Developing a Legacy to SOA Migration Method

G. Reijnders

R. Khadka

S. Jansen

Technical Report UU-CS-2011-008
April 2011

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Developing a legacy to SOA Migration Method

Gijs Reijnders, Ravi Khadka, Slinger Jansen, and Jurriaan Hage
Institute of Informatics and Computing Sciences
Utrecht University
Utrecht, The Netherlands
{gwreijnd, ravi, s.jansen, jur}@cs.uu.nl

Abstract

This document proposes a legacy to SOA migration method that can be used to extract services from legacy systems. We use the method engineering approach followed by the program slicing and concept assignment approaches to extract services from legacy codes. The proposed method is developed by reusing existing and proven parts from existing Service-oriented methodologies, which not only saves time and reduces the adoption and introduction problems. As per our knowledge, such a method is missing until now. Our proposed method has been initially reviewed by the experts and the method was enhanced as per the findings of the experts review. Finally, the method has been successfully evaluated with a case study.

1. Introduction

The Service Oriented Computing (SOC) paradigm uses services to support development of rapid, low-cost, interoperable, evolvable and massively distributed applications (Papazoglou, 2007). Key to realizing service oriented computing is Service Oriented Architecture (SOA). Market studies have shown that SOA is increasingly considered and implemented in businesses (Gartner[a], 2009; Gartner[b], 2009). Those businesses that implement a SOA open their borders for the purchasing and integration of third party services (Erl, 2004). These developments gave rise to what we refer to as service markets, for example the market surrounding Salesforce.com. Service markets are still young but already provide business opportunities for product software companies. Managers of those product software companies that want to enter the service market(s) perceive a cost and experience barrier for the development of a service portfolio. One possibility to lower or remove those barriers is to reuse their existing software features as services to, for example, generate additional profit. We reason that a method to efficiently extract software services from existing monolithic software products aimed at managers, provides them with a structured approach to exploit the business opportunities the service markets offer. Additionally we expect that by simplifying the creation of services we can bootstrap service markets by populating them with extracted services increasing the opportunities the service markets offer for other companies.

In this report, we focus on the development of a method that can be used by managers of product software companies to extract services from their monolithic software product based on their existing well tested code. We use method engineering (Brinkkemper, 1996) followed by the program slicing (Weiser, 1981; Harman et al., 2002) and concept assignment (Biggerstaff et al., 1993) approaches to extract the services from the legacy codes.

The report proceeds as follows: Section 2 presents the background of the underlying concepts of method engineering, program slicing and concept assignment. Section 3 describes the method engineering approach to develop the legacy to SOA migration method. Section 4 briefly presents the evaluation of the proposed approach and finally, Section 4 presents the conclusion and future works.

2. Background

In this section, we describe the method engineering approach, program slicing and concept assignment approach as the foundation of our method.

2.1 Method engineering approach

No method can be defined that is well suited in every situation, as a solution to this problem methodology engineering was introduced (Kumar and Welke, 1992). Methodology engineering or method engineering as it is referred to in more recent literature (Brinkkemper, 1996 ; Ralyté, 2003).

> *Method engineering: "is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems". (Brinkkemper, 1996)*

In literature different method engineering approaches exist. Brinkkemper (1996) approach to situational method engineering is depicted in Figure1.

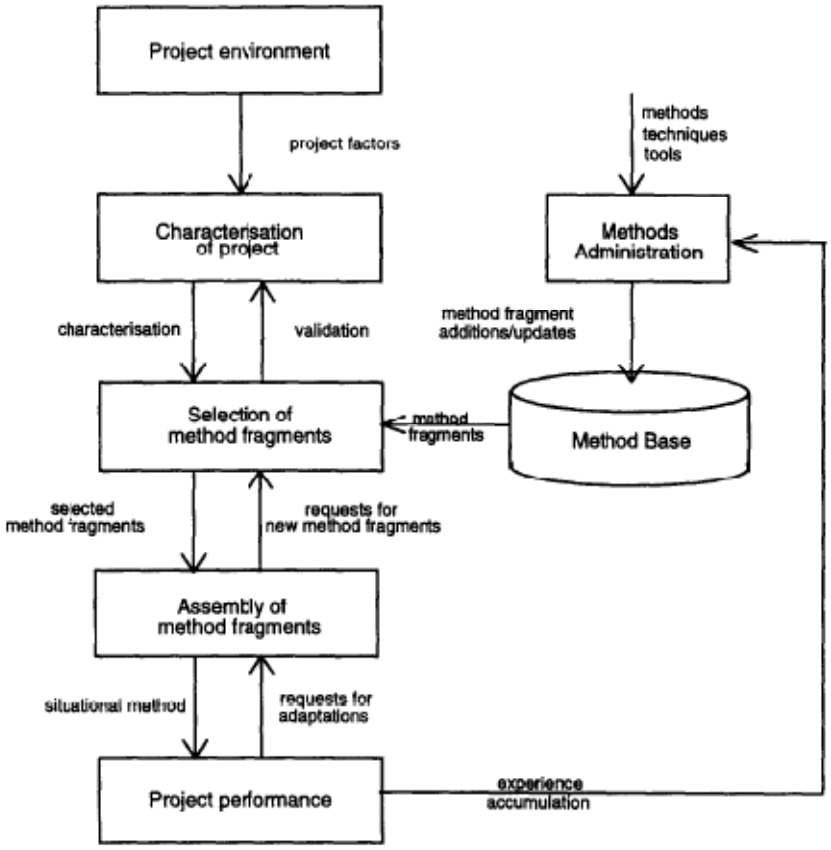

Figure1 : Configuration approach for situational methods (Brinkkemper, 1996)

Ralyté et al. (2003) describes a similar approach, as depicted in Figure 2, to Brinkkemper (1996) with the addition of defining tree different engineering strategies for situational method engineering.

1. **Assembly based:** reuse of method components extracted from existing methods to create a new method or enrich an existing one.
2. **Extension based:** extend a method by applying extension patterns.
3. **Paradigm based:** creating a new/fresh method from a paradigm model or meta-model.

Each of the three engineering strategies can be used separately or in combination.

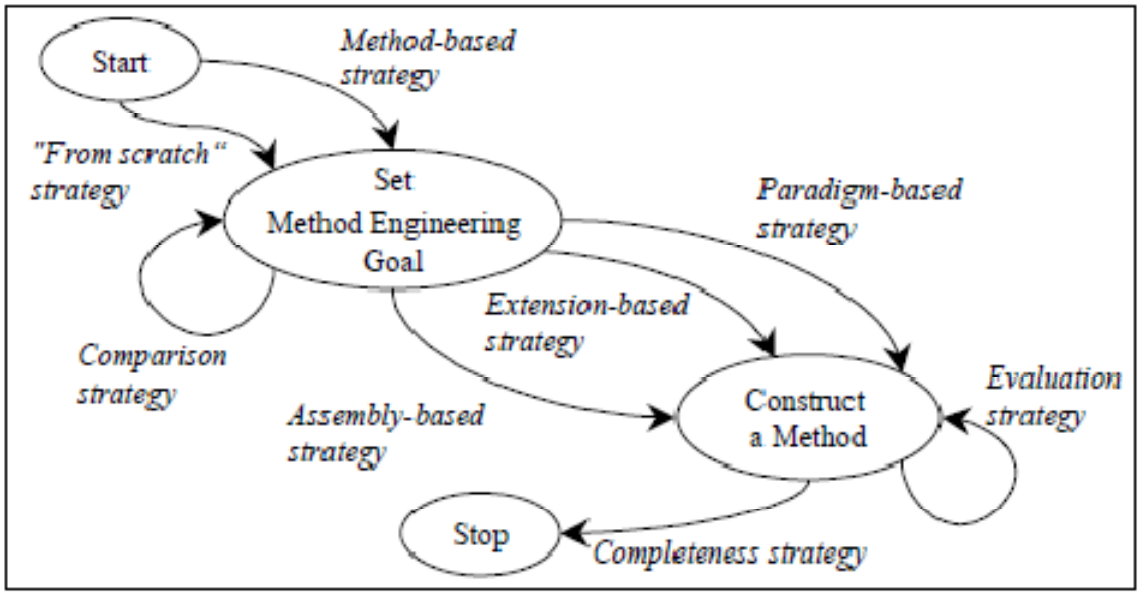

Figure 2 : Generic process model for situational method engineering (Ralyté et al., 2003)

Both the approach of Brinkkemper (1996) and Ralyté et al. (2003) make use of method chunks or fragments; we adopt the term (method-) fragment. Method fragments are parts of existing methods that are stored in a so called method base to be used in the development of new methods. In order to compare and select method fragments in a scientific verifiable way a specific tool is needed, such a tool is meta-modeling (Brinkkemper, 1996). Weerd et al. (2009) describe a meta-modeling technique based on UML, this technique depicts a method (-fragments) in a Process-Deliverable Diagram (PDD). A PDD consists of two parts a process model, left side, (UML activity model) and the deliverable model, right side, (UML class diagram). An example PDD is depicted in Figure 3. compared to standard UML two major adjustments have been made.

1) **Unordered activities:** are used when activities within a method have no predefined order in which they have to be performed.
2) **New types of concepts**
    a. Open: A collection of (sub)concepts that are explained in more detail elsewhere
    b. Closed: A collection of (sub)concepts which is deemed unnecessary to be explained in the current context.
    c. Simple: a concept that does not contain any other (sub) concepts.

