A systematic literature review on SOA migration

Citation for published version (APA):

Razavian, M., & Lago, P. (2015). A systematic literature review on SOA migration. *Journal of Software : Evolution and Process*, 27(5), 337-372. <https://doi.org/10.1002/smr.1712>

DOI:

[10.1002/smr.1712](https://doi.org/10.1002/smr.1712)

Document status and date:

Published: 01/01/2015

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

A systematic literature review on SOA migration

Maryam Razavian* and Patricia Lago

Department of Computer Science, VU University Amsterdam, The Netherlands

ABSTRACT

When Service Orientation was introduced as the solution for retaining and rehabilitating legacy assets, both researchers and practitioners proposed techniques, methods, and guidelines for SOA migration. With so much hype surrounding SOA, it is not surprising that the concept was interpreted in many different ways, and consequently, different approaches to SOA migration were proposed. Accordingly, soon there was an abundance of methods that were hard to compare and eventually adopt. Against this backdrop, this paper reports on a systematic literature review that was conducted to extract the categories of SOA migration proposed by the research community. We provide the state-of-the-art in SOA migration approaches, and discuss categories of activities carried out and knowledge elements used or produced in those approaches. From such categorization, we derive a reference model, called SOA migration frame of reference, that can be used for selecting and defining SOA migration approaches. As a co-product of the analysis, we shed light on how SOA migration is perceived in the field, which further points to promising future research directions. Copyright © 2015 John Wiley & Sons, Ltd.

Received 2 April 2014; Revised 11 February 2015; Accepted 19 February 2015

KEY WORDS: migration; service orientation; systematic literature review; knowledge management

1. INTRODUCTION

Service Orientation promises to rehabilitate pre-existing legacy assets by encapsulating them as added-value services. Such services embed business functions from legacy assets on diverse hardware and software platforms. Some of those legacy assets may be legacy systems while others could be technically healthy and value-adding applications, business processes, or data of an enterprise. Migrating those legacy assets into services that can smoothly operate with modern technology has become one of the major challenges of service engineering methodologies [1, 2].

Over the last decade, migration to services has received much attention in both academia and practice. With so much hype surrounding SOA, it is not surprising that researchers interpreted the concept in many different ways and proposed different approaches to SOA migration. Moreover, the many approaches were developed to pursue very different goals. Accordingly, those approaches in their descriptions have different terminology and focus. Some of the differences, however, are essential. Such essential differences include approaches designed for modernizing legacy code through automatic code transformation versus approaches for reengineering pre-existing business processes interacting with stakeholders, approaches with main focus on reverse engineering versus approaches for forward engineering ideal services while taking existing code as a reference.

On the other hand, migration approaches must have much in common as they deal with the same basic problem, and emphasize similar goals. For example, many approaches, which possibly use different terminology and are designed in various domains, deal with the single problem of *dealing*

*Correspondence to: Maryam Razavian, Eindhoven University of Technology, PAV D.8a, P.O. Box 513, 5600 MB Eindhoven The Netherlands.

[†]E-mail: m.razavian@tue.nl

with *heterogeneity* and emphasize *integration of legacy systems*. It is thus of significant interest to understand the fundamental commonalities and differences between different migration approaches and to develop a reference model embracing categories of such approaches. Such a reference model would help us to better understand the fundamentals of existing approaches as well as provide a common ground for developing new approaches better suited to specific needs.

With this goal in mind, we conducted a systematic review that extracts migration categories existing in the field. We chose systematic review as our research method in aggregating existing SOA migration approaches for two main reasons: (i) it encourages methodologically rigorous results in identifying and selecting the relevant studies on SOA migration, and (ii) the strength of systematic reviews in minimizing the bias in the review process enhances the extraction of sound and meaningful categories of migration approaches. Through in-depth analysis of the studies focused on migration activities carried out by different approaches and the knowledge assets used or produced, we arrived at a reference model for SOA migration, called *SOA migration frame of reference*. The basic idea behind the *frame of reference* is to generalize the categories of knowledge and activities that underlie migration approaches. Such frame of reference provides a thinking framework for one to reason about ‘what approach suits the migration’; this way it enhances understanding of how SOA migration can be carried out and what should drive the migration. As a co-product of the analysis, insight was gained about the current focus of SOA migration in the field. This insight reflects how currently SOA migration is perceived in academia and pinpoints promising research directions.

