A Method Engineering Based Legacy to SOA Migration Method

Ravi Khadka, Gijs Reijnders, Amir Saeidi, Slinger Jansen and Jurriaan Hage
Department of Information and Computing Sciences
Utrecht University
Email: {ravi, gwreijnd, amir, slinger, jur}@cs.uu.nl

***Abstract*—Legacy systems are vitally important for the continuation of business in an enterprise as they support complex core business processes. However, legacy systems have several well-known disadvantages such as being inflexible and hard to maintain, so momentum is growing to evolve those systems into new technology environments. Recently, service-oriented architecture has emerged as a promising architectural style that enables existing legacy systems to expose their functionality as services, without making significant changes to the legacy systems themselves. A significant number of the legacy to service migration approaches address the technical perspective (i.e., supporting technology) to expose the legacy code as services. The other approaches focus on determining the feasibility of the migration that includes economical and technical feasibility, based on the characteristics of existing legacy system and the requirements of the target SOA system. In this paper, a legacy to SOA migration method that does not single out the migration feasibility and technical perspectives, but combines these two perspectives of migration, is proposed. Method engineering is used to develop the migration method by reusing method fragments from existing service-oriented development methods. Then, concept slicing is used to develop the service by extracting the relevant parts of the legacy code. The method is evaluated and enhanced by interviewing experts and further validated with two case studies. The method is found to be appropriate and effective in extracting services from legacy code with the aim of reusing these services in new configurations.**

I. Introduction

A large number of enterprises depend on business-critical systems for consolidating business information that have been developed over the last three decades or more using 3GL programming languages such as COBOL, RPG, C, C++ [1]. These systems are called *legacy systems*. It is estimated that more than 80% of the world's business runs on COBOL, and 50-70% of the total IT costs are devoted in the maintenance of these systems [2]. Legacy systems are now a roadblock for the evolution of the IT infrastructures in an enterprise due to their well-known disadvantages such as being inflexible and hard to maintain [3].

However, enterprises still rely on these legacy systems as they usually implement complex core business processes, and the high risk associated with necessary changes [4]. Since legacy systems are vitally important for the continuation of business in the enterprises, momentum is growing to evolve those systems into new technology environments [3]. Recently, Service-Oriented Architecture (SOA) has emerged as a promising architectural style that enables existing legacy systems to expose their functionality as services, without making significant changes to the legacy systems themselves [5]. The migration from legacy systems to SOA can be beneficial from an economical and technical perspective. From the economical perspective, enterprises are constantly challenged by an accelerating pace of changes, such as intra-organizational changes, changes in market demands and opportunities, and, consequently, changes in enterprise collaboration. The migration of legacy to SOA enables legacy systems adapt to such changes [6] and aims at reducing the maintenance costs [7]. From the technical perspective, seamless enterprise collaboration through service composition and heterogeneous application integration within/outside the enterprises [8] are claimed.

Several approaches have been reported in literature to migrate legacy systems to SOA and web service technology. A significant number of such approaches focus on the development of the supporting technology to address the technical migration perspective (i.e., implementation techniques to expose the legacy code as service) [2], [9], [10], [11], [12], [13], [14], [15], [16]. Other approaches focus on developing a migration strategy to determine the migration feasibility. Such feasibility is determined based on the characteristics of the existing legacy systems for their potential to be exposed as services and the requirements of the target SOA system [5], [17], [18], [19]. However, a legacy migration method requires the consolidation of both the aforementioned perspectives (i.e., migration feasibility and supporting technology) [20], which, as per our knowledge, is still missing.

In this paper, a legacy to SOA migration method, hereafter called *ServiciFi method*, is developed that combines the migration feasibility and development of supporting technology of the legacy to SOA migration. The serviciFi method is developed by assembling the fragments of existing service-oriented development methods using method engineering [21]. For the development of the supporting technology, concept slicing [22] is used to facilitate the extraction of the services from the legacy code.

The rest of the paper proceeds as follows: Section II introduces the research design that has been followed to develop the serviciFi method. Section III explains the serviciFi method followed by the evaluation using experts review and two case studies in Section IV. Finally, Section V concludes the paper with an outlook to future research directions.

II. Research Design

The serviciFi method is designed following the design science in information systems research, suggested by Henver et al. [23]. The serviciFi method is first designed and evaluated with experts review followed by two case studies. Method engineering is used to design the serviciFi method by assembling the activities of existing service-oriented development methods. In the following subsections, the method engineering approach and it's related concepts used while designing the servicFi method are detailed. Further, concept slicing is also explained as a supporting technology for the migration.

Method engineering for information system development is an approach to construct advanced development methods by reusing parts of existing methods [21]. In the work of Brinkkemper [21], a *"method"* is defined as an approach to perform a system development method consisting of directions and rules, structured in a systematic way in the development activities with corresponding products. The development activities and corresponding products are called *"method fragments"*. The method engineering approach is used to develop the serviciFi method due to the fact that reusing the existing and proven method fragments from existing service-oriented development methods saves time and reduces the adoption problem (i.e., easy to adapt to the existing standards/methods). A specific strategy of method engineering is assembly-based situational method engineering [24] [25], which includes a step to create a method base in case, if it does not exist. A method base is a repository where the method fragments can be stored and retrieved [21]. To create the serviciFi method, there is no existing method base from which the method fragments can be reused. So, the assembly-based situational method engineering is used to create the method base for the serviciFi method. The assembly-based situational method engineering has the following steps that are followed to create the serviciFi method.

A. Analyze Situation and Identify Needs

The need for the serviciFi method has been described in Section I. The serviciFi method should combine the migration feasibility and development of supporting technology to extract legacy codess and expose them as services. Also, the extracted services should adhere to the current standards of SOA to facilitate the heterogenous application integration across/within the enterprises.

B. Select Candidate Methods

In this step, the service-oriented development methods containing relevant method fragments are selected. Based on higher number of citations (popularity), availability of documentation and completeness of the method, the following service-oriented development methods are selected: Service-Oriented Design and Development Methodology (SODDM) [26], Web Service Implementation Methodology (WSIM) [27], and Service-Oriented Modeling and Architecture (SOMA) [28]. Hereafter, these methods are called *"candidate methods"*.

