process like RUP, 3) to design support and governance model which explains how organization will support SOA, 4) to prepare a realistic project plan, 5) to define technical infrastructure that is required to support entire SOA.

The purpose of this phase during ongoing SOA development is to identify services that are already available and what new services are required for new SOA project. Requirements gathering activities during ongoing SOA development does not need to follow formal standards like during initial SOA development. Instead, agile requirements gathering techniques involving user stories or Class, Responsibilities, Collaborator (CRC) cards can be used.

A number of activities and a set of artifacts for initial and ongoing SOUP *Define* phase are described in table 3 (Mittal, 2010).

Table 3. Define phase of SOUP: Activity to Artifact Mapping

Activity	Artifact
Requirements gathering and analysis	*Functional requirement document:* This document provides a detailed explanation of all business requirements that SOA will group and cover as business services.
*Services Identification** *Only for Ongoing SOA Development	*Services strategy:* This document identifies services that already exist and can be consumed along with new services that will be exposed.
*Non-functional requirements definition and analysis** *Only for Initial SOA Development	*Non-functional requirement document.* This document describes business requirements including performance considerations, service-level agreements (SLAs), operational-level agreements (OLAs), infrastructure requirements, and etc.
Use case/user story definition and realization	*Use cases or user stories definition and realization.* This document includes detailed use cases or user stories for all services that will be built.
Overall architecture definition and documentation	*SOA architecture document.* This document describes the overall architecture including hardware and software components and is created only during initial SOA development. *SOA applicability document.* This document explains which other projects fall within the scope of the SOA project and how ongoing projects can be built on top of the SOA. For Ongoing SOA projects this document outlines how the existing SOA framework applies to the project.
Technical infrastructure definition *Only for Initial SOA Development	*Infrastructure definition document.* This document includes detailed infrastructure deployment diagrams, outlining the servers, and the connections between them necessary to implement the SOA.
Creating a project plan	*Project plan.* This document provides detailed plan for the whole project includes activities, workers, estimates, milestones and etc. *Governance and support model.* This document describes how SOA will be supported and governed also it includes considerations such as SLA monitoring and management.
Test Case Development *Only for Ongoing SOA Development	*Test plan:* This document includes test cases for SOA project.

SOUP: Design phase. The purpose of *Design* phase for initial SOA development is to translate use case realizations and SOA architecture document into detailed design documents. The objectives of this phase are: 1) to create detailed design document consisting data base model, 2) to structure the development process by defining the technology, coding standards and etc.

SOUP *Design* phase for ongoing SOA development is really quick, because existing SOA design from initial SOA development is updated. The only concerns are how to reuse the existing services and design new services.

A number of activities and a set of artifacts initial and ongoing SOUP *Design* phase are described in the table 4 (Mittal, 2010).

Table 4. Design Phase of SOUP: Activity to Artifact mapping

Activity	Artifact
Create detailed architecture document with data base model	*Detailed design document.* This document explains how services are designed and built *Data base model.* This document includes ERD (Entity Relationship Diagram) of databases used for SOA.
*To structure the development process** *Only for Initial SOA Development	*Application programming model.* This document includes guidelines on how the development will be structured. It covers such topics as process and technologies being used, coding standards, deployment procedures and so on.
*Prepare for testing** *Only for Initial SOA Development	*Testing and QA plan.* This document includes detailed test and quality assurance cases.

RUP: Elaboration phase. The purpose of this phase is to analyze the problem domain, build an executable architecture prototype, develop the project plan and eliminate the project's highest risk elements. The objectives of this phase are: 1) to define, validate and baseline the architecture, 2) to baseline the vision and scope of the system, 3) to prepare a high-fidelity plan for the construction phase, 3) to demonstrate that the architecture will support base lined vision for a reasonable cost in a reasonable time.

A number of essential activities and a set of essential artifacts for RUP *Elaboration* phase are described in the table 5 (Kruchten, 2000).