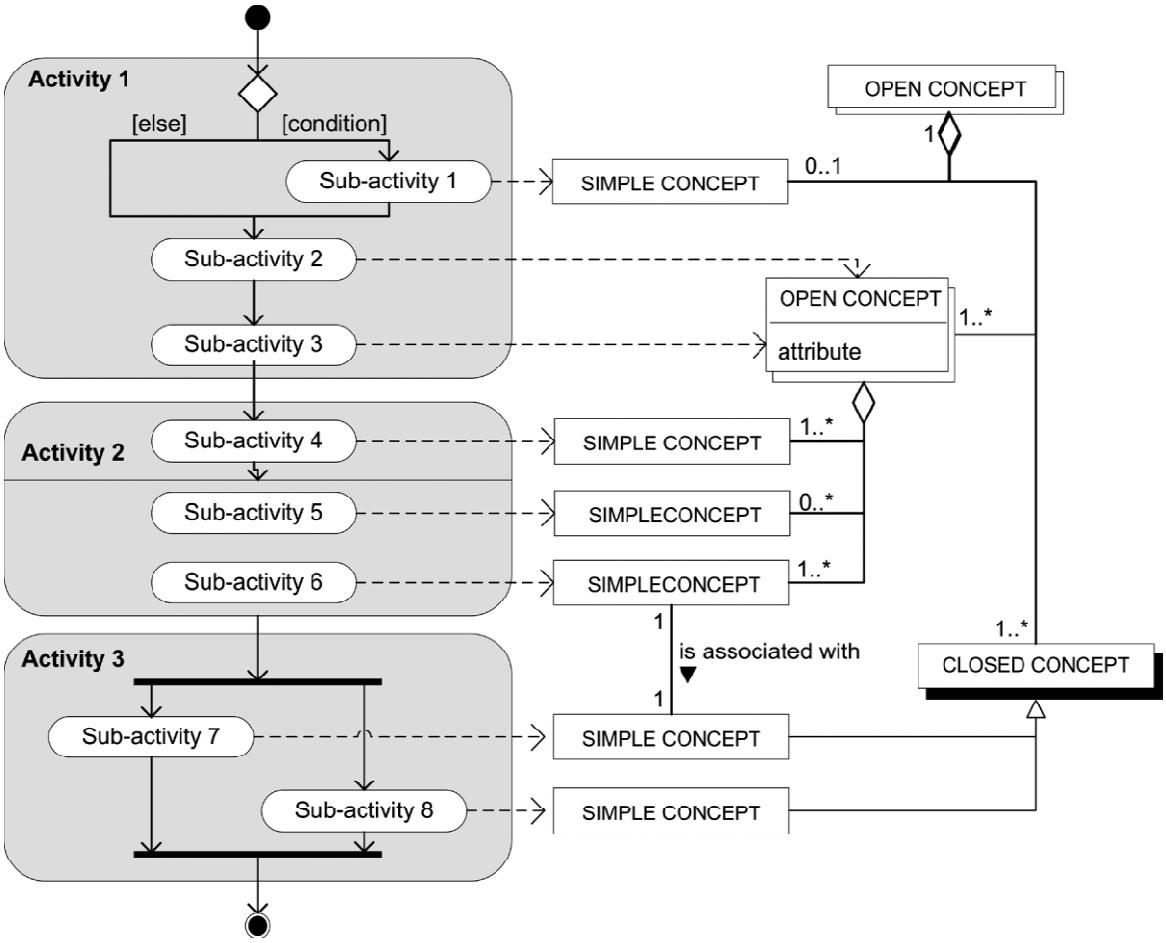

Figure 3 : Process-data diagram (Weerd et al., 2009)

For more information on meta-modeling we refer to Weerd & Brinkkemper (2009).

Looking ahead we intend to extend current literature with a new method. In order to develop a new method we have to select a strategy and approach. First we select a strategy. Comparing the goals for the proposed method to the situational method engineering strategies by Ralyté et al. (2003) we reason that assembly based situational method engineering should be used to adopt and adept parts of existing methods to ensure that we develop a method that integrates well with current practice. Because, by reusing parts of existing methods we ensure that a) the method is easy to use and understand by anyone that has worked with other software development methods; b) we use proven activities and phases reducing the need for testing and increasing the likelihood of a successful method; c) current methods already implement best practices and industry standards that can be adopted in the proposed method.

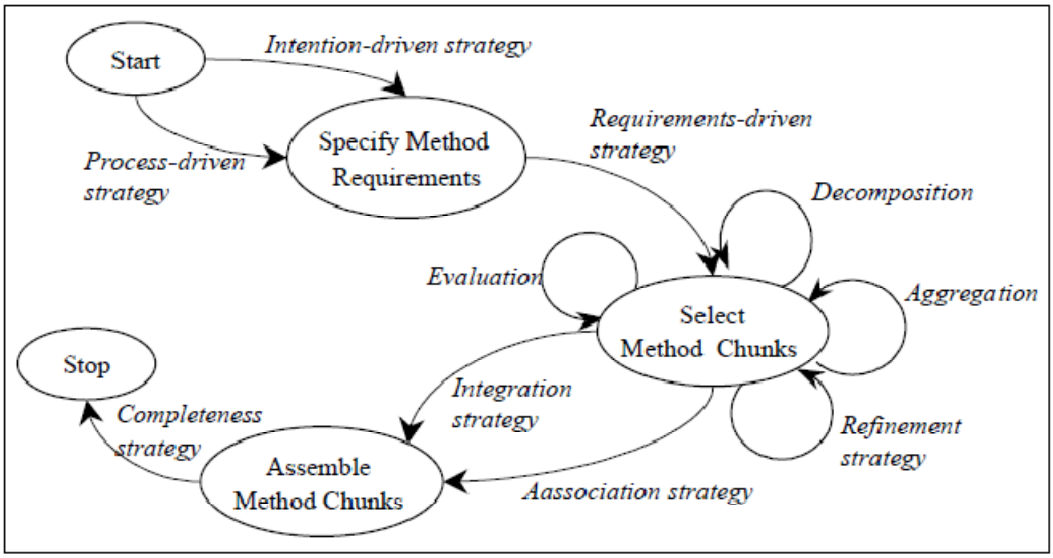

Figure 4 : Assembly-based process model for situational method engineering (Ralyté et al., 2003)

Figure 4 shows the process model for the assembly based situational method engineering approach. Where the first step is to 'specify the requirements' the method has to fulfill. Two strategies are identified namely:

1) Intention-driven: for method adaptation to adopt, enhance and/or limit the selected method
2) Process-driven: for new method construction.

The second step is the 'selection of method chunk' (or fragments). Using several measures a selection is made to determine what fragments can be used to satisfy parts of the requirements set for the method. The last step is to 'assemble the method fragments' into a new situational method. There are two available strategies to be executed. First, fragments have totally different goals, e.g. one fragment uses the result of the other. Second, fragments are similar and the commonalities have to be identified and merged.

The method engineering approaches detailed above assume that the method base is filled. Our search for a method base with service-oriented method fragments led to a dead end. Fortunately, Weerd et al. (2006) propose different generic approaches that incorporate the lack of a filled method base. They have applied their approach in an assembly based method engineering project to develop a situational method for the implementation of web-based content management system-applications (Weerd et al., 2006). The four steps of the approach are detailed below.

1) Analyze implementation situation.
2) Select candidate methods that meet one or more aspects of the identified needs.
3) Analyze candidate methods and store relevant method fragments in method base.
4) Select useful method fragments and assemble them in a new method by using route map configuration to obtain situational methods.

The similarity between the starting situation for project of Weerd et al. (2006) and the proposed method project advocates the adoption of their approach. Combining method engineering, meta-modeling and the approach by Weerd et al. (2006) provides sufficient tools to develop a methodically sound and valid method.

When considering what methods can be used to extract fragments from Papazoglou & van de Heuvel (2006) state that, even though Object Oriented (OO) development and Component Based Development (CBD) methods are not well suited for SOA and web-service development limiting our scope to service-oriented methods. However, they also state that OO and CBD can still serve as a basis for new SOA/web-service methods. The basis for software engineering we base our research on is based on the book of Sommerville (2007) defining four general phases:

1. *Requirement engineering*: Define what requirements need to be fulfilled.
2. *Design*: Make a functional and technical design how to realize the requirements.
3. *Develop*: Realize the design.
4. *Verify and validate*: Test the realization versus the set requirements.

To determine if the phases of OO and CBD software engineering and existing service-oriented methods are indeed similar we compare several such methods against the four phases from Sommerville (2007) and analyze if they indeed follow a similar process. Five methods for service-oriented development where found in literature:

1. Service Oriented Modeling Architecture (SOMA) (Arsanjani et al., 2008).
2. Service-Oriented Design and Development Methodology (SODDM) (Papazoglou & van de Heuvel , 2006).
3. SOCRATES™ (van Es et al., 2005).
4. Web-service implementation methodology for SOA application (WSIM) (Lee et al., 2006; OASIS, 2005).
5. FREMA method for describing Web-services in a Service Oriented Architecture (SRI-DM) (Millard et al., 2006).

To create an overview we conduct a comparison on the following four subjects:

1. *Phases:* Which phases are performed when executing the method?
2. *Paradigm*: Which development paradigm(s) are used? e.g. waterfall, incremental, agile.
3. *Orientation*: From which perspective is the development approached?
4. *Goal*: What is the reasoning for this method to be created and which issues does it address?

Method	Phases	Development paradigm	Modeling paradigm	Goal
OO and CBD based software engineering	1. Requirement engineering 2. Design 3. Development 4. Verify and validate	• Waterfall • Agile • Incremental /iterative	Object Component Aspect	Developing software to tackle a certain problem.
Service responsibility and interaction design method (SRI-DM)	1. Defines scenario 2. Create services set 3. Represent high level overview 4. Order services to fulfill a scenario.	• Agile	Service-oriented	A lightweight development method to create web services based on scenarios.
Socrates™	1. Positioning 2. Scoping 3. Planning 4. Realize 5. Maintain	• Agile	Service-oriented	Implementing SOA within a company.
Service-oriented Modeling Architecture (SOMA)	1. Business modeling & solution management 2. Identification 3. Specification 4. Realization 5. Implementation 6. Deployment/monitoring and management.	• Risk driven • Incremental /iterative • Fractal	Service-oriented	Increase business agility and integration to implement a service-oriented solution.
Service-Oriented Design and Development Methodology (SODDM)	1. Planning 2. Analysis & design 3. Construction & testing 4. Provisioning 5. Deployment 6. Execution & monitoring	• Incremental / iterative	Service-oriented	Design and develop service-oriented solutions. Based on theory surrounding web-services and processes principles.
WS implementation methodology (WSIM)	1. Requirements 2. Analysis 3. Design 4. Implementation 5. Testing 6. Deployment	• Agile	Web-service	Complement existing agile methods to take into account web-service best practices.

Table 1 : Overview of service-oriented methods

Based on Table 1 we deduce that service-oriented methods:

1. Have similar phases both between themselves and compared to the generic phases of OO/CBD based methods.
2. Reason around implementation of the service-oriented paradigm within the company.
3. Use an incremental/iterative or agile development approach.

We now detail each of the service-oriented methods mentioned above. Based on their available documentation in the references mentioned.

Service oriented methods

In this section we detail each of the service-oriented methods mentioned in Table 1 to develop an understanding of service-oriented methods.

Service Responsibility and Interaction Design Method

The following description is based on the paper of Millard et al. (2006). Service responsibility and interaction design method (SRI-DM) is an agile method to develop abstract services independent of implementation. It advocates minimal documentation and fast iterations to deliver artifacts with a small team. SRI-DM defines 4 steps.