C. Analyze Candidate Methods and Store Relevant Method Fragments in Method Base

In this step, the method base is filled with the relevant method fragments derived from the three candidate methods. For each candidate method, a Process Deliverable Diagram (PDD) is created, by applying the metamodeling technique as described by Weerd et al. [25]. The PDD of the candidate methods depicts every activity and deliverable of each phase. The PDD of each candidate method is not presented in this paper due to the space limitation. To give an impression of how a PDD is represented, the PDD of the serviciFi method (see Figure 4) is presented as an example. A PDD consists of two integrated diagrams: the process view on the left-hand side of the diagram is based on a UML activity diagram, and the deliverable view on the right-hand side of the diagram is based on a UML class diagram. The process/deliverable view diagram constitutes different types of activities/concepts. These activities/concepts are depicted in Figure 1 and explained as follows:

- *Standard activity/concept:* A standard activity/concept contains no further (sub)activities/concepts.
- *Open activity/concept:* An open activity/concept contains further (sub)activities/concepts.
- *Closed activity/concept:* An activity/concept whose (sub)activites/concepts are not elaborated since it is not known or not relevant in the specific context.

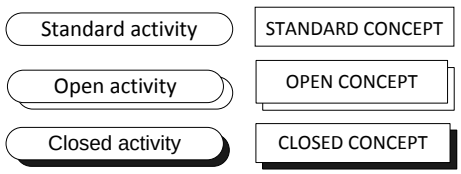

Fig. 1. Activity and Concept types [25]

D. Select Useful Method Fragments and Assemble a New Method

In this step, useful method fragments from the method base are selected and assembled to form the serviciFi method. This step is divided into three sub-activities.

First, the general phases of the serviciFi method need to be identified, which is done by comparing and analyzing the phases of the three candidate methods. The phase comparison is depicted in Table I. As a result of this comparison, five phases of the serviciFi method are identified, resembling the phases of the three candidate methods. To make these phases reflect their intent, they are renamed as follows:

- Project initiation
- Candidate service identification
- Service specification
- Service construction and testing
- Deployment, monitoring and management

Second, a method comparison among the method fragments, stored in the method base, is done using the method

TABLE I
PHASE COMPARISON

	WSIM	SODDM	SOMA
1	Requirements	Planning	Business Modeling & Transformation
			Solution Management
2	Analysis	Analysis & Design	Identification
3	Design		Specifications
4	Coding	Construction & Testing	Realization
	Testing		Implementation
5	Deployment	Provisioning	Deployment, Monitoring & Management
		Deployment	
		Execution & Monitoring	

comparison matrix [29] to create a so-called "*super-method*". The super-method contains the activities that are considered reusable for the serviciFi method. Creating the super-method involves the step-by-step comparison of all the activities and deliverables among the three candidate methods of the method base. Each activity, in the same phase, was evaluated based on their description to find if the activity was:

- Out of scope such that the activity is discarded.
- Equal to another activity such that only one of the activities is included in the super-method.
- Fully contained within another activity such that scoping decision is made.
- Not relevant in the current phase such that the activity is discarded.

TABLE II
EXCERPT OF THE PROJECT INITIATION PHASE OF THE METHOD COMPARISON MATRIX

WSIM	SODDM	SOMA
Determining the need of web service	**Analyzing the business needs**	-
-	**Review current technological landscape**	-
Elicit web service requirement	**Conceptualize the new requirements**	-
Manage Web service requirements		
Model usage scenario		
-	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*

Table II depicts an excerpt of the method comparison matrix of the project initiation phase. The activities within **bold** are the assembled method fragments to form the super-method. The activities within *italics* are out of scope. The activities within * * are not relevant in the current phase. The activities that are similar or if combined are similar to higher-grained activities, are presented in same row of the method comparison matrix. The corresponding excerpt of the super-method representing the excerpt of the method comparison matrix (Table II) is depicted in Figure 2. Among *"Analyzing the business need"* of SODDM and *"Determine the need for web service"* of WSIM, which are functionally similar activities, the earlier one is chosen as the naming is more meaningful. The *"Conceptualize the new requirements"* activity of SODDM is chosen since it represents the higher-grained activity as compared to the three similar activities of WSIM (i.e., *"Elicit Web service requirements"*, *"Manage web service requirements"*, *"Model usage scenario"*). Finally, the *"Initiate project management"* activity of SOMA is chosen as compared to *"Manage project deliverables and resources"* activity of SODDM as naming of the activity is more meaningful. By comparing the activities and the deliverables of each phases, the super-method of the serviciFi method is created.

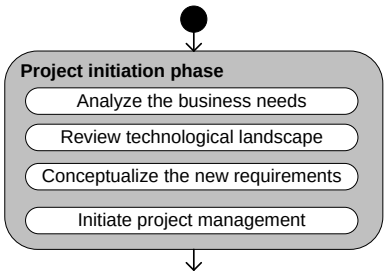

Fig. 2. Excerpt of the project initiation phase of the super-method

Third, the super-method by now consists of method fragments assembled from the three candidate methods. But, to adapt the super-method for migration, several other activities and deliverables such as cost-benefit analysis [30], identifying third party services that has similar functionality to the to-be-extracted services, priority techniques to determine the priority of extraction of identified services are added. Also, the assembled method fragments are renamed to adapt with the migration context. For instance, the *"Analyze the business needs"* of the super-method (see Figure 2) is renamed to *"Define project goals"*, *"Review technological landscapes"* to *"Analyze technological landscape"*, and so on. Based on these modifications in the super-method, the serviciFi method is finalized and is explained in Section III.

Figure 2 and Table II are only the excerpts of the super-method and the method comparison matrix, respectively. The details of the super-method and the method comparison matrix are not included in this paper due to the reasons of brevity and space limitation. Their details have been reported in the work of Reijnders et al. [31].

The serviciFi method aims at reusing the existing legacy code to extract services. The *"Service construction and testing"* phase of the serviciFi method is distinct with the corresponding *"Implementation"* phase of the three candidate methods. The *"Service construction and testing"* phase should include activities to facilitate the legacy source code extraction, whereas, the corresponding *"Implementation"* phase of the three candidate methods aims at creating new services from

scratch. Concept slicing is used as an implementation technique (i.e., supporting technology) to extract the legacy codes and expose them as services in the *"Service construction and testing"* phase.