Table 5. Elaboration Phase of RUP: Activity to Artifact mapping

Activity	Artifact
Prepare use cases	A *use-case model* (at least 80% complete) that includes all use cases and actors that were identified in the use-case model survey.
Elicitate non-functional requirements	*The list of supplementary requirements.* It captures the non-functional requirements and any requirements that are not associated with a specific use case.
Create an executable architectural prototype and architecture description	*Software architecture description.* This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. *Executable architectural prototype.* It is a partial implementation of the system that demonstrates selected system functions and properties, essentially those showing non-functional requirements.
Revise risks	*Risk list.* This document lists all known risks in a decreasing order of importance. *Risk identification and management plan.* This document describes how to manage risks associated with a project. It shows what risks management tasks will be carried out, assigned responsibilities and any additional resources required.
Revise business case	*Revised business case.* This document includes: business context, success criteria, financial forecast.
Create a project development plan	*A development plan* for the overall project which includes the coarse-grained project plan, showing iterations and evaluation criteria for each iteration
Update development process	*An updated development case.* This document includes life-cycle model, development team structure description, activities to be performed, the practices to be followed and the artifacts to be produced.
Prepare for testing	*Test case.* It contains a specific set of test inputs, execution conditions, and expected results, identified for the purpose of making an evaluation of some particular aspect. *Test plan.* This document defines the test items being targeted, approach to be taken, resources required and deliverables produced.

XP: Iteration to Release phase. The purpose of this phase is to iteratively perform system analysis, design, coding and testing and *2-n* iteration planning activities. The objectives of this phase are: 1) to create the whole system architecture during first iteration, 2) to pick the most valuable user stories for customer for the next iteration, 3) to perform analysis, design, coding and testing activities within all remaining iterations. XP practices used in this phase are: *Pair Programming, Planning Game, Test Driven Development, Whole Team, Continuous Integration, Design Improvement, Coding Standards, Collective Code Ownership, Simple Design, System Metaphor, Sustainable Pace* (Beck, 2000).

The analysis of SOUP Define and Design phases for Initial and Ongoing SOA Development. These two SOUP phases are highly based on RUP *Inception* and *Elaboration* phases meaning that most of SOUP *Define* and *Design* phase's activities and artifacts overlap with RUP *Inception* and *Elaboration* phase activities and artifacts. The biggest lacks of SOUP *Define* and *Design* phases is no explicit activity for project risk management and executable SOA architecture prototype creation.

In addition to this, SOUP *Define* phase contains *Services Identification, Non-functional requirements definition and analysis, Overall architecture definition and documentation* that are highly SOA specific and cannot be taken neither from RUP, nor from XP methodology. Furthermore, SOUP *Define* and *Design* phases cover XP *Iteration to Release* phase analysis, design activities and *System Metaphor* practice.

SOUP: Construct phase. The purpose of this phase for initial SOA development is to construct SOA application. The objectives of this phase are: 1) to iteratively develop, integrate and test SOA, 2) to create user documentation.

Construct phase for ongoing SOA development involves more assembly than development activities. As more services become available, each SOA project will have more to reuse and less to build. XP's iterative development techniques are ideal at this stage of development. In XP, iterative cycles are theoretically set at two weeks, which should be enough time for one development-QA cycle when building a small service or reusing a set of services in an SOA environment.

A number of activities and a set of artifacts for SOUP *Construct* phase are described in the table 6 (Mittal, 2010).

Table 6. Construct Phase of SOUP: Activity to Artifact Mapping

Activity	Artifact
Iterative development	*Code base* that should be stored in repositories. *New services.* Any new services that are being exposed and will be ready by the end of this phase (only for ongoing SOA development).
Iterative QA and testing	*Test results.* The results of testing and quality assurance should be stored and kept available for examination.
User documentation	*User documentation* which was developed in design phase should be kept up-to-date including detailed documentation of SOA system.

RUP: Construction phase. The purpose of this phase is to develop, integrate and test all components and application features. The objectives of this phase are: 1) to develop software product and achieve adequate quality by minimizing development costs and optimizing resources, 2) to achieve useful versions (alpha, beta) and make descriptions of the releases.

A number of activities and a set of artifacts for RUP *Construction* phase are described in the table 7 (Kruchten, 2000).

Table 7. Construction phase of RUP: Activity to Artifact Mapping

Activity	Artifact
Develop software product	*The software product.* The software product integrated on the adequate platforms.
Write user manuals	*User manual.* This document is used in training sessions and assist users with product use, operation or maintenance.
Make descriptions of software releases	*A description of the current release.* This document describes new functionality.
Test the software	*Test results.* The results of testing and quality assurance should be stored and kept available for examination.