1. Define a scenario with a use case diagram
2. Use the individual use cases to factor a set of services
3. Represent these at a high level using service responsibility and collaboration cards (SRCs)
4. Define how SRCs can interact to fulfill the scenario using a sequence diagram

SRI-DM starts by defining a problem scenario presented by an UML use-case diagram. A scenario may consist of more than one service, each separate service is described at an abstract logical level presented using service responsibility and collaboration cards (SRC's) referred to as service profile. Service profiles are treated as atomic to ensure atomic services are developed. Finally, UML sequence diagram are used to detail the flows and relations between services implement a scenario.

Socrates™

The following description is based on the book of van Es et al. (2005). Socrates is an agile approach to implement SOA based solution. It is developed by several SOA experts based on their experience in the field of SOA implementations. Socrates defines five phases:

1. Positioning - Is conducted to identify if a SOA is a viable solution to implement. The following aspects are analyzed to determine the validity: bottlenecks in the information system and business processes, business goals and strategy. If SOA appears to be the desired solution for the problem at hand the current starting situation is globally documented in a business case.

2. Scoping - The scope for the rest of the project is defined. If a scope has already been defined it is reevaluated on completeness and correctness for the current project. In case no scope was defined new enterprise architecture is defined depicting: business, processes, information, applications and technology from which the scope is defined.

3. Planning - The planning phase defines the path from the as-is to the to-be situation. Projects are defined to guide the changes required in the business, processes, information, applications and technology.

4. Realize - This phase is concerned with executing the project planning defined in phase 3. New insights acquired during the execution are used to update the project plans

5. Maintain - Finally, the last phase is concerned with maintaining the implemented solution as changes and updates are always required.

Service Oriented Modeling Architecture (3.1)

The following description is based in the paper of Arsanjani et al. (2008) SOMA is created based on SOA design and implementation projects executed by IBM. It aims to define key-techniques, prescriptive tasks and guidelines for the development phases of a SOA. SOMA uses a non-linear approach to the project, phases are executed in a risk-driven, iterative and incremental approach.

Figure 5 shows the SOMA project phases. The paper only details the following phases:

1. Business modeling and transformation
2. Solution management
3. Identification
4. Specification
5. Realization

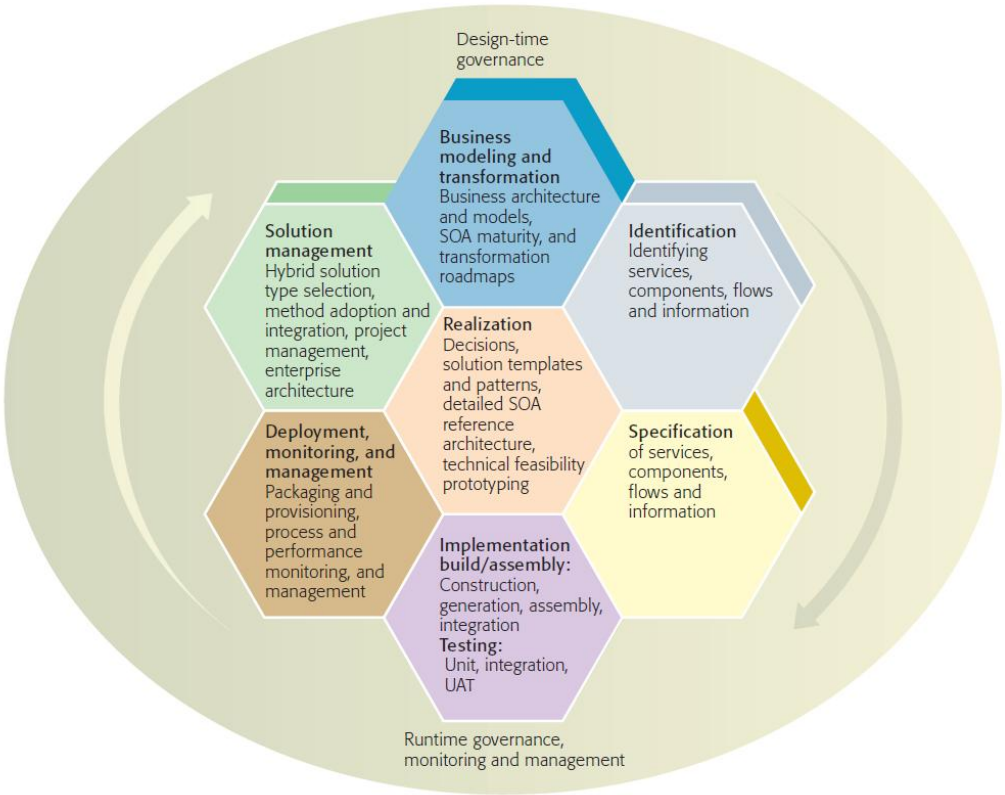

Figure 5 : Service Oriented modeling architecture (Arsanjani et al., 2008)

1. Business modeling and transformation

This phase is used to model the business and then simulate and optimize to identify focal areas for transformation. It is noted that this phase is not mandatory but highly recommended.

2. Solution management
SOMA defines several so-called solution templates that are used to adopt the method depending on the expected solution type. Several example solution types are custom development, legacy integration, transformation, and package application integration. Depending on the expected solution, specific templates are used to specify the rest of the project.

3. Identification
Identification focuses on identifying candidate services, components and flows. The authors recommend the use of complementary service identification techniques to generate better results than when using a single technique. Several techniques are provided:

- **Goal service modeling**
  This approach decomposes business goals into smaller sub-goals until they reach a level where the goals can be achieved by a service.
- **Domain decomposition**
  Divide the business domain into smaller functional areas which form the basis for services.
- **Existing asset analysis**
  Analyze existing assets such as applications, services, API's and systems to see what can be reused.

After all candidate services have been identified three steps are performed. First low-level services are refactored, where they are grouped according to their logical affinity into a higher-level service. Secondly, each service is tested against the set business/project goals using a services litmus test (SLT). SLT tests each service against filters that dictate what is expected of a service, based on the results candidate services are selected to achieve the goal set for the project (Zimmermann et al., 2005; Erl, 2004). An example of such a filter: processes and/or functions that facilitate data exchange between applications are good candidate services. To conclude the phase, the service-model is reviewed by a panel of stakeholders. They determine if the candidate services within the model are relevant to the project. Any service found irrelevant is discarded.

4. Specification
The specification phase focuses on designing the architecture on a high level as well as parts of the detailed design. The following subjects are detailed: service dependencies, flows, composition, events, rules and policies. Before the actual designing process is started two preparatory activities are performed. First, the information models are specified and elaborated. Entity relationship diagrams are used to define the business entities and their structure and relationships. The conceptual data model is remodeled to a logical data model to define the attributes relevant to the domain. Secondly, a fine-grained analysis and specification of existing assets is performed.

When both preparatory activities are completed the service specification begins. The service specification depicts "The dependencies of services on other services, components, or applications, the composition of services, and the flow among services together define the realization and choreography of services to enable business functions and processes". Another addition to the service model, are the service operations that show the IT implementation of a business function. These operations should be

depicted using the service case model. A service case, compared to a use case, does not depict a static interaction between actor and system but shows an ecosystem of providers and consumers where the roles are interchangeable. Providing an overview of "business aligned IT services that collectively enable the fulfillment of business goals". Finally the service case also contains the non-functional requirements to provide a measure for resources allocation services.

5. Realization

In the realization phase, decisions are made as well as validated using extensible prototypes that outline the decisions made in the realization decisions.

Service-Oriented Design and Development Methodology

The following description is based in the paper of Papazoglou & van den Heuvel (2008) SODDM aims to deliver a "services design and development methodology that provides sufficient principles and guidelines to specify, construct and refine and customize highly volatile business processes choreographed from a set of internal and external Web services"

The method contains six steps of which only three are discussed in detail in the paper: planning, analysis & design, construction & testing. The method is depicted in Figure 6.

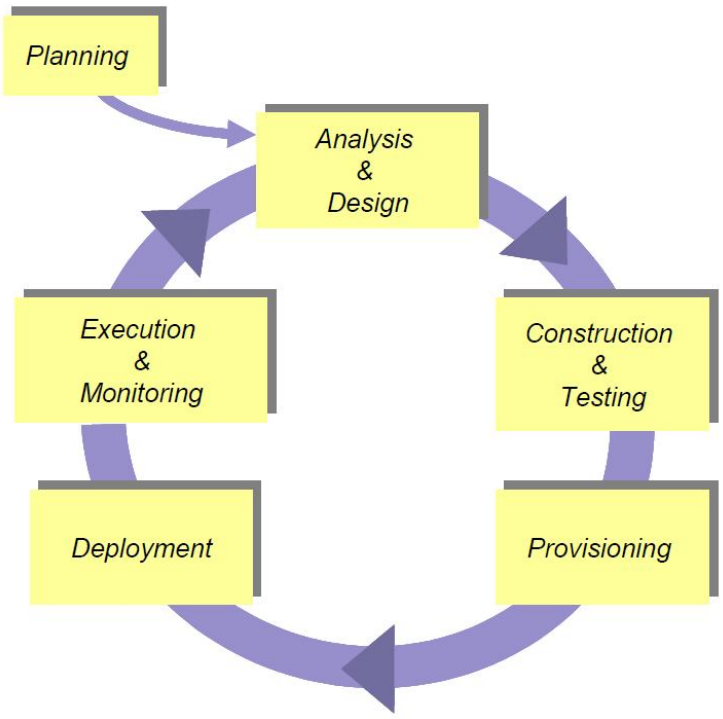

Figure 6 : Service Oriented Design and Development Methodology (Papazolglou et al., 2006)

The planning phase:

In this phase the initial project scope and planning is made as well as a feasibility study is conducted to evaluate the use of the project. This phase has high similarity to the planning phase of Rational Unified Process (RUP) and was not detailed further.

The analysis phase:
This analysis focuses first on an "as-is" model to depict the current infrastructure, after which a "to-be" model is designed to show the intended effects of the implementation of SOA principals. To create these models four steps are conducted.

1. *Process identification:* In this activity a clear understanding is generated about what the company does in terms of business processes and services required to perform them.
2. *Process scoping:* To prevent making services large and monolithic a clear understanding of the scope of a business process is required.
3. *Business GAP analysis:* Applying gap analysis to incrementally add more details to an abstract service/process interface. It results in a recommendation to develop, reuse or purchase services from third parties.
4. *Process realization analysis:* Based on evaluation criteria a business process realization scenario is created. Four realization options are commonly considered (or a combination there of)
    - Green-field development: describes how the interface of a new web services forms the basis for its implementation.
    - Top-down development: New services are developed to conform to an existing service interface.
    - Bottom-up development: developing a new interface for an existing application.
    - Meet-in-the-middle development: an already existing web-interface is mapped onto a new service or process definition.