Concept slicing [22] combines two techniques from software (re)engineering and maintenance domain: *program slicing* and *concept assignment* to generate an "*Executable Concept Slice*" (ECS). Program slicing [32] is a well known code analysis technique that is used to identify and abstract the smallest possible subset of a program that can perform an expected functionality. Concept assignment [33] is a technique that assigns individual human-oriented concepts to portions of source code. Both techniques have been used as source code extraction techniques that take a criterion and program source code as input and yield parts of the program as output. However, the extraction criterion of program slicing is expressed at a very low level to construct a slicing criterion such as using program variables, which makes slicing difficult to apply. In concept assignment, the extraction criterion is expressed at the domain level, making it more practical to apply, but unlike program slicing, the extracted code is not executable as a separate (sub)program. To achieve the combined advantages, while overcoming the individual weaknesses, Gold et al. [22] combined these two techniques as concept slicing and successfully extracted executable source code from legacy code.

III. The ServiciFi Method

The serviciFi method is depicted in Figure 4 as a PDD. In the following subsections, each phase and its constituent activities are detailed.

A. Project Initiation

The *"Project initiation"* phase performs the assessment of the viability of the legacy to SOA migration by analyzing the technical and economical feasibility. The phase starts with the "*Define project goals*" activity that identifies what functionalities of the legacy system need to be exposed as services. The second activity, "*Analyze technological landscape*", analyzes the technical aspects of the existing legacy systems such as the programming language used to build the legacy system, availability of the documentation or resources and also the requirements of the target SOA system. In the *"Analyze technological landscape"* activity, the *"portfolio analysis"* [30] of the identified functionalities that are to be exposed as services, is performed. The portfolio analysis assesses both the technical information and business values of the identified functionalities. The technical information includes the overall system functioning of the legacy system and the functionalities present in the legacy system from which the identified functionalities are to be migrated. The business value of the identified functionalities includes the preliminary benefits that is achievable by exposing the identified functionalities as services. The *"Portfolio analysis"* gives the high level overview of the legacy system, its functionalities and the economic benefits. The portfolio analysis is performed by interviewing the developers, if there are any, and/or consulting to the available documentation, and/or the current users of the legacy systems. The business value indicates the viable business investment and the preliminary return of investment, which is calculated in terms of maintenance cost (if possible). The business value of the migration is analyzed by interviewing the business managers and by investigating the market needs. The project goals and analysis of technological landscape provide new requirements for the project, which are documented as requirements in "*Elicit new requirements*" activity. Finally, the *"Project management plan"* is stated. The outcomes of the project initiation phase are project management plan and the (dis)approval of the migration project.

B. Candidate Service Identification

This phase focuses on identifying candidate services to satisfy the requirements detailed in the "*project initiation*" phase. The first activity, "*Analyze as-is situation*", is an open-activity consisting of three sub-activities as shown in Figure 3. The

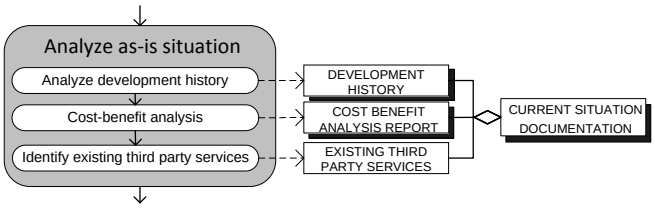

Fig. 3. Analyze as-is situation

"*Analyze development history*" sub-activity investigates the artifacts of the legacy system, such as requirements documents, UML diagrams, data diagrams, source code, class diagrams, system dependence graphs and even comments in the code in detail as compared to the "*Analyze technical landscape*" activity of the "*Project initiation*" phase. All these information provide better understanding about the development history as well as functionalities contained within the legacy system. The *"Cost-benefit analysis"*, as suggested by Sneed [30], is performed to estimate the cost of the migration. The cost-benefit analysis is carried out to compare the migration costs with expected benefits. Typically, in this step a comparison between the benefits of migration, redeveloping and doing nothing is performed. The *"Cost-benefit analysis"* sub-activity determines if the migration project is economically viable. The "*Identify existing third party services*" activity investigates if the services that the migration method aims to extract is already available in the market. Availability of such existing services can either provide opportunities to create composite functionalities or already decide to reuse those existing services rather than extracting from the legacy system, if possible. For instance, if one of the functionalities to be extracted as a service is the "*Validation of credit card*" functionality, then reusing the available free web services such as *ValidateCreditNumber*[1] can be economical.

[1]http://www.webservicex.net/ws/WSDetails.aspx?WSID=14&CATID=2, Last accessed on: 30 March 2011

Once the "*Analyze as-is situation*" is documented, the next activity is the "*Identify candidate services*". As for now, this activity is carried out manually based on the functionalities identified in the legacy code against the requirements identified in the "*project initiation*" phase. It is possible that there could be mismatches between the identified functionalities in the legacy code and the requirements of the services to be exposed. Such mismatches are documented in "*Goal comparison*" activity, which can be used to determine if the identified candidate service can satisfy the business requirements. Depending on the goal comparison, the project might be canceled if the identified candidate services do not fulfil the requirements. For each of the identified candidate services, the granularity has to be determined and is performed in "*Determine service granularity*" activity. The service granularity determines if the identified candidate services need to be extracted as an atomic service or as a composite service, representing the composition of the atomic services. Both the granularities have their own advantages such as extracting the atomic services allows composing the new functionalities in future and hence, increases reusability of the extracted services. Whereas, extracting the composite services allows maintaining few services after deploying in service infrastructure and hence, reduces the maintenance cost. The final activity is "*Set priority of services*" in which the identified candidate services are prioritized based on the requirements and business needs. Priority technique MoSCoW [34] is used to determine and create the priority list of the identified candidate services. Based on the priority list the development iterations are planned. The first iteration is started with the highest priority functionality. After every iteration, the priority list is re-evaluated. The iteration fosters the incremental development of the migration.

C. Service Specification

The "*Service specification*" phase further details the identified candidate services for the current iteration as well as redesigning any existing third party services, if any third party services are identified from the *"Identify existing third party services"* sub-activity of the *"Analyze as-is situation"* activity. Such third party services involved in the current iteration need to be mapped against the existing data types and variable names of the legacy code. This activity provides an overview of the input and output messages required by these candidate services. Once the mapping between the existing services and candidate services is done, the candidate services are detailed on a technical level. Based on the work of Johnston [35], the following three steps are performed to complete the service specification in the technical level: (i) *Structural specification* that determines the necessary operations that represent the functionality of the candidate services and messages that are communicated via the operations by observing the legacy code, (ii) *Behavioral specification* that defines service interfaces that a client will use along with the necessary input, output messages and operations, and (iii) *Policy specification* that denotes policy assertions and constraints for each specific service.