XP: Productionizing phase. The purpose of this phase is to prepare system for release to customer. The objectives of this phase are: 1) to perform additional tests (system testing, load testing, installation testing), 2) to improve systems' performance before the system is released to the customer, 3) to create documentation such as *system documentation* (includes an overview of the technical architecture, the business architecture, the high-level requirements for the system, a summary of critical design decisions, architecture-level diagrams, and important design models), *operations documentation* (includes system dependencies with other systems, the nature of its interaction with other systems, databases, and files, references to backup procedures, the expected load profile of the system and troubleshooting guideline), *support documentation* (includes training materials specific to support staff, all user documentation to use as reference when solving problems, escalation procedures for handling difficult problems), *user documentation* (including reference manual, a usage guide, support guide and training materials). At this phase, new changes may still be found and the decision has to be made if they will be included in the current or next release. The postponed ideas and suggestions are documented for later implementation during, e.g., the maintenance phase. XP practices used in this phase are: *Test Driven Development, Whole Team, Continuous Integration, Design Improvement, Small Releases, Sustainable Pace* (Beck, 2000).

The analysis of SOUP Construct phase for Initial and Ongoing SOA Development. The purposes of SOUP *Construct* phase and of RUP *Construction* phase are almost the same, although RUP gives attention to software releases. SOUP omits software release description activity. SOUP *Construct* phase is highly based on XP *Iteration to Release* and *Productionizing* phases and incorporates *Pair Programming, Test Driven Development, Whole Team, Continuous Integration, Design Improvement, Coding Standards, Collective Code Ownership, Simple Design, System Metaphor, Sustainable Pace* practices.

SOUP: Deploy phase. The purpose of this phase for initial and ongoing SOA project is to deploy SOA project in production. In this phase, either SOA application or new services are launched. The objectives of this phase for initial SOA project are: 1) to create SOA deployment model, 2) to create SOA application and support models.

A number of activities and a set of artifacts for SOUP Deploy phase are listed in the table 8 (Mittal, 2010).

Table 8. Deploy phase of SOUP: Activity to Artifact Mapping

Activity	Artifact
*Create deployment model** *Only for Initial SOA Development	*Deployment model.* This document describes the structure of SOA deployment.
*Create application model** *Only for Initial SOA Development	*Usage model.* This document gives guides how to use SOA. It becomes important as various project teams begin to use new architecture.
*Create support model** *Only for Initial SOA Development	*Ongoing support levels model.* This document systematically organizes updates of governance and support model that was developed in the Define phase.

SOUP: Support phase. The purpose of this phase for initial SOA project is to ensure ongoing SOA support (Mittal, 2010). The objectives of this phase are to support SOA by making bug fixes, trainings and new functionality development.

During *Support* phase for ongoing SOA project support is provided only for new services. In doing so, support model laid down during the initial SOA development is followed.

SOUP provides only a number of activities that should be accomplished during this phase. The list of activities is as follows: *Maintenance, Bug fixes, Training, Continuous project buy-in.*

RUP: Transition phase. The purpose of this phase is to move the software product to the end user by accomplishing beta testing, making conversions of operational data bases, operating parallel with existing legacy system and training the users. The objectives of this phase are: 1) to achieve user self-supportability, 2) to achieve stakeholder approval that deployment baselines are complete and consistent with the evaluation criteria of the vision, 3) to achieve final product baseline.

RUP provides only a number of activities that should be accomplished during this phase. The list of activities is as follows: *To organize and provide training for the end users, To fix the remaining bugs and enhance the performance and usability, Assess the deployment baselines against the vision and the acceptance criteria for the product.* (Kruchten, 2000)

XP: Maintenance phase. The purpose of this phase is to keep the system in the production while at the same time produce new iteration. This phase encompasses the *Planning*, *Iterations to Release*, and *Productionizing* phases for iterations 2 through *N* of the system. The objectives of this phase are: 1) produce new functionality; 2) keep existing system running; 3) refactor system; 4) prepare new user stories for next iterations. XP practices used in this phase are: *Pair Programming, Planning Game, Test Driven Development, Whole Team, Continuous Integration, Design Improvement, Small Releases, Coding Standards, Collective Code Ownership, Simple Design, System Metaphor, Sustainable Pace* (Beck, 2000).

XP: Death phase. The purpose of this phase is to close the project. *Death* phase is when the customer does no longer have any user stories to be implemented. The objectives of this phase are: 1) to create final system documentation and 2) to close project. XP practices used in this phase are: *Whole Team, System Metaphor* (Beck, 2000).

The analysis of SOUP Deploy and Support phases for Initial and Ongoing SOA Development. SOUP *Support* phase differs a lot from RUP *Transition* phase as it is aimed at software support that should be carried after software deployment. RUP *Transition* phase is aimed at the last tasks that should be done before software deployment to production.