The design phase:
After the services have been identified they need to be specified to create useful and reliable services design that adhere to the design principles of service coupling and cohesion. Applying these design principals guarantee the design of self-contained, modular service that can be composed into bigger composite services.

Each service specification should consist of the following elements (Johnston, 2005):

- *Structural specification:* define the service types, messages, port types and operations.
- *Behavioral specification*: understanding the effects of how a service operates as well as the semantics of the messages.
- *Policy specification*: denote the policy and constraints on the service(s).

Besides the three specifications given above the service programming style should also be considered. Services are either data or process oriented. While designing a service this should be taken into account. Finally, the business process for which the services where created has to be designed. For each business process the following facts need to be addressed: The process structure, the business roles, and the non-functional business process concerns.

Service construction & testing phase
In this phase the definition of the service interface and implementation description are developed for both the provider and requester. When the implementation has finished the implementation is tested on behavior, performance, and interface and functional.

Web-Service Implementation Methodology (WSIM)

The following method description is based on Lee et al. (2006) & OASIS (2005). WSIM is a method that extends existing software development methods to take into account web-service specifics. An overview of the method is depicted in Figure 7.

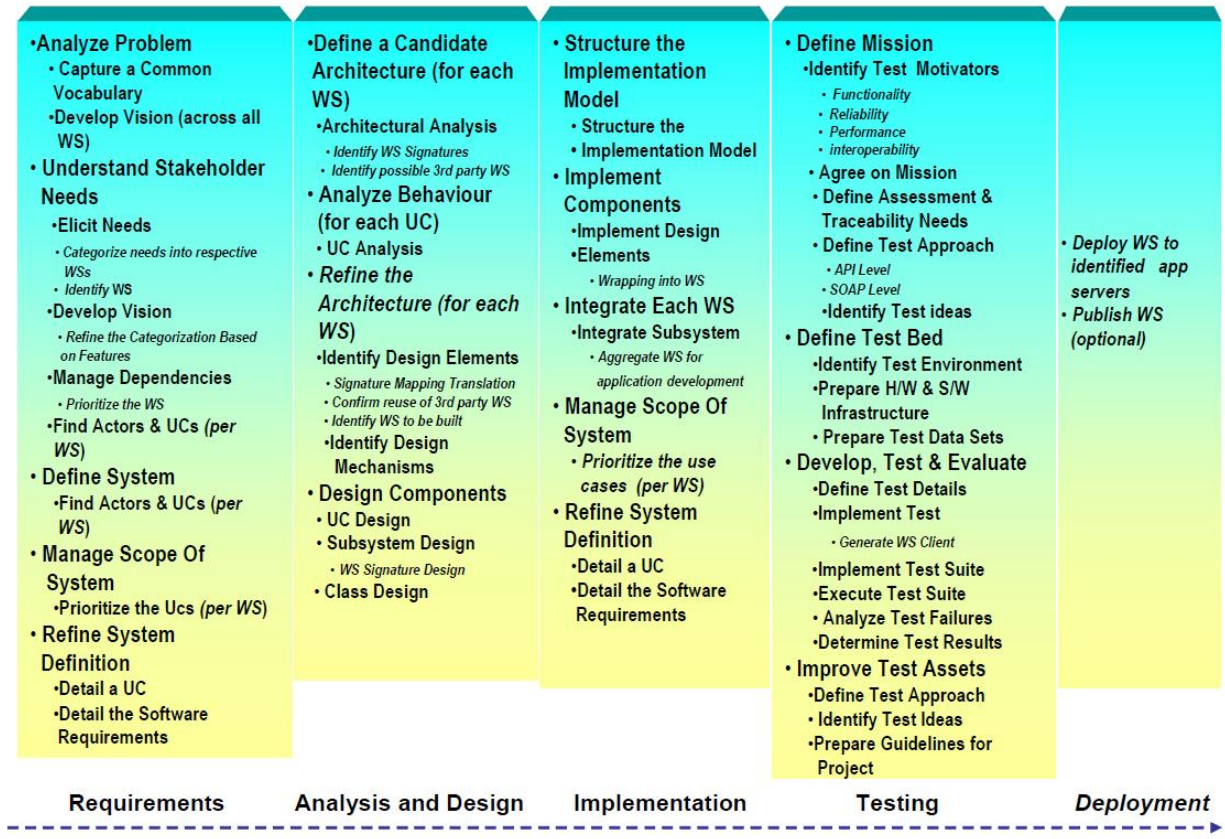

Figure 7 : Web services implementation methodology (Lee et al., 2006)

Requirement phase - The first phase focuses on understanding the business requirements and constraints to translate them into requirements for web-services, both functional and non-functional.

Analysis phase - The analysis phase focuses on refining the initially identified requirements into a conceptual model which defines granularity, implementation platform and architecture decisions.

Design phase - Within this phase the design for each web-service is detailed. The design details both the interface and the interaction between server and client.

Implementation phase - The actual realization of the web-service design is performed in this phase. Coding, generating the WSDL and deploying the web-service to the server are the three steps performed.

Testing phase - In this phase the implemented web-service is tested against the design.

Deployment phase - The actual deployment is tested in this phase.

By detailing the five methods we have created sufficient knowledge and insight into service-oriented methods. From these findings we conclude that existing service-oriented methods can indeed be used as a basis for the development of the proposed method by using them to populate the method-base. However, we think that several specific techniques are required to achieve all the set method goals.

2.2 Program Slicing and Concept assignment

Program slicing is used to strip the original source code of all lines of code not relevant to the variables/computation that is being extracted/researched (Weiser, 1981; Gold et al, 2005; Tip, 1995; Harman et al., 2002). Resulting in a slice containing all the code relevant to certain variables/computations.

> *Program slicing: a technique that removes components from the source code that do not relate to the computation of interest The resulting slice gives a projection of the original programs semantics* (Harman et al., 2002).

Program slicing uses program statements as criteria to determine what code to maintain and what code to remove. However, it is often unclear or uneasy to identify all or the correct statements. Concept assignment is introduced as a complementary technique; together they produce better results than either technique separately and the combination of the two overcomes their individual weaknesses (Harman et al., 2002).

> *Concept assignment:" is a process of recognizing concepts within a computer program and building up an ''understanding" of the program by relating the recognized concepts to portions of the program, its operational context and to one another" (Biggerstaff et al., 1993).*

Different approaches exist for model driven concept assignment. Gold and Bennett (2002) identifies three plausible concept assignment systems:

1. Domain model - The Adaptive Observer (DM-TAO) (Biggerstaff et al., 1993)
2. IRENE (Karakostas, 1992)
3. Hypothesis-based concept assignment (HB-CA) (Gold & Bennett, 2002)

Gold and Bennett (2002) detail several differences between the different systems: DM-TAO is more complex because of its richer concept representation, resulting in a higher cost base and a slower process of application. Both IRENE and HB-CA are easier and cheaper to maintain and work with. Finally, HB-CA is a fully automated process where the other two are interactive. We do not detail the exact working of each system and their underlying algorithm and knowledge base. For a more detailed comparison see Gold (2000).

Weiser (1981) also introduced five slice-based metrics to measure program cohesion and continuity: 1) tightness, 2) coverage, 3) overlap, 4) parallelism and 5) clustering. These metrics provide additional information about the structure of the program and can be used to create better slices. For example slices with only a few lines of difference might be combined to form a single large slice. We think that the combination of concept assignment and slicing provides an adequate solution for the identification and extraction of source code for reuse as web-services.

3. Legacy to SOA migration

Engineering the legacy to SOA migration from scratch is unnecessary or even desirable. Existing service-oriented methods provide fragments that can be reused to develop the proposed method. Several service-oriented methods are discussed in Section 2, most of these methods have their roots in existing software engineering methods such as DSDM and RUP. We reason that by reusing parts of methods that use common software development phases, activities and deliverables the learning curve and adoption barrier for users are reduced or even removed. Secondly, those phases, activities and deliverables have been tested and proven by years of software engineering history.

To ensure proper method design, assembly based situational method engineering is used (Ralyté et al., 2003; Weerd et al., 2006) to guide the process of situational method engineering. The approach proposed by Weerd et al. (2006) is adopted:

1. Analyze situation and identify needs.
2. Select candidate methods that meet one or more aspects of the identified needs.
3. Analyze candidate methods and store relevant method fragments in a method base.
4. Select useful method fragments and assemble them into a new method.

Note that we have made a minor change to the first step where the name “analyze implementation situation and identify needs” was changed to “analyze situation and identify needs” because this method is not concerned with implementations the term was removed. For the fourth step we removed the situational and route map segment because the proposed method does not detail different situations.

3.1 Situation and Needs

The start of the method engineering approach is to define the situation and needs that define the proposed method. To clarify the situation several key project characteristics for service-oriented methods, consolidated from the literature study, are identified and compared to the proposed method, depicted in Table 1.

Characteristic	service-oriented methods	Proposed method
Method goal	Guide a company through the process of implementing the service-oriented paradigm.	Identify, design and extract services from existing software monoliths by reusing source code.
Method users	Companies that want to implement the service-oriented paradigm	Product software companies, and managers in particular, that see

		opportunities in reusing existing functionalities as service(s).
Users of the results	The results are, most commonly, used by the company that executes the method to improve business processes they own or take part in.	The main group of targeted users is external parties that are willing to purchase software functionality in the form of services.
Main information sources	Driven by the process models and enterprise architecture.	Software development history and market studies and existing source code.
Service applicability	Most commonly, the services are developed for a specific business process within the implementing company.	The services are developed for, probably unknown, external processes.
Distribution	The services are made available internally or within the business process boundaries.	The services are published for use by external third parties.

Table 1 : Method situation comparison

The difference between the situations shows how the proposed method should differ from the existing service-oriented methods and what parts can be reused. The focus for this research lies with the identification of candidate services and source code extraction for the development of those services. Based on the insight required by conducting the literature review and the situational differences compared to other service-oriented methods several needs and constraints are defined. These needs and constraints influence the development decisions made for proposed method. Table 2 describes the needs and constrains define for the proposed method. Furthermore, several needs for the results of the proposed method are also defined in Table 3. Several of the needs are defined based on the literature review. First, Papazoglou and van den Heuvel (2006) note that OO and CB development are good basis for service-oriented methods. They also note that the service-oriented principles, mentioned in the literature review, should be adhered to by the method. Secondly, Arsanjani et al. (2008) state that complementary techniques should be used to identify candidate services. Thirdly, the OASIS group (2005) note that by extending existing methodology they can reduce the adoption barrier by making the method easier to use and understand. Fourth, Millard et al. (2006) apply UML to develop service scenarios. We reason that existing techniques can be applied to develop services.