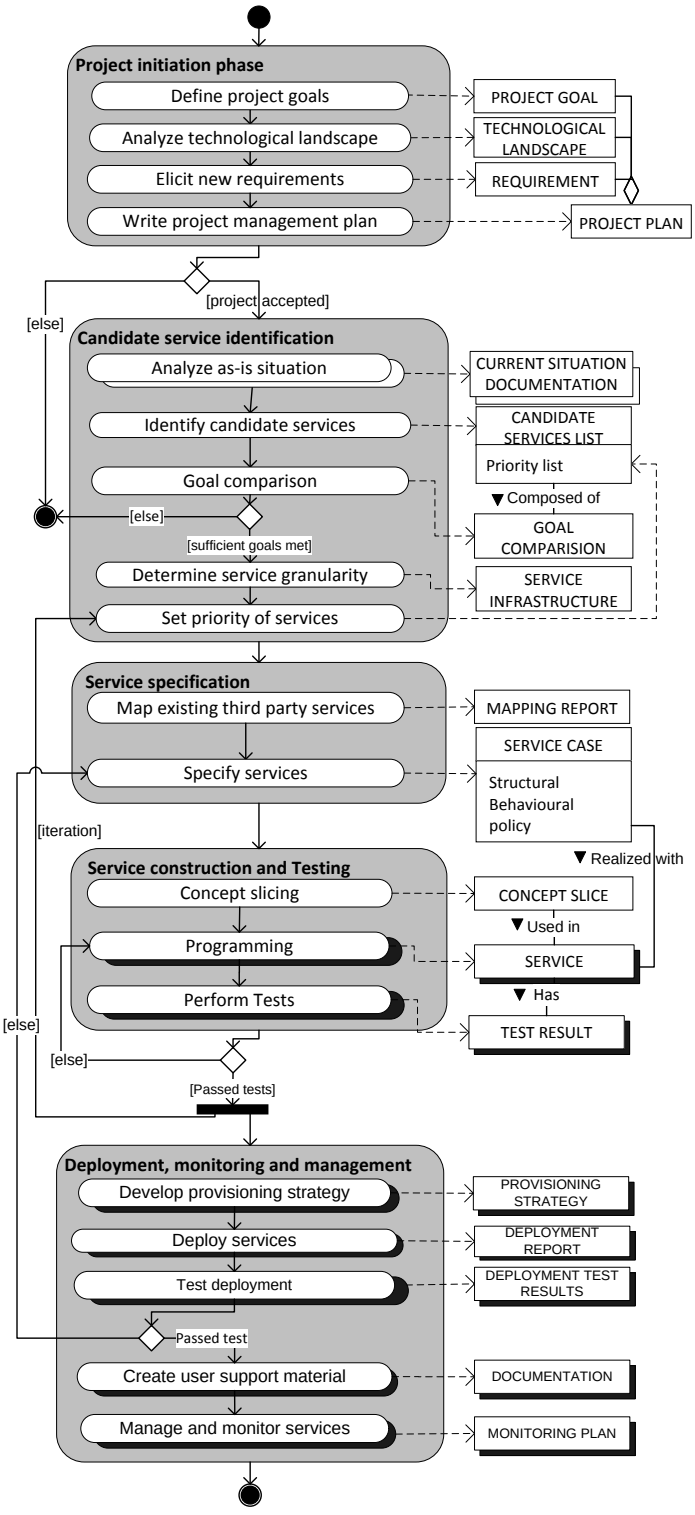

Fig. 4. PDD of the serviciFi method

D. Service Construction and Testing

The "*Service construction and testing*" phase is concerned with the extraction of the source code from the legacy system. The "*Concept slicing*" technique is used to extract the legacy code as an executable concept slice (ECS). This technique

is successfully used by Harman et al. [36] to extract the executable (sub)programs from the COBOL-based financial application and by Gold et al. [22] from C-based accounting software. The concept slicing activity is followed by "*Programming*" activity to facilitate the execution of the ECS, if required. For instance, the ECS might need to be deployed in different environments so the ECS might need to be refactored and/or the system calls present in the ECS might need to be replaced or re-programmed. Such modifications are done in the *"Programming"* activity. Finally, each ECS is tested to determine if it is functioning in accordance with the legacy code. In "*Perform tests*" activity, following tests are performed: unit tests, functional tests, and system tests depending on how the ECS is developed. In case of automatic extraction of ECS using concept slicing tools, only functional test is carried out. In case, if the ECS is developed using manual extraction, then all the specified tests need to be executed. When the ECS passes these tests, it is safe to assume the extracted code does in fact satisfy the required functionality and is ready to be deployed as a service in a service infrastructure. At the same time, the iteration for the next candidate service in the priority list is initiated.

E. Deployment, Monitoring and Management

The "*Deployment, monitoring and management*" phase concerns with the deployment of the ECS as a service. The initial activity "*Develop provisioning strategy*" facilitates the usage of the services from both technical and business aspect by a consumer [37]. Service provisioning typically includes activities such as publishing services into a catalog, versioning of services, metering & billing of the usage of the services [38]. The services are then deployed in the service infrastructures and tested. Based on the outcome of the "*Test deployment*" activity, either the extraction of the service has to be performed again if the specified functionality is not met or the service is ready for usage. Subsequently, the user support materials such as documentation are created. Furthermore, to ensure the proper functioning of the deployed services, the support for management and monitoring is provided in the "*Manage and monitor services*" activity.

IV. Evaluation

In order to evaluate the serviciFi method, initially eight semi-structured interviews have been conducted with experts. Later, the serviciFi method was applied to two case studies to evaluate in practice. In the following subsections, each evaluation method is detailed.

A. Experts Review

The method is evaluated with several experts from industry and academia by conducting semi-structured interviews. This method was chosen because it includes a mixture of open-ended and specific questions, designed to elicit not only the information foreseen, but also unexpected types of information [39]. Table III depicts the details of the experts. The names are kept anonymous and the experience (in years) in the related field was explicitly asked before the interview was conducted. The variation in the expertise of the experts enabled the assessment of the serviciFi method from different perspectives of software practitioners.