Migration of legacy systems to SOA has been the focus of few literature review papers. Almonaies *et al.* [3] report a brief overview of the SOA migration approaches by categorizing them into four groups of replacement, redevelopment, wrapping, and migration. Khadka *et al.* [4] provide a systematic literature review on evolution of legacy systems focusing on identifying existing techniques, methods, and practices specific to different phases of SOA migration. While both works focus on technological and operational aspects of migration, in this paper we document what seems to be fundamental in academic SOA migration approaches, both from the process and knowledge perspectives.

This paper substantially extends previous publication [5] in the following ways: (i) it covers more recent work, resulting in 31 new primary studies; (ii) the analysis is both broader and more in-depth. By broadening the scope of analysis, this paper thoroughly consolidates the categorization into a novel frame of reference that addresses knowledge elements, knowledge conversions, and activities. This adds up to the previous publication that only covers the categories of activities carried out. Furthermore, the in-depth analysis of the common types of migration approaches reveals the dominant view of SOA migration in-the-field; (iii) using a real-world example, this paper illustrates the application of the frame of reference to a level that could be adopted for definition of one own migration approach.

The paper is organized as follows. We introduce the background in Section 2. Section 3 introduces study design. In Section 4, we present the categories of SOA migration. Then Section 5 introduces the frame of reference and illustrates its application. Section 6 discusses the threats to validity and our additional observations. Finally, Section 7 concludes the paper.

2. BACKGROUND

We introduce here the context and basic ideas underlying this study.

2.1. Two-view SOA migration approach

To analyze and categorize the migration approaches, we represent the migration approaches using views. In this section, we explain the vision behind the view-based representation of the migration approaches.

An approach for SOA migration states a path from the As-Is state (i.e., As-Is legacy assets) to the To-Be state (i.e., To-Be services) [6]. To define such an approach, two questions have to be answered: (i) what knowledge is available about the As-Is and To-Be states, and (ii) what activities

need to be performed to carry out the migration [6]. We argue that each of these questions reflects a distinct view on the approach.

The notion of views has been introduced in the software architecture field since the early 1990s [7]. An architectural view, there, is a partial representation of a system, from the perspective of a stakeholder's concern. Following the same perspective, we define a view of the SOA migration approach as a partial representation of the migration approach from the perspective of a particular concern. We propose the following views for SOA migration approach: (i) *Knowledge view* highlights the types of knowledge that shape and drive the migration, and (ii) *Activity view* reflects *what needs to be performed* in SOA migration [8]. We here discuss our rationale behind choosing this two-view representation.

1. *SOA migration is a reengineering problem.* We consider the problem of migration of legacy assets to SOA as a reengineering problem. In this view, we follow a line of thought shared among various researchers. In [9], migration is defined as a modernization technique that moves the system to a new platform while retaining the original system's data and functionality. According to [10], reengineering is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form. The commonalities among these two definitions are considerable. In practice, the notions of 'legacy migration', 'integration', and 'architectural recovery', which all deal with legacy applications, are considered as approaches to reengineering.
2. *Knowledge defining As-Is and To-Be states.* According to [6], any reengineering effort embraces an approach indicating how to move from the As-Is state to the To-Be state. As such, understanding the *As-Is and To-Be states* is essential for carrying out the migration. To determine the migration approach, the concern of what knowledge elements define the As-Is and To-Be states have to be addressed. This concern is reflected in the *knowledge view*.
3. *Activities to move from As-Is state to To-Be.* To move from the As-Is to the To-Be state, one needs to identify the best-fitting *set of activities* to perform the reengineering [6]. To this end, the decisions regarding the best-fitting activities have to be addressed in the migration approach, that is, the focus of *activity view*.