	#	Description
Method needs	N1	The method is easy to understand and use.
	N2	Not all identified services have to be delivered in a single release.
	N3	The method needs to incorporate service-oriented design principles.
	N4	The method is not company specific.
	N5	The method is not program language specific.
	N6	The method allows users to apply their own tools and techniques when appropriate.
	N7	The method contains activities concerned with the identification and extraction of source code.
	N8	The method can be used and understood by managers of product software companies.
	N9	Market research has to provide additional information in regard to requirements.

	N10	Services need to be identified selected and designed before extraction takes place.
	N11	Candidate service selection should be performed with complementary techniques.
	N12	Services identified for which no source code is available are marked as out of scope of the proposed method. However, they do influences candidate services selection when they influence the development process of a candidate service.
Method constraints	C1	The software monoliths needs to supports code extraction.
	C2	The market has to support service distribution and purchasing.
	C3	The method does not detail service development from scratch.

Table 2 : Needs and constraints for the proposed method

	#	Description
Needs for the method results	R1	Source code should be extracted to form concept slices that are preferably executable.
	R2	The service design details the structure, behavior and policy.
	R3	A project plan is created that details the project from a managerial perspective.

Table 3 : Needs for the method results

Based on the situation, needs and constraints detailed in this chapter the next activity in the assembly process can be performed.

3.2 Method selection

In order to continue with the development of the proposed method the method base needs to be populated. No existing method base, or parts there off, could be found that already incorporate any of the methods discussed in Section 2. Therefore the method base has to be populated from scratch. The first step in this process is to select the methods to populate the method base, a total of five methods were identified in the literature review each is analyzed using the following criteria and considerations.

- Acceptance in literature based on citations and publication. Papers only get published after peer reviews and the number of citations to the paper can be seen as a, maybe somewhat unreliable, scale for how embedded the method is in literature.
- Completeness of the method. Not all methods fully explain each phase. The methods those are interesting for the proposed method need to at least detail the analysis, design and development phase.
- Based on existing OO en CBD methodology. Both OO and CBD methods have been long since accepted by the software development community and the level of communality with such methods should increase ease of use and acceptance with the users of those already existing methods.

The first criterion is analyzed by the citation count from Google scholar and the source in which the paper is published. The second criterion filters out the methods that have insufficient documentation of their activities and deliverables. Without sufficient documentation the Process Deliverable Diagram (PDD) created will lack detail making it difficult to use them from method comparison later on in the

process. Finally, the third criterion is analyzed by comparing the phases identified in Table 1 compared to the software engineering phases (Sommerville, 2007) at the top of the same table. The result of the comparison is depicted in 4.

Criterion	SOMA	SODDM	Socrates™	SRI-DM	WSIM
Citations	62 (2008) 231 (2004)	143	unknown	2	13
Publication	IBM system journal	International Journal of Web Engineering and Technology	Book	Unknown	OASIS website and Proceedings of the International Conference on Industrial Informatics
Complete phases	Identification Specification Realization	Analysis & design Construction & Testing	All	Analysis and Design	All
Similarity to OO and CB	High	High	Low	Low	High

4 : Method selection comparison

Based on 4 the following three methods were selected:

1. WSIM: Web-service implementation Methodology for SOA application (Lee et al., 2006; OASIS, 2005).
2. SODDM: Service-Oriented Design and Development Methodology (Papazoglou & van de Heuvel, 2006).
3. SOMA: Service Oriented Modeling Architecture (Arsanjani et al., 2008).

Because:

- Both SOMA and SODDM have the highest score on the citation score and all three methods are either published in a renowned journal or conference publication.
- All three methods have high similarity to OO and CBD software engineering methods.
- Both SOMA and SODDM do not provide rich detail for all phases but the important phases are well documented. WSIM provides clear detail for all phases.

The other two methods where discarded because:

- SRI-DM was both lacking in similarity to existing engineering methods as well as its acceptance in literature.
- Socrates™ was not similar to existing OO and CBD methods either.

Weerd et al. (2006) used a total of three methods to populate their method base. Reasoning that with similar numbers similar results can be generated the three methods selected are sufficient.

Our method base can be found in appendix A. Every method is depicted as a whole, where each phase and activity can be considered a separate method-fragment. We opted to depict methods as a whole to maintain a clear overview of the method.

3.3 Method assembly

In this section we engineer our proposed method, in correspondence to the fourth step of the approach adopted from Weerd et al. (2006).

3.3.1 Method phases

When analyzing the method base for adoption of method-fragments the conclusion was drawn that the reusable method-fragments were all at activity and deliverable level, no phase as a whole could be adopted. To accommodate this fact and ensure the correct phases are present within the proposed method, the first step is to define the phases of the proposed method.

	WSIM for SOA	SODDM	SOMA
1	Requirements	Planning	Business modeling and transformation
			Solution management
2	Analysis	Analysis & Design	Identification
3	Design		Specification
4	Coding	Construction & testing	Realization
	Testing		Implementation
5	Deployment	Provisioning	Deployment, monitoring and management
		Deployment	
		Execution & monitoring	

Table 5 : Method phase comparison

To determine the phases for the proposed method, the methods in the method-base are compared to analyze similarities between their phases. The phases found to be similar in intent are grouped together to form the phases of the proposed method. The result of the comparison is depicted in the Table 5 from which we conclude that the phases between the different methods are similar allowing for the identification of the phases for the proposed method. We define and name the following phases for the proposed method:

1. Project initiation phase
2. Candidate service identification
3. Service specification
4. Service construction and testing
5. Deployment, monitoring and management

Each of the numbers corresponds with the same number in Table 5 and the phases behind that number.

After the phases are defined the activities that can be reused from the existing methods were identified and used to fill in the phases. The selection is conducted by comparing every activity against the situation and needs defined for the proposed method. Some activities were outside the scope of the proposed method and for that reason left out. In case several activities, between the different methods,

had high overlap one activity was selected. The resulting method comparison matrix is depicted in Table 6. The table is created based on the comparison techniques used by Weerd et al. (2007) and Hong et al. (1993). At the end of the matrix we included the legend of how it should be interpreted. Based on the comparison matrix a super-method was created where all the common activities between the three methods are included in a single method. The super-method is depicted in Figure .

SODDM	WSIM	SOMA
1. Project initiation phase		
Analyzing the business needs	Determine the need for web service	-
Review current technological landscape	-	-
Conceptualize the new requirements	Elicit web service requirements	-
	Manage web service requirements	
	Model usage scenarios	
Manage project deliverables and resources	-	Initiate project management
-	Prepare test cases for user acceptance test and system test	-
-	-	Define business architecture and models
-	-	Select solution templates and patterns
-	-	conduct method adoption workshop
2. Candidate service identification		
Process identification	-	-
Process scoping	-	-
Business Gap Analysis	Identify reusable web-services	Analyze existing assets
Process realization analysis	-	-
-	Select a technology platform as implementation framework	-
-	Define candidate architecture	-
-	Decide granularity	Service refactoring and rationalization
-	Identify service interface	-
-	Prepare test cases	-
-	Test bed preparation	-
-	-	Goal service modeling
-	-	Decompose domains
3. Service specification		
Structural specification	Design web services	Specify service
Behavioral specification		
Policy specification		
None functional business process concerns	-	-
Describing business roles	-	-
Describe the business process structure	-	-
-	Transform signature of reusable web-services	Analysis and specification of existing assets.
-	Refine service interface	
-	Refine test cases for functional test	-
-	-	Elaborate and specify information models
-	-	Analyze subsystems
-	-	Specify components
-	-	Refactor and rationalize services
4. Service construction and Testing		

Define service interface	-	Refine and detail components
Service implementation	Construct web-service code	Establish realization decisions
	Construct client code	
Dynamic test	Unit test	-
	Test Functionality	-
	Integration test	-
	System test	-
	User acceptance test	-
-	-	Perform technical feasibility exploration
-	-	Detail SOA solution stack layers
5. Deployment, monitoring and management		
Develop provisioning strategy	-	-
Roll out	Prepare deployment environment	Deploy services
	deploy	
	Publish	
Process implementation	-	-
Monitor and report	-	Monitor and management processes and performance
-	Test deployment	Execute user acceptance test
-	Create end user support material	-

Table 6: method comparison matrix

The matrix should be read as follows:

- Red marked texts are activities that are considered to be outside the scope of the proposed method and should thus not be incorporated.
- Yellow marked boxes are “problem” areas where we were uncertain if the activity should a) be on that position within the method or b) should be incorporated within the proposed method at all.
- Orange marked boxes are cases where similar activities where in two different phases.
- Activities that are similar or that if combined are similar to a higher grained services are depicted on a single row.

Based on the comparison matrix the following decisions were made to develop the super-method.

Project initiation phase:

First we analyzed all activities that were not mark as out of scope, “problematic” or occurring in two different phases. Starting with “analyzing the business needs“ and “determine the need for web-services” two activities that are both similar in intent and placement within the method. “analyzing the business needs“ was chosen to be incorporated into the super-method because its description was on a higher level. Secondly “Conceptualize the new requirements” has three similar smaller granular activities in the WSIM method. To ensure that the method could be applied by as many different product software companies and that those companies can apply their own existing techniques the highest granularity activity was selected. The last of the unmarked activities are involved with project management. Similar to the pervious decision we have selected the activity with the highest granularity because it suits the method needs and situation.

Secondly, three activities were marked as out of scope because they are either involved with method specific activities for the SOMA method or with internal business processes that are not relevant to the proposed method.

Finally two activities were found that did not fit the profile. First the “review current technological” activity in the SODDM method has no similar activity in the phase of either of the other methods in the method-base. However, a somewhat similar activity was present in the second phase of the WSIM method. Papazoglou & van den Heuvel (2006) state that a company has the strategic task to ensure that the service technology fits with the current environment. Agreeing with that statement we reason that the activity should be incorporated and its proper place is within the first phase where many strategic decisions are made. The last activity that was considered was the WSIM activity involved with preparing test cases. For our proposed method it was decided that at insufficient knowledge would be available at this point in the method to develop test cases. For that reason all test related activities were grouped up into a single activity in the service construction and testing phase.

Candidate service selection:
As we did with the previous phase we started with analyzing all the unmarked activities. The first set of activities we encountered were concerned with analyzing the existing assets where assets sometimes existing third party web-services but usually internal software products. The activity from SOMA was selected because of the available description. Furthermore, because we foresee that existing assets are more important in the proposed method than they are in the methods in the method base the activity was moved to be the first in the phase. The next activity is to determine the service granularity based on the activities “decide granularity” from the WSIM and “service refactoring and rationalization” from SOMA. Finally, two more activities from SOMA where incorporated to complete the need from complementary service identification techniques defined by Arsanjani et al. (2008).