TABLE III
Details of the experts

Name	Expertise	Experience	Sector
A	Software product manager	5	Industry
B	Application manager	5	Industry
C	Software engineering researcher	5	Academia
D	Migration from legacy to SOA/cloud	5	Industry
E	Migration from legacy to SOA	3	Industry
F	Software migration	3	Industry
G	SOA researcher	7	Academia
H	Requirement engineering researcher	6	Academia

Prior to the interview, the experts were provided the PDD of the serviciFi method and the corresponding documentation. The method was first explained to each expert before conducting the actual interview. Later, the experts were asked to give feedbacks or remarks on the method. The interviews were conducted in English and each of them took between 80 to 120 minutes. Every interview was recorded and then transcribed. The interviews were conducted to evaluate the proposed method on the basis of five quality measures, suggested by Brinkkemper et al. [24] while designing methods using method engineering. The five quality measures are described as following:

- *Completeness:* The serviciFi method is the assembly of the method fragments of existing service-oriented development methods, so the experts were asked if the serviciFi method captures the complete activities/phases of a migration scenario.
- *Applicability:* The experts were asked if any activities/phases in the serviciFi method were in contradiction with each other in a way that hinders the applicability in any migration project.
- *Efficiency:* The experts were asked if the serviciFi method is efficient and does not contain any repetition of the phases/activities that would increase cost and effort.
- *Consistency:* The experts were asked if the phases/activities were consistent (i.e., semantically correct and meaningful) with each other.
- *Reliability:* The experts were asked if the serviciFi method is reliable to apply in any real world legacy to SOA migration project.

B. Discussion of the Experts Review

After all experts were interviewed, Constant Comparative Analysis (CCA) [40], [41] was used to analyze the results. The CCA method is used to create knowledge from the data source by avoiding subjective interpretation [42] (i.e., interpretation of the data in accordance with the research objectives). The outcome of the analysis was categorized into two: (i) *minor change requests* that included changes such as renaming the activity names, renaming the deliverables, and (ii) *major change requests* that included changes such as adding/deleting

activities, adding iterations among the phases, adding/deleting branchings. The PDD depicted in Figure 4 already includes the adjustments made after CCA.

In response to the *completeness* and *consistency* quality measures, around 75% (6 out of 8 experts) agreed that the phases of the serviciFi method are complete and consistent. However, the addition of *"Cost-benefit analysis"* sub-activity in *"As-is situation"* activity of *"Candidate service identification"* phase was emphasized. Also, the analysis resulted in the emphasis on using the iterative development method. One of the experts described the need for the iterative development method as "*the migration process is time consuming so there needs to be iterative development cycles to include the changes occurred within the project life time.*"

In response to the *applicability* quality measure, the analysis indicated that almost all experts agree on the applicability of the method as the serviciFi method contains all the typical phases and activities required for a migration project.

In response to the *efficiency* and *reliability* quality measures, around half of the experts (4 out of 8 experts) were not confident about the efficiency and reliability of the method. The analysis showed that efficiency and reliability might be influenced by the tools and techniques used in the method along with the various factors of the legacy systems such as availability of the documentation, support from the current users to understand the system, quality of the source code, redundancies in the source code, and understandability and maintainability of the code. One of the experts explained this situation as "*determining the efficiency is contextual because the candidate service identification phase and service construction phase highly depend on the status of the legacy system such as how well the legacy code is written and maintained so far.*"

The analysis also emphasized the need of migration for the current users. The migration of such users is indeed an important aspect, which should not be underestimated [43], [44]. It is recommended to carry our proper user migration planning by conducting training programs [45], however, user migration is out of scope of this research.

Overall the analysis of the expert reviews revealed that most of the experts agreed on *completeness*, *consistency*, and *applicability* quality measures but were not confident on the *efficiency* and *reliability*. In order to evaluate the *efficiency* and *reliability* quality measures, two case studies were conducted in which the serviciFi method was used to extract services from legacy system. The details of those case studies are described in IV-C and IV-D.

C. COBOL Case Study

The initial case study was conducted in a Dutch product company that uses a COBOL-based legacy system for wage and personnel administration. For the reason of confidentiality, information such as company name, product name, available functionalities of the legacy systems are kept anonymous. The company wanted to reuse one of the functionalities present in the legacy system as a web service. For the migration of the functionality, the serviciFi method was used. The project goal was to extract the specific functionality and expose it as a web service. The portfolio analysis of the *"Analyze technological landscape"* revealed that the system was developed about 10 years ago in COBOL. Various other functionalities of the whole legacy system were also dependent on this particular functionality. The aim of this migration was to attract more customers to use their software as a service. Also, the company aimed at reducing the maintenance cost by exposing the functionality as a service. The *"Elicit new requirements"* activity of the *"Project initiation"* phase emphasized on the use of web service standards to be followed while migrating. The project plan was documented and the *"Candidate service identification"* phase was started. In the *"Candidate service identification"* phase, the *"As-is situation"* sub-activity indicated that the legacy code consists of 1588 lines of code with a lot of exceptions. The availability of the original programmer of the source code helped us to understand the specific functionality. The *"Cost-benefit analysis"* was done by the company which showed that the migration is economically viable and profitable. The *"Identifying existing third party services"* was not relevant in this case because the company wanted to have it's own service and also the *"Goal comparison"* activity was not relevant. The service was to be extracted as an atomic service such that it can be reused in future for any other service composition. Due to the extraction of a single service, no priority list was created. In the *"Service specification"* phase, the initial activity *"Map existing third party services"* was not relevant for the current functionality. The structural, behavioral and policy of the service were specified by consulting the original programmer and the product manager of the system.

The *"Concept slicing"* was performed manually due to the unavailability of a slicing tool for COBOL program. Using the concept assignment, 14 out of 132 variables were identified that were related to the concept (i.e., the functionality to be extracted). Based on those 14 variables, program slicing was applied that extracted 426 lines of code from the 1588 lines of code. The extracted code was successfully compiled and in total 240 test cases were run to validate the functionality of the extracted code. All the results of the test cases were compared with the results of the test cases that were run on the original code and 100% successful result was obtained.

D. C++ Case Study

In a second case study, a C++ program called SrnaCalc[2] has been used to demonstrate and evaluate the proposed approach. It is an open-source calculator issued under GNU General Public License ver.3.0 (GPLv3). SrnaCalc is a simple command-line calculator with essential mathematic functions, memory and scripting capability. The *"Analyze technological landscape"* activity indicated that the following functionalities were present: *operators* to display the operators used, *eval* to evaluate the expression, *getPrecision* and *setPrecision* to

[2]http://sourceforge.net/projects/srnacalc/

manage precision, *memory* to list the contents of memory, *add*,*set*, *isset*, *get*, *remove* and *read* to add, change, find, append, remove, and read a variable of the memory, respectively.