Figure 1 gives an example of a two-view migration approach. The knowledge view (Figure 1A) indicates the input and output knowledge elements. The available knowledge is shown in *white*, and the required knowledge that is not available is in *gray*. The knowledge conversions, shown using arrows, indicate what input knowledge is required to create a certain output. The conceptual framework behind knowledge view is presented in Section 2.2.

Figure 1. An example of two-view approach representation. (A) Knowledge view and (B) activity view.

The activity view (Figure 1B) represents the activities covered by the migration approach. The conceptual framework behind the knowledge view is introduced in Section 2.3.

The meta-model presented in Figure 2 represents the concepts of the two-view migration approach and their relationships. The meta-model shows that the two views are interrelated. In the following, we further explain these concepts in both views. We point out to the concepts of the meta-model in typewriter (e.g., activity).

2.2. A framework for representing knowledge view

The knowledge framework shown in Figure 3 is a skeleton of generic knowledge elements and knowledge conversions that shape and drive the migration. In the following, we describe these categories.

2.2.0.1. Categories of knowledge elements. As shown in Figure 3, knowledge elements are classified into four categories. This categorization stems from the following generic distinctions in knowledge elements: (i) the distinction between *tacit* and *explicit* knowledge and (ii) distinction between *problem-related* and *solution-related* knowledge. Related to the first distinction, Nonaka and Takeuchi [11] refer to two main types of knowledge, *tacit* and *explicit*. Tacit knowledge is a type of knowledge that a human is not able to express explicitly, but is guiding the behavior of the human. Explicit knowledge is knowledge that we can represent, for example, in reports, books, talks, or other formal or informal communication. The distinction between problem-related knowledge and solution-related knowledge has been pointed out in a number of related works [12, 13]. We define these two types of knowledge as follows: (a) problem-related knowledge regards the type of knowledge presenting the analysis of the problem, the problem decomposition, and the world in which it is located; (b) solution-related knowledge addresses the solution to a specific problem domain. Because the two distinctions are orthogonal, the combination of the two results in the following four generic categories of knowledge are illustrated as quadrants in Figure 3.

2.2.0.2. Categories of knowledge conversions. Based on the categories of knowledge elements presented in Figure 3, in total, there are 16 possible types of knowledge conversion that are determined by pairing one of the four conversions between tacit and explicit with one of the four conversions between problem-related and solution-related knowledge. To illustrate the knowledge conversions between tacit and explicit, we use the terms introduced by Nonaka and Takeuchi [11]

Figure 2. Meta-model of two-view approach representation.

Figure 3. A framework for knowledge view.

including: *socialization* (tacit to tacit), *externalization* (tacit to explicit), *internalization* (explicit to tacit), and *combination* (explicit to explicit).

2.3. A framework for representing activity view

The activity framework, called SOA-MF [14], is a skeleton of generic activities representing what needs to be performed in a migration project. These activities mainly transform a number of knowledge elements.

Inspired by the prominent reengineering horseshoe model [15], the activity framework consists of three processes: reverse engineering and transformation and forward engineering. SOA-MF (Figure 8A) follows a horseshoe model by first recovering the lost abstractions and eliciting the legacy fragments that are suitable for migration to SOA (reverse engineering), altering and reshaping the legacy abstractions to service-based abstractions (transformations), and finally, renovating the target system based on transformed abstractions as well as new requirements (forward engineering). Reverse engineering starts from existing implementation and continues with extracting the design entities (*code analysis*), recovering the architecture (*architectural recovery*), and recapturing abstractions in requirements or business models (*business model recovery*). Within the transformation process the activities of *design element transformation*, *composition transformation*, and *business model transformation*, respectively, realize the tasks of reshaping design elements, restructuring the architecture, and altering business models and business strategies. The forward engineering process involves the activities of *service analysis*, *service design*, and *service implementation*. Finally, the framework covers different levels of abstraction including *concept*, *composite design element*, *basic design element*, and *code*.

3. STUDY DESIGN

In our study, we followed a systematic literature review process based on the guidelines proposed in [16, 17]. The protocol specified the method to be followed in terms of the research questions, the search process, and the data to be extracted. We developed the protocol by running a pilot study.