For the red marked activities three of them were concerned with the impact of implementing the service-oriented paradigm within a business process. The last activity from the WSIM was concerned with identifying service interfaces is concerned with technical aspects of service development that we do not want to address in this phase.

All yellow marked activities were either concerned with testing and thus skipped or incorporated in the first phase.

Service specification:
For the service specification the first activity selected concerned a granularity decision where all three methods had an activity or a set thereof concerned with designing a service. The highest granularity was chosen because it was both most common between the methods and better suits the needs for the proposed method. The second activity, concerned with the transformation of reusable web-services is considered important considering the situation of the proposed method and possible composite services are likely results of the method.

We had several activities flagged as out of scope for the SODDM these activities all involved business processes. As for the activities in SOMA one of them had no description and for that reason we did not

incorporate it. The other activity was “analyzing sub-systems” where sub-systems are analyzed to create dependency diagrams. We also consider this activity out of scope because the proposed method does not incorporate extraction from none monolithic software products.

The two activities for the SOMA method marked with yellow are considered to be part of the “specify service” activity. This decision is based on the granularity of the activities of the SODDM and WSIM that both incorporate these activities in the specification activity.

Finally, the orange activity “refine service interface” has a similar activity in the construction and testing phase.

Service construction and testing:
For this phase three activities were selected after some granularity decisions and two activities were removed based on the scope. The first selected activity is a combination of the activities in SOMA and SODDM that detail the service interface. The activities both lacked a detailed description however, Papazoglou & van de Heuvel (2006) provide a reference to Brittenham (2001) for interface design. For that reason we adopt their activity. The second activity is simply concerned with programming or realizing the code that implements the design no further clarification about the process is provided. Considering that the proposed method also wants its users to reuse internal techniques we adopted the activity without providing any in-depth description. Finally, we incorporated all testing related activities in the three methods into a single activity “perform tests”.

Both the activities from SOMA marked as red were marked because we simple had no available description of the activity to be able to analyze them.

Deployment, monitoring and management:
As this phase falls outside the scope of this project we have decided to only analyze similarity between the activities and incorporate only the highest level activities. Granularity and scoping decisions were made with no grounded scientific basis. We opted to do this to provide future research with a first step to the deployment phase and clarify that it is in fact part of the whole process.

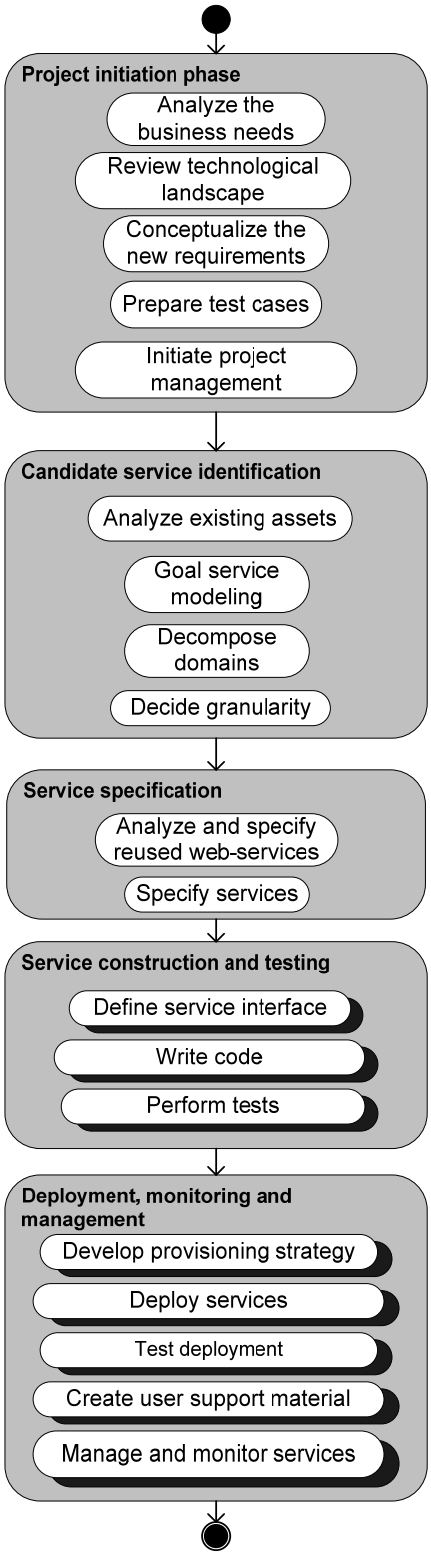

Figure 8 : proposed method version 1

In Figure we depict the basis for the legacy to SOA migration method. The next step in the development of the method is to adapt and extend the basis to reflect the needs and situation defined in the first step. Before we create a model that reflects the changes and additions made to the basis to create the proposed method we detail the envisioned workings of our proposed method in natural language.

3.3.2 Informal method description

In this section we describe, using natural language, the goals and results for each of the phases of the proposed method, as defined in Table 5.

Project initiation phase

The project initiation phase deals with analyzing if the project is a viable business investment. The first activity in this phase is to define the project goals, even though the goals might vary between projects. We assume the proposed method is used for goals that are in line with the extraction of services from existing software monoliths. For example, publishing parts of the existing software as web services on a service market to generate new revenues. The second activity analyzes all technologies that are required to achieve the project goals. These related technologies have to be analyzed to evaluate if the project goals are technologically feasible. Additionally, this activity also includes the analysis of the software monolith from which the services are to be extracted for which the possibility for source code extraction needs to be evaluated. Finally, the industry in which the service is to be implemented is analyzed for existing industry standards and best practices. Adhering to the standards ensures that services are usable by parties operating within the industry. After the project goals have been deemed technologically attainable and the industry standards and best practices have been identified the first high level requirements are identified in the third activity. In order to identify, the high level requirements additional information is required. Existing service-oriented methods analyze business processes and/or enterprise architecture combined with stakeholder interviews to provide the information to identify requirements. However, considering the proposed method aims to develop services for third party customers we foresee complications with the identification of stakeholders and acquire business process and enterprise architecture information. To accommodate this possible lack of information, a preliminary market study is introduced. The execution of the market study is considered out of scope for this research and we leave it to the marketing experts to figure out how to obtain the correct information. However, we provide a global idea what is expected. The main goal of the preliminary market analysis is to evaluate the opportunities the market offers for service and what the expectations are regarding those services. We expect answers to questions like:

- Who will be my customers and what are their expectations?
- Are there competitors on the service market and if so what are they doing?
- What services standards and distributed methods are used in the market?
- How mature is the market in regard to Service Oriented thinking?
- Are there cheaper alternatives?

After both the technological analysis and preliminary market study have been performed the project plan is written in the final activity of the phase. The project plan details the goals and the process how to reach them including the planning, available resources and estimated ROI for the project. When finished

it is presented to the manager(s) or board for approval. At this point the project will either receive a "go" or "no-go".

Candidate service identification

The identification phase focuses on identifying candidate services to satisfy the requirements detailed in the initiation phase. The first step towards identifying candidate services is to analyze and document the current situation. The proposed method prescribes three sub activities to analyze the current situation. The first sub activity analyzes the software development history of the software monolith. Such as: requirements documents, UML diagrams, data diagrams, source code, class diagrams, System Dependence Graphs (SDG) and even comments in code. This information provides understanding about the development history as well as clear understanding of the functionality contained within the software and the reasoning behind its current implementation. Secondly a detailed market study is conducted. The market study is used to further detail the requirements identified in the preliminary market study. Finally, the service market is analyzed to determine what services already exist. Existing services can either provide opportunities to create composite functionalities or already provide similar functionality thus prompting a make or buy decision. Once the "as-is" situation has been defined the next activity identifies candidate services. Identification of candidate services is done by a set of complementary activities:

- The first activity matches functionalities identified in the software monolith against identified requirements. Any match could warrant the creation of a service.
- Secondly, existing external services are analyzed to identify possibilities for service compositions, prompting make or buy decisions.
- Thirdly the mismatch between the requirements and the identified candidate services is analyzed, to estimate how much new functionality has to be developed to achieve the business goals.

After the candidate services have been identified the project is evaluated. Depending on the mismatch between the identified candidate services, requirements and business goals, the project might be canceled. For each of the candidate services the granularity has to be decided. Determining the granularity of the new services comes down to analyzing if candidate services are either part of a composition that should be offered as a single functionality or if a candidate services offer such general functionality that parts thereof can be reused by other services and the service should be split into smaller services. The decision-making regarding granularity is influenced by the non-functional requirements such as speed and reliability. When considering granularity decisions the three different types of services (utility, business and controller) should be taken into account. Coarse-grained services commonly perform better than fine-grained services but offer less opportunity for reuse (Erl, 2004). Resulting in a list of all services with a clearly defined functionality that the service exposes. The relations between the services are depicted in a service infrastructure. No project is ever the same and the amount of candidate services and available resources varies between projects. To incorporate this difference between projects a prioritization activity takes place. Prioritization techniques such as MoSCoW can be used to select the most important services. Based on the priority list the developed iterations can be planned. After each iteration the priority list is reevaluated.

Service specification

The specification phase further details the service cases for the current iteration as well as redesigning any existing third party services. The first activity performed is to analyze the third party services involved with the current iteration. To be able to use the external services they need to be mapped against the existing data types and variable names. This activity provides a clear overview of the data the services require and what output they provide. It also provides an understanding of how those messages relate to the concepts used within the company. Once all the services have been mapped the services are detailed on a technical level. Three steps are performed to complete the service case (Johnston, 2005):

- Structural specification: Defines the service types, messages, and port type. Note that the actual structure of the code is not defined, as it is already present in the existing code.
- Behavioral specification: Entails service operations and semantics of the input and output messages. For example a user cannot delete a record before creating it.
- Policy specification: Denotes policy assertions and constrains for each specific service.

The finished service cases are input for the construction and testing phase.

Service construction and testing

The construction and testing phase is concerned with the extraction of code from the software monolith and the actual programming and testing of the services. The aim for this phase is to apply program slicing and concept assignment to extract functionality in the form of executable concept slices to decrease costs and time spent on development. The first step is to extract code related to the service from the software monolith. For this research we selected the combination of concept assignment and program slicing to create an ECS to facilitate the code extraction. We reason that applying ECS can save a lot of time and resources. The service case details the input and output message and operations this information can be used to assist with concept assignment where a service case describes the concept what is to be extracted. Once concept assignment is finished program slicing comes into play using the variables identified by concept assignment to extract an executable part of the code that implements the concept. When finished, the ECS is the basis for all further programming efforts required to realize the service case. Whichever approach used to guide the programming efforts are left open for the executing party to decide. We reason that as product software companies are targeted they already have reusable procedures and processes in place for this activity. Finally, each service is tested to determine if it is functioning according to the expectations. Similar to the programming activity we expect product software companies to have reusable testing procedures in place. However, we do provide an overview of tests present in the methods in the method base: user acceptance test, system test, performance test, integration / interoperability test, functional test, unit test, deployment test, client test and interface test. When a service passes all these tests it is safe to assume the service does in fact satisfy the service case it implements and the service is ready to be deployed.