The goal was to extract *eval* functionality that calculates expressions and to expose it as a service. The *"Elicit new requirement"* and the *"Write project management plan"* activities were not relevant in this case study. The *"Analysis as-is situation"* activity resulted in program metrics as depicted in Table IV and program dependency graph as depicted in Figure 5. The program dependency graph was generated using Understand tool [46] and was manually edited to enrich it visually by filling with colors and differentiating the edges. The *SC\eval.cpp*, represented with the dark background, is the main code which implements the *eval* functionality. The dotted edges represent the non-relevant dependencies of the main code with other program files after the concept slicing. The comments in the program were written in Czech so Google translator was used to translate the comments for a better understanding of the program.

TABLE IV
PROGRAM MATRIX

Attributes	Count
#Classes	12
#Class & header Files	21
#Methods	84
#Library Units	114
#Lines of Code	2196
#Blank Lines	236
#Comment Lines	223
#Ratio Comment/Code	0.13

The *"Goal comparison"*, *"Determine service granularity"*, *"Set priority of services"*, and *"Map existing third party"* activities were not relevant in this case study. By manually investigating the code, the service specifications were documented by identifying the function that evaluates the expression and the required parameters.

To facilitate the *"Service construction and testing"* phase, CodeSurfer [47], a C/C++ slicing tool, has been used to extract the legacy code. The extracted code has been initially tested in Visual Studio 2010 Ultimate edition and the shared library file, *evaluate.dll*, is created. To deploy the service and make it available to the clients, a service descriptor file, *service.xml*, is created. Listing 1 depicts the *evaluate* interface of the *evaluate* web service. Finally, the *service.xml* and the *evaluate.dll* is deployed in the WSO2/C++ web service framework [48]. The service is successfully tested by developing a client application.

Listing 1. Service description file (service.xml)

```
<service name="evaluate">
  <parameter name="ServiceClass"
        locked="xsd:false">evaluate</parameter>
  <description>
    The Calculator evaluation function
    presented as Evaluate service
  </description>
  <operation name="evaluate">
        <messageReceiver class="wsf_cpp_msg_recv"/>
  </operation>
</service>
```

E. Discussion of the Case Studies

The serviciFi method has been successfully assessed and proven to be feasible and practical in the two case studies. Moreover, the result of these case studies also complemented some of the quality measures and the findings of the expert reviews. The successful extraction of services in both the case studies supported the applicability, efficiency and reliability of the serviciFi method. However, the completeness and consistency quality measures are yet to be determined as some of the activities in different phases of the serviciFi method were skipped. Also, the extraction of services as an iterative process was not applicable as both the case studies involved only a single service extraction. However, the experience of service extraction in the case studies complemented the experts review regarding the efficiency and reliability quality measures. As per the analysis of expert reviews, the efficiency and reliability of the method are influenced by the availability of tools and techniques along with the characteristics of the legacy system, which was reflected while conducting the case studies. Considerable time has been invested to conduct the manual slicing in the COBOL case study while it was easy in the C++ case study due to the availability of slicing tool. Also, the availability of the original programmer and his support enhanced the service extraction process in COBOL case study.

F. Threats to Validity

The following three are the main threats to the validity of the results of a case study [49]: (i) reliability validity, (ii) internal validity, and (iii) external validity. The reliability validity concerns with the repeatability (i.e., if the same case study is performed by another researcher later following the same procedures as conducted by the earlier researcher, the result of the latter should also arrive at same findings and conclusions). With respect to reliability, the repetition of the COBOL-based industrial case study was not possible. But, the procedures followed in the initial case study was well documented such that the same procedures were followed in the C++ case study, which was conducted by a different researcher. The potential threat to the internal validity was the direct involvement of the authors in both case studies, which can introduce bias in the result. To minimize the threat to internal validity, the two case studies were conducted by two different researchers and the result was analyzed by a third researcher. With respect to the external validity, more case studies will have to be performed to extend the results of the current case studies. The different execution context of the two case studies (i.e., one being industrial and the other being experimental) has supplemented the generalizability (i.e., external validity).

V. CONCLUSION

In this paper, the serviciFi method for legacy to SOA migration has been presented that was developed using method engineering and concept slicing. Unlike other approaches

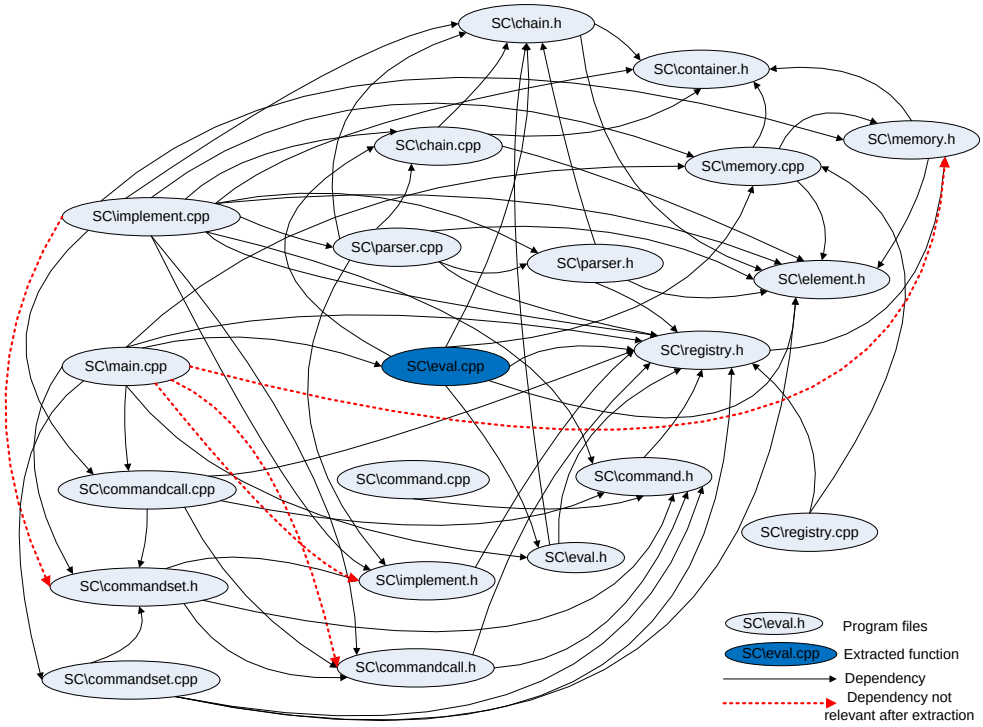

Fig. 5. Dependency Graph

reported in the literature, the serviciFi method successfully combined the migration feasibility with supporting technology to expose the legacy code as services. The serviciFi method has been evaluated and enhanced with expert reviews and has been proven to be feasible to migrate legacy systems to SOA with two case studies. The core of the method is method engineering that reuses the method fragments from existing service-oriented development methods and enhanced by the concept slicing method to develop services by extracting the legacy code.