In addition to this, SOUP *Deploy* phase contains *Create deployment model, Create application model, Create support model* activities that are SOA specific and cannot be taken neither from RUP, nor from XP methodology.

SOUP *Support* phase is highly based on XP *Maintenance* phase and incorporates *Pair Programming, Planning Game, Test Driven Development, Whole Team, Continuous Integration, Design Improvement, Small Releases, Coding Standards, Collective Code Ownership, Simple Design, System Metaphor, Sustainable Pace* practices. *XP Death* phase is not covered in SOUP.

Conclusion

SOA is a new software development paradigm and the methodologies for successful SOA implementation are still under research. The aim of this paper was to compare SOUP - a new software development methodology for service-oriented paradigm with RUP - a methodology, which has provided a large database of knowledge and best practices from successful developments and is targeted at object-oriented paradigm solutions and with XP- an iterative agile methodology for small project teams.

Our research revealed that SOUP methodology has at least few big drawbacks:

- it does not extensively address project risk management activities as RUP methodology suggests;
- it does not include any project iteration planning activities likes the ones described in XP *Planning* phase, as a result XP *Planning Game* practice is not included in SOUP;
- it does not include software release management activities as RUP methodology suggests;

All these three SOUP lacks are of great importance for successful SOA development, because SOA projects are significantly more complex than typical software projects, they require larger, cross-functional team, the use of new unexplored technologies and tools, iterative development with frequent software releases resulting in a higher risk of failure than other traditional software development projects.

To sum up, listed drawbacks makes SOUP methodology immature and in need of reconsideration and improvement. At this moment, SOUP methodology can be qualified only as a candidate methodology for SOA development. It should be revised, tested and improved to assure successful SOA implementations.

References

Beck, K. (2000). Extreme Programming Explained: Embrace Change 2nd Edition Addison-Wesley.

Kolawa, A., Huizinga, D. (2007). Automated Defect Prevention: Best Practices in Software Management. Wiley-IEEE Computer Society Press.

Krunchen, P. (2000). The Rational Unified Process, An Introduction, Second Edition, Addison Wesley.

Lewis, G., Smith, D. B., Kontogiannis, K. (2010). A Research Agenda for Service-Oriented Architecture (SOA): Maintenance and Evolution of Service-Oriented Systems. SEI. http://www.sei.cmu.edu/reports/10tn003.pdf

Mamaghani, N. D., Mousavi, F., Hakamizadeh, F., Sadeghi, M. (2010). Proposed Combined Framework of SOA and RUP. Information Sciences and Interaction Sciences (ICIS), 2010 3rd International Conference on. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5534806

Mittal, K. (2010). Service Oriented Unified Process (SOUP). http://www.kunalmittal.com/html/soup.html

SOAPRPC (2004). Service-oriented architecture: A brief introduction, http://soaprpc.wordpress.com/2009/05/06/service-oriented-architecture-a-brief-introduction/

S. Svanidzaitė is a doctoral student at Vilnius University Institute of Mathematics and Informatics. Her main research interests include software processes, service-oriented architectures, software requirements engineering, especially service-oriented requirements engineering and service-oriented architectures design and development methodologies.

AN APPROACH TO SOA DEVELOPMENT METHODOLOGY: SOUP COMPARISON WITH RUP AND XP

Sandra Svanidzaitė

Summary

SOA - tai naujas programinės įrangos architektūros modelis, kurio pagrindinė sudedamoji dalis yra paslauga. Paslauga – tai pasikartojanti verslo užduotis. Į paslaugas orientuotos architektūros kūrimo metu visi verslo reikalavimai yra grupuojami į vieną nuo kitos nepriklausančias paslaugas, šios paslaugos yra komponuojamos tarpusavyje. Tai nauja programų sistemų kūrimo paradigma neturinti jai pritaikytų metodikų, kurios užtikrintų sėkmingą SOA kūrimą. Šio straipsnio tikslas yra palyginti tris metodikas: RUP metodiką, kuri yra skirta objektiškai orientuotos paradigmos sistemoms kurti, agilią XP metodiką, ir SOUP metodiką, kuri yra sukurta RUP ir XP metodikų pagrindu, siekiant išsiaiškinti ar SOUP metodika yra pakankamai išsami ir tinkama SOA sistemoms kurti.

Pagrindiniai žodžiai: SOA, RUP for SOA, SOUP, SDDM.