Deployment, monitoring and management

Due to the focus of this research on the extraction of services we decided to let deployment, monitoring and management of a service fall outside the scope of this research. The activities and deliverables of

the phase are incorporated into the proposed method only to clarify that although out of scope for this project there are still specific activities required to deploy, monitor and manage the extracted services. These activities should be taken into account, but are out of scope for this research.

3.4.3 Meta Model

To formally detail the proposed method meta modeling, as described by Weerd and Brinkkemper (2008), is used. Figure 9 depicts the actual proposed method resulting from the method engineering approach and adapted based on the feedback from the expert reviews. The activities and deliverables are detailed in Table 7 and Table 11 respectively.

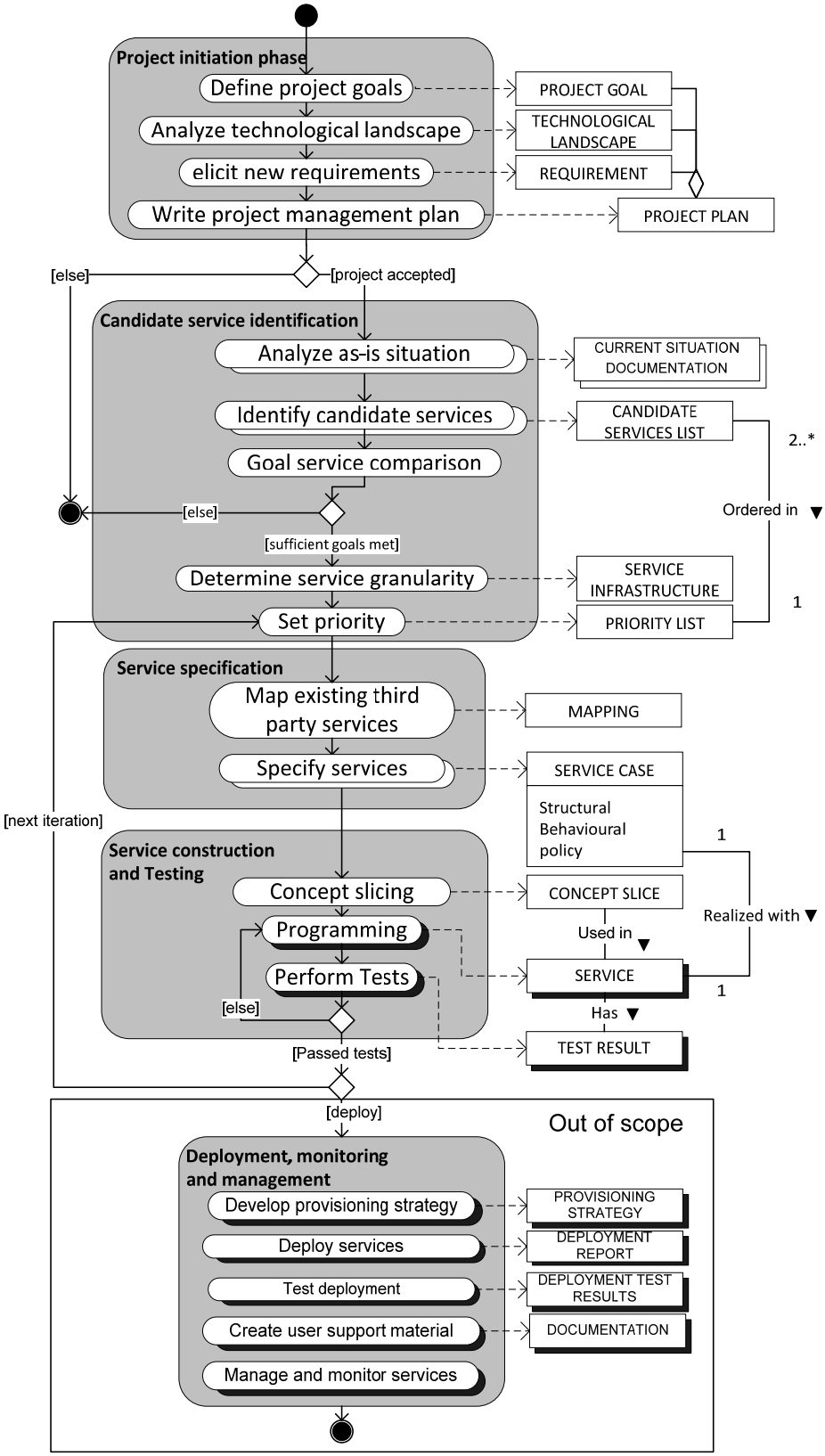

Figure 9 : Process Deliverable Diagram (PDD) for the method

Activity	Sub-Activity	Description
Project initiation phase	Define project goals	The business goals and project scope are defined and stored as a set of the PROJECT GOALS.
	Analyze technological landscape	Technologies and standards of influence on the project are identified and analyzed to determine current standards and practices. The results are detailed in the TECHNOLOGICAL LANDSCAPE document.
	Elicit new requirements	A preliminary market study is conducted as the main input for requirement elicitation. If available third party process, enterprise architecture and stakeholder information is also analyzed for requirements. Resulting in a PRELIMINARY MARKET ANALYSIS.
	Write project management plan	The PROJECT GOALS, TECHNOLOGICAL LANDSCAPE and REQUIREMENT documents are combined into a PROJECT PLAN that gives an overview for the (top) management to base the go/no-go decision on. In addition it also details the planning and available resources and estimated ROI for the project.
Candidate service identification	Analyze as-is situation	An open activity consisting of three separate activities detailed in Table 8.
	Identify candidate services	An open activity consisting of three separate activities detailed in Table 9.
	Goal service comparison	In order to determine if the PROJECT GOALS are still obtainable they are compared to the identified CANDIDATE SERVICES. If enough goals can be satisfied to projects can continue. Otherwise, the project should be canceled.
	Determine service granularity	Each service is analyzed on coupling and cohesion to determine the best scope for each service to increase reuse and satisfy performance criteria. The resulting groups of services are depicted in a SERVICE INFRASTUCTURE.
	Set priority	All CANDIDATE SERVICES are ranked in the PRIORITY LIST accordingly to priority criteria, such as the Service Litmus Test (SLT) and estimated ROI.
Service specification	Map existing assets	All services not owned by the company executing the proposed method have their interface mapped to the situation of the company.
	Specify services	An open activity consisting of three separate activities detailed in Table 10.
Service construction and testing	Concept slicing	Based on the SERVICE CASE concept assignments is performed within the code. Afterwards program slicing is used to create CONCEPT SLICES that represent (executable) slices of code related to a specific concept.
	Programming	The service design is realizes as a SERVICE by implementing the SERVICE CASE, reusing the source code from the CONCEPT SLICE.
	Testing	Each SERVICE is thoroughly tested. No further details are give because of the scope of the project.
Deployment, monitoring and management	Develop provisioning strategy	This activity although recognized is considered outside the scope of this project and thus not detailed.
	Deploy services	This activity although recognized is considered outside the scope of this project and thus not detailed.
	Test deployment	This activity although recognized is considered outside the scope of this project and thus not detailed.
	Create user support material	This activity although recognized is considered outside the scope of this project and thus not detailed.
	Manage and monitor services	This activity although recognized is considered outside the scope of this project and thus not detailed.

Table 7 : Activities of the software chop shop

Figure details the open activity “analyze as-is situation” from the candidate service identification phase.

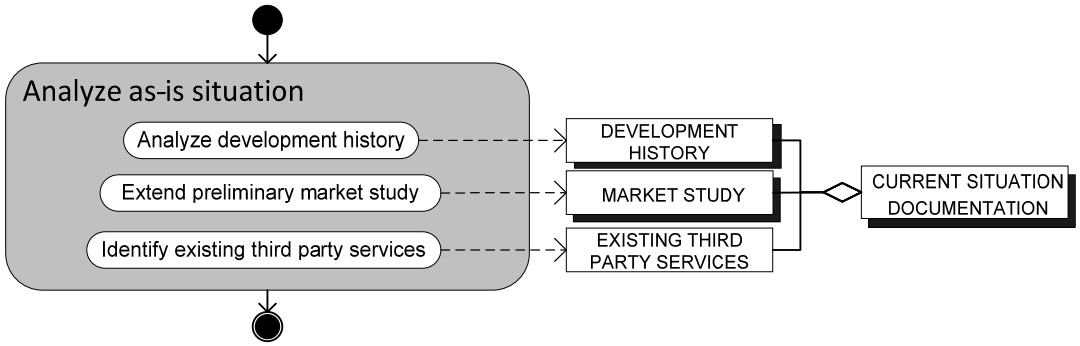

Figure 10: Analyze as-is situation

Table 8 details all the activities depicted in Figure .

Activity	Sub-Activity	Description
Analyze as-is situation	Analyze development history	Analyzing the software development history to identify and understand all the functionality it offers.
	Perform market study	An in depth market study is performed to detail the market requirements.
	Identify existing third party services	Existing third party services are identified for possibilities to be reused.

Table 8 : Sub-activities for the open activity analyze as-is situation

Figure details the open activity “identify candidate services” from the candidate service identification phase.

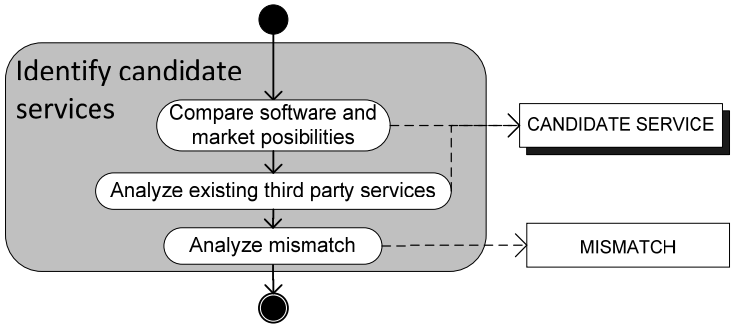

Figure 11 : Identify candidate services

Table 9 details all the activities depicted in Figure .