As a part of future work, the serviciFi method needs to be evaluated with realistic industrial case studies. Also, the serviciFi method can be improved in several ways. The first one concerns the possibility of identifying the service-rich areas in legacy code and automating the identification of the candidate services. Currently, the candidate service identification activity is carried out manually, which can be time consuming and difficult in realistic legacy to SOA migration projects. Pattern identification techniques such as architectural and structural patterns [50], process mining techniques [4], and feature location techniques [51] are possible directions for future work to automate the candidate service identification phase. The next possible direction of the future work concerns the enhancement of service construction phase of the serviciFi method by implementing service extraction activity using *code query technologies* [52], [53]. Code query technologies support the so-called extract-abstract-present paradigm. Extraction maps source code to relations, the query language then provides the means to query these relations, in order to build new relations and, finally, the results can be presented.

ACKNOWLEDGMENT

This research has been sponsored by the Dutch Joint Academic and Commercial Quality Research and Development (JACQUARD) program on Software Engineering Research via contract 638.001.213 ServiciFi[3]: Service Extraction from Decomposed Software Monoliths in the Financial Domain.

REFERENCES

[1] K. Bennett, "Legacy systems: Coping with success," *IEEE Software*, vol. 12, no. 1, pp. 19–23, 1995.

[2] G. Koutsoukos, L. Andrade, J. Gouveia, and M. El-Ramly, "Service Extraction," Sensoria Project, Tech. Rep. 016004, 2006, Available from:http://www.pst.ifi.lmu.de/projekte/Sensoria/del_12/D6.2.a.pdf.

[3] J. Bisbal, D. Lawless, B. Wu, and J. Grimson, "Legacy information systems: Issues and directions," *Software, IEEE*, vol. 16, no. 5, pp. 103–111, 1999.

[4] A. Kalsing, G. do Nascimento, C. Iochpe, and L. Thom, "An Incremental Process Mining Approach to Extract Knowledge from Legacy Systems," in *14th IEEE International Enterprise Distributed Object Computing Conference (EDOC'10), 2010*. IEEE, 2010, pp. 79–88.

[5] G. Lewis, E. Morris, L. O'Brien, D. Smith, and L. Wrage, "Smart: The service-oriented migration and reuse technique," CMU/SEI, Tech. Rep. CMU/SEI-2005-TN-029, Sept 2005, Available from:http://www.sei.cmu.edu/reports/05tn029.pdf.

[6] R. Khadka, B. Sapkota, L. F. Pires, M. van Sinderen, and S. Jansen, "Model-driven development of service compositions for enterprise interoperability," in *Enterprise Interoperability*, M. van Sinderen and P. Johnson, Eds. Springer, 2011, vol. 76, pp. 177–190.

[7] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: a research roadmap," *International Journal of Cooperative Information Systems*, vol. 17, no. 2, pp. 223–255, 2008.

[8] R. Khadka and B. Sapkota, "An evaluation of dynamic web service composition approaches," in *Proceeding of the 4th International Workshop on Architectures, Concepts and Technologies for Service-Oriented Computing (ACT4SOC'10)*, Athens, Greece, 2010, pp. 67–79.

[3]http://servicifi.org/

[9] Z. Zhang, H. Yang, and W. Chu, "Extracting reusable object-oriented legacy code segments with combined formal concept analysis and slicing techniques for service integration," in *Proceedings of 6th International Conference of Software Quality (QSIC'06)*. IEEE, 2006, pp. 385–392.

[10] H. M. Sneed, "Integrating legacy software into a service oriented architecture," in *Proceedings of 10th European Conference on Software Maintenance and Reengineering (CSMR'06)*, IEEE. IEEE Computer Society, 2006, pp. 3–14.

[11] S. Li, S. Huang, D. Yen, and C. Chang, "Migrating legacy information systems to web services architecture," *Journal of Database Management (JDM)*, vol. 18, no. 4, pp. 1–25, 2007.

[12] Y. Liu, Q. Wang, M. Zhuang, and Y. Zhu, "Reengineering Legacy Systems with RESTful Web Service," in *Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference, (COMPSAC'08)*. IEEE, 2008, pp. 785–790.

[13] S. Chung, P. Young, and J. Nelson, "Service-oriented software reengineering: Bertie3 as web services," in *Proceedings of the IEEE International Conference on Web Services (ICWS'05)*. IEEE, 2005, pp. 837–838.

[14] A. Almonaies, J. Cordy, and T. Dean, "Legacy system evolution towards service-oriented architecture," in *International Workshop on SOA Migration and Evolution (SOAME'10)*. IEEE, 2010, pp. 53–62.

[15] C. Lawrence, "Adapting legacy systems for soa," Online, Jun 2007, Available from: http://www.ibm.com/developerworks/webservices/library/ws-soa-adaptleg/.

[16] F. Cuadrado, B. García, J. Dueas, and H. Parada, "A case study on software evolution towards service-oriented architecture," in *Proceddings of the 22nd International Conference on Advanced Information Networking and Applications-Workshops, (AINAW'08)*. IEEE, 2008, pp. 1399–1404.

[17] K. Channabasavaiah, E. Yuggle, and K. Holley, "Mirgating to a service-oriented architecture," Online, Dec 2003, Available from:http://www.ibm.com/developerworks/webservices/library/ws-migratesoa/.

[18] A. Umar and A. Zordan, "Reengineering for service oriented architectures: A strategic decision model for integration versus migration," *Journal of Systems and Software*, vol. 82, no. 3, pp. 448–462, 2009.

[19] V. Reddy, A. Dubey, S. Lakshmanan, S. Sukumaran, and R. Sisodia, "Evaluating legacy assets in the context of migration to SOA," *Software Quality Journal*, vol. 17, no. 1, pp. 51–63, 2009.

[20] A. De Lucia, R. Francese, G. Scanniello, and G. Tortora, "Developing legacy system migration methods and tools for technology transfer," *Software: Practice and Experience*, vol. 38, no. 13, pp. 1333–1364, 2008.

[21] S. Brinkkemper, "Method engineering: engineering of information systems development methods and tools," *Information and Software Technology*, vol. 38, no. 4, pp. 275–280, 1996.

[22] N. Gold, M. Harman, D. Binkley, and R. Hierons, "Unifying program slicing and concept assignment for higher-level executable source code extraction," *Software: Practice and Experience*, vol. 35, no. 10, pp. 977–1006, 2005.

[23] A. Hevner, S. March, J. Park, and S. Ram, "Design science in information systems research," *Mis Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[24] S. Brinkkemper, M. Saeki, and F. Harmsen, "Meta-modelling based assembly techniques for situational method engineering," *Information Systems*, vol. 24, no. 3, pp. 209–228, 1999.

[25] I. van de Weerd and S. Brinkkemper, *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications*. Idea Global Publishing, 2008, ch. Meta-modeling for situational analysis and design methods, pp. 38–58.

[26] M. Papazoglou and W. Van Den Heuvel, "Service-oriented design and development methodology," *International Journal of Web Engineering and Technology*, vol. 2, no. 4, pp. 412–442, 2006.

[27] S. P. Lee, L. P. Chan, and E. W. Lee, "Web services implementation methodology for soa application," in *Proceeding of the 4th IEEE International Conference on Industrial Informatics*, 2006, pp. 335–340.

[28] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, "SOMA: A method for developing service-oriented solutions," *IBM Systems Journal*, vol. 47, no. 3, pp. 377–396, 2008.

[29] I. van de Weerd, S. de Weerd, and S. Brinkkemper, "Developing a reference method for game production by method comparison," in *Situational Method Engineering: Fundamentals and Experiences*, ser. IFIP International Federation for Information Processing, J. Ralyt, S. Brinkkemper, and B. Henderson-Sellers, Eds. Springer Boston, 2007, vol. 244, pp. 313–327.

[30] H. Sneed, "Planning the reengineering of legacy systems," *Software*, vol. 12, no. 1, pp. 24–34, 1995.

[31] G. Reijnders, R. Khadka, S. Jansen, and J. Hage, "Developing a legacy to soa migration method," Department of Information and Computing Sciences, Utrecht University, Tech. Rep. UU-CS-2011-008, 2011.

[32] M. Weiser, "Program slicing," in *Proceedings of the 5th international conference on Software engineering*. IEEE, 1981, pp. 439–449.

[33] T. Biggerstaff, B. Mitbander, and D. Webster, "Program understanding and the concept assignment problem," *Communications of the ACM*, vol. 37, no. 5, pp. 72–82, 1994.

[34] MOSCOW, "MoSCoW Prioritisation," Online, 2011, Available from: http://www.coleyconsulting.co.uk/moscow.htm.

[35] S. Johnston, "Modeling service-oriented solutions," Online, Jul 2005, Available from: http://www.ibm.com/developerworks/rational/library/jul05/johnston/.

[36] M. Harman, N. Gold, R. Hierons, and D. Binkley, "Code extraction algorithms which unify slicing and concept assignment," in *Proceeding of the 9th Working Conference on Reverse Engineering*. IEEE, 2002, pp. 11–20.

[37] M. Polan, "Web service provisioning," Online, January 2002, Available from: http://www.ibm.com/developerworks/library/ws-wsht.

[38] R. Khadka, A. Saeidi, S. Jansen, J. Hage, and R. Helms, "An evaluation of service framework for the management of service ecosystem," in *Proceedings of the 15th Pacific Asia Conference on Information System (PACIS'11)*. AIS, 2011, Available at: http://projects.business.uq.edu.au/pacis2011/papers/PACIS2011-087.pdf.

[39] C. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 557–572, 1999.

[40] A. Strauss and J. Corbin, *Basics of qualitative research: Grounded theory procedures and techniques*. Sage, 1990, vol. 1.

[41] H. Boeije, "A purposeful approach to the constant comparative method in the analysis of qualitative interviews," *Quality and Quantity*, vol. 36, no. 4, pp. 391–409, 2002.

[42] S. Thorne, "Data analysis in qualitative research," *Evidence Based Nursing*, vol. 3, no. 3, p. 68, 2000.

[43] Z. Mahmood, "The Promise and Limitations of Service-Oriented Architecture," *International Journal of Computers*, vol. 1, no. 3, pp. 74–78, 2007.

[44] K. Nasr, H. Gross, and A. van Deursen, "Adopting and Evaluating Service-Oriented Architecture in Industry," in *14th European Conference on Software Maintenance and Reengineering (CSMR'10)*. IEEE, 2010, pp. 11–20.

[45] E. Roch, "Soa benefits, challenges and risk mitigation," Online, May 2006, Available from: http://it.toolbox.com/blogs/the-soa-blog/soa-benefits-challenges-and-risk-mitigation-8075.

[46] Scitools, "Understand tool," Online, 2011, Available at: http://www.scitools.com/index.php.

[47] Grammatech, "Codesurfer," Online, 2011, Available at: http://www.grammatech.com/products/codesurfer/academic.html.

[48] WSO2, "Wso2/c++ web service framework," Online, 2011, Available from: http://wso2.com/products/web-services-framework/cpp/.

[49] R. Yin, *Case study research: Design and methods*. Sage Publications, Inc, 2009.

[50] D. Harris, A. Yeh, and H. Reubenstein, "Extracting architectural features from source code," *Automated Software Engineering*, vol. 3, no. 1, pp. 109–138, 1996.

[51] J. V. Geet and S. Demeyer, "Feature location in COBOL mainframe systems: An experience report," in *25th IEEE International Conference on Software Maintenance (ICSM'09)*. IEEE, 2009, pp. 361–370.

[52] S. R. Tilley, D. B. Smith, and S. Paul, "Towards a framework for program understanding," in *4th International Workshop on Program Comprehension (WPC'96)*. IEEE, 1996, pp. 19–28.

[53] L. Feijs, R. Krikhaar, and R. van Ommering, "A relational approach to support software architecture analysis," *Software: Practice and Experience*, vol. 28, no. 4, pp. 371–400, 1998.