Activity	Sub-Activity	Description
Identify candidate services	Compare software and market possibilities	Compare the REQUIREMENTS identified in the project initiation phase with the CURRENT SITUATION to identify matches.
	Analyze existing assets	Analyze existing internal and external assets for opportunities of reuse in combination with functionality identified in the CURRENT SITUATION to form composite services.
	Analyze mismatch	The mismatch between the REQUIREMENTS and the CANDIDATE SERVICES is analyzed to determine how much new code needs to be realized for a REQUIREMENTS to meet its business goal.

Table 9 : Sub-activities for the open activity identify candidate services

Figure details the open activity “specify services” from the service specification phase.

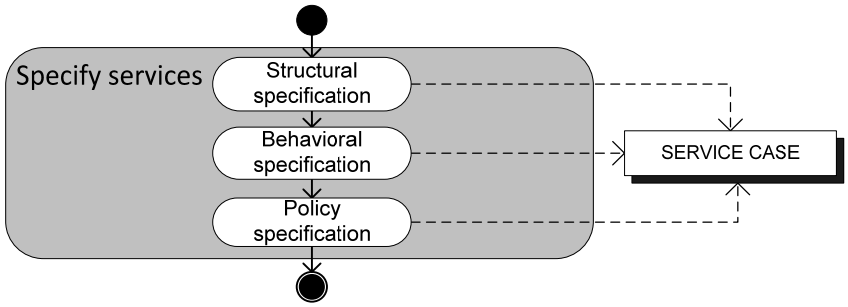

Figure 12 : Specify services

Table 10 details all the activities as depicted in Figure .

Activity	Sub-Activity	Description
Specify services (Johnston, 2005)	Structural specification	The service type, message and port type are added to the SERVICE CASE
	Behavioral specification	The service operations and semantic of the in and output messages are detailed in the SERVICE CASE document
	Policy specification	For each service the policy constrains and assertions are specified and added to the SERVICE CASE.

Table 10 : Sub-activities for the open activity specify services.

Table 11 details all the deliverables for our proposed method.

Concept	Description
PROJECT GOAL	A goal that is to be met in this specific project.
TECHNOLOGICAL LANDSCAPE	A document about the current developments in regard to the service-oriented paradigm and the best practices and standards used in the market.
REQUIREMENT	A description of functionality identified by a market study, process analysis, enterprise architecture and/or stakeholder interviews for which demand exists in the market.
PROJECT PLAN	The PROJECT PLAN contains the REQUIREMENTS, TECHNOLOGICAL LANDSCAPE and PROJECT GOALS and adds the project planning, required resources and estimated ROI for the project.
CURRENT SITUATION	A composition of the following three documents: DEVELOPMENT HISTORY, MARKET STUDY and EXISTING THIRD PARTY SERVICES.
MARKET STUDY	A study detailing the wishes and expectations identified in the market regarding services.
DEVELOPMENT HISTORY	All the historical documentation involved with the development of the software monolith.
EXISTING THIRD PARTY SERVICE	A list of existing publicly available services that fall within the scope of the project.
CANDIDATE SERVICE	A detailed functional description of a service that is a potential candidate to be developed and published.
SERVICE INFRASTRUCTUUR	Dependency diagram depicting the relations between the different CANDIDATE SERVICES.
PRIORITY LIST	A list containing all the CANDIDATE SERVICES and ordering them by set criteria.
MAPPING	A map to detail how the semantics of the in and output message of EXISTING THIRD PARTY SERVICES relates to the situation at hand, linking variables and data types.
SERVICE CASE	A detailed description of how a SERVICE should be structured, behave and what policies are in effect.
CONCEPT SLICE	An executable part of the software monolith source code that contains all the code related to a certain SERVICE CASE.
SERVICE	The realization of the SERVICE CASE. Where the service is a “self-defined, standardized and Internet-enabled software component that can be composed into distributed applications” (Papazoglou, 2003).
TEST RESULT	A document detailing all the results generated when testing the SERVICES.

Table 11 : All the deliverables for the proposed method

4. Evaluation

In order to evaluate our method, we conducted experts review by interviewing. For the expert reviews semi-structured interviews are conducted. To structure the interviews a set of topics is prepared beforehand. The topics are the method quality measures from Brinkkemper et al. (1999). The five quality measures were selected to guide the reviews because if addressed correctly we can determine the quality of a method based on the feedback of the experts. The five topics for the expert reviews are as follows:

1. **Completeness**: the method contains all the method fragments that are referred to by other fragments in the method.
2. **Consistency**: all activities, products, tools and people plus their mutual relationships in a method do not contain any contradiction and are thus mutually consistent.
3. **Efficiency**: the method can be performed at minimal cost and effort.
4. **Reliability**: the method is semantically correct and meaningful.
5. **Applicability**: the developers are able to apply the method.

Later, the change requests made by the experts were adopted to enhance our approach.

A single-case study was performed to validate the proposed method. The test was performed in a private company. The test case involved a COBOL system from which one functionality was to be exposed as service. We successfully extracted the required functionality using program slicing and concept assignment.

4.1 Result of the evaluation

First we evaluated the design by means of interviews with stakeholders for the project. Stakeholders have deemed the design created to both clearly reflect the scope for the projects as well as utilize in house practices making the methods and its results easier to use and understand. As a second evaluation the ECS was tested. The first test performed was to test if the code could be compiled with successful results. Secondly, with the compiled code the second test was performed to test if it was functioning properly a total of 240 test calculations were run and compared to the results of the tax authorities with a result of zero errors. Thirdly the ECS was evaluated by the original programmer who concluded that the code did implement the scope; however, a total of 126 lines out of the slice of 426 was dead code. Further analyzes showed that the algorithm used was correct in copying those lines because removing them required manual reprogramming of the code. Finally, an interview with the product manager was conducted to evaluate the project as a whole and the guidance the SCS provided. He was pleased with both the speed and the final results delivered. Based on our evaluation results we deem the SCS successfully applied in the project.

Due to the scope and time constraints not every activity was performed in the study. Being able to adopt the method to correspond to the project at hand is an important feature for a method that targets every product software company, where projects are bound to vary greatly. However, these adoptions also make it impossible for us to fully validate the proposed approach. We suggest a solution to these validation implications in the discussion.

5. Conclusion

In this report, we proposed and evaluated the legacy to SOA migration method, which was developed using method engineering approach from the existing SO design and development methodologies. Further, we implemented the programming slicing and concept assignment techniques to extract the so legacy code. Our approach was evaluated by the experts from the SOA fields and the software engineers and their reviews were reflected accordingly. The case study performed at a company to extract

functionality from their legacy code validated the usability of our approach. Based on our evaluation results we conclude that the method is on the correct part to achieve its goal of extracting the services from legacy systems.

However, we also identified following shortcomings of our research, which are the potential future works. First of all the deployment of services extracted for use on service markets was not addressed. Secondly, due to insufficient time and resources the SCS was not thoroughly validated. We propose that additional research is conducted to incorporate the implications brought forth by the deployment phase within the method as well as a more thorough validation has to be conducted. We also note that the extraction of code still requires considerable manual work. We reason that to increase efficiency of the method an automated service extraction tool can provide significant improvements in that regard and we aim to develop tooling in this area. We also aim for conducting a industrial case study in order to evaluate the overall method and conduct more interviews with SOA migration experts to enhance the proposed method.

References

Papazoglou, M.P., Traverso, P., Dustdar, S. and Leymann, F. (2007). *Service-Oriented Computing: State of the Art and Research Challenges*. Computer, 64-71.

Erl, T. (2004). *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice hall

Gartner[a] (2009). Gartner's 2009 Hype Cycle Special Report Evaluates Maturity of 1,650 Technologies. Retrieved February 15, 2010 from http://www.gartner.com/it/page.jsp?id=1124212

Gartner[b] (2009). Gartner Says SOA Is Evolving Beyond Its Traditional Roots, *press release.* Retrieved February 11, 2010 from http://www.gartner.com/it/page.jsp?id=927612.

Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 28, 275-280.

Brinkkemper, S., Saeki, M., & Harmsen, F. (1999). Meta-modeling based assembly techniques for situational method engineering. *Information Systems , 24* (3), 209-228.

Harman, M. Gold, N.E., Hierons, R., Binkley, D. (2002). Code Extraction Algorithms which Unify Slicing and Concept Assignment. Proceedings of the Ninth Working Conference on Reverse Engineering, 11-20.

Biggerstaff, T.J., Mitbander, B.G., Webster, D.E.(2003): **'**The Concept Assignment Problem in Program Understanding'. Proceedings of the Fifteenth International Conference on Software Engineering, ICSE'93, IEEE Computer Society Press, 482-498.

Weiser, M. (1981). Program Slicing. Proceedings of the 5th International Conference on Software Engineering. IEEE Press 1981, pp. 439-449

Ralyté, J. , Deneckère, R. and Rolland, C. “Towards a generic model for situational method engineering,” Proceedings of Advanced information systems engineering: 15th international conference, CAiSE 2003, LNCS, 2003, 95-111

Kumar, K., & Welke, R. (1992). *Methodology engineering: A proposal for situation specific methodology.* Washington, DC, USA: John Wiley and Sons.

Weerd van de, I., Brinkkemper, S. (2009). Meta-Modeling for Situational Analysis and Design Methods. Handbook of Research on Modern Systems Analysis and Design Technologies and Applications, 35-54.

Weerd van de, I., Brinkkemper, S., Souer J., and Versendaal, J. . “A situational implementation method for web-based content management system-applications: method engineering and validation in practice,” software process improvement and practice, vol 11, 2006, pp. 521-538.

Weerd van de, I., Weerd de, S., Brinkkemper, S. (2007). in IFIP International Federation for Information Processing, Volume 244, Situational Method Engineering: Fundamentals and Experiences, eds. Ralyt6, J., Brinkkemper, S., Henderson-Sellers B., (Boston Springer), 313-327.

Sommerville, I. (2007). Software Engineering, Eighth edition. Addison-wesley

Papazoglou, M.P., Traverso, P., Dustdar, S. and Leymann, F. (2007). *Service-Oriented Computing: State of the Art and Research Challenges*. Computer, 64-71.

Kumar, K., & Welke, R. (1992). *Methodology engineering: A proposal for situation specific methodology.* Washington, DC, USA: John Wiley and Sons.

Lee, S.P., Chan, L.P. & Lee, E.W. (2006). Web services implementation methodology for SOA application. *Proceedings of the International Conference on Industrial Informatics*, 335-340.

Gold, N.E., Harman, M., Binkley, D., and Hierons R.M., (2005). Unifying program slicing and concept assignment for higher-level executable source code extraction: Research articles. Softw. Pract. Exper., 35(10), 977–1006.