3.1. Pilot study

The goal of the pilot study was to execute the protocol on one of the libraries (IEEE Xplore) and check if the protocol was generally rigorous enough and to improve it where necessary. We initially developed the search string inspired by relevant literature and prior systematic reviews in the field of SOA [18]. In order to validate the search string, we ran the string on IEEE Xplore library and checked if the search string can detect a pilot set of 25 known relevant studies. The pilot study uncovered the problems with the search string as it overlooked some of the known studies. For example, the term ‘re-use’ was not included in the initial search string, and the pilot study served to include this term to include known studies. In addition, the pilot study served to identify and add the alternative spellings of a specific term (re-engineering and reengineering). To make sure that the data analysis method best fits this study, we consulted an expert in qualitative data analysis methods. Finally, the resulting protocol was checked by senior researchers experienced in software engineering and systematic reviews. The feedback helped us to improve the protocol. Examples are tuning the inclusion/exclusion criteria and better formalizing the data extraction.

3.1.1. Research questions. The systematic review aimed at extracting the categories of SOA migration approaches. To achieve this goal, we defined the following research questions:

What approaches regarding legacy to SOA migration have been proposed in research community so far? In particular, the following aspects facilitate characterizing the SOA migration approaches:

- (RQ-I) What are the activities carried out?
- (RQ-II) What is the sequencing of the activities?
- (RQ-III) What are the knowledge elements that are used and produced?

3.1.2. Search strategy. As the first step of systematic search, three main keywords were built from our research question, namely: *migration*, *legacy assets*, and *SOA*. Furthermore, for each keyword, a set of related terms addressing synonyms and alternative spellings were identified. Based on the keywords and their related terms we defined the following search string:

(SOA or ‘service-oriented’ or ‘service-computing’ or ‘service-based’ or ‘service-centric’ or ‘service’ or ‘service-engineering’ or SOSE) AND (‘legacy code’ or ‘legacy system’ or ‘existing system’ or ‘legacy component’ or ‘existing code’ or ‘existing asset’ or ‘existing component’ or ‘pre-existing code’ or ‘pre-existing system’ or ‘pre-existing component’ or ‘legacy software’ or ‘existing software’ or ‘pre-existing software’) AND (migration or modernization or modernisation or transformation or reengineering or re-engineer or evolving or reuse or ‘service mining’ or ‘service identification’ or ‘service extraction’)

Data sources We used the following libraries as main resources: IEEE Xplore, ACM Digital Library, ISI Web of Knowledge, SpringerLink, ScienceDirect, and Wiley InterScience Journal Finder.

Search process As major venues on service-oriented systems started in 2003 onwards (ICSOC¹ in 2003, SCC² in 2004, and SOSE in 2005³), we decided to set 2000 as the start date to minimize the chance of overlooking relevant studies. This implies that a study is selected as a candidate study if it contains the search terms in the abstract and is published between Jan 1, 2000 and Dec 31, 2013 (i. e., when this review was conducted). We applied the search terms to titles and abstracts considering that they provide a concise summary of the work. This decision was validated by running the search string on data sources and checking if the pilot studies are retrieved. Because of lack of standardization between the digital libraries, we had to adapt the search string for each data source. To ensure that search strings are adapted correctly, two reviewers independently ran the search process, and this way, the overlooked articles were identified.

¹International Conference on Service Oriented Computing

²International Conference on Services Computing

³International Workshop on Service-Oriented System Engineering

Figure 4. Selection of primary study process.

3.1.3. Selection of primary studies. Articles that satisfied the following inclusion criteria were selected as a primary study.

- (I1) A study that proposes approaches addressing migration of legacy assets to services. Rationale: We are interested in studies that are about SOA migration.
- (I2) A study that is developed by either of academics and practitioners. Rationale: Both academic and industrial migration approaches are relevant to this study.
- (I3) A study that is published in software engineering field. Rationale: We seek for approaches specifically addressing migration of pre-existing software to services.
- (I3) A study that is peer-reviewed. Rationale: A peer-reviewed paper guarantees a certain level of quality and contains reasonable amount of content.
- (I4) A study that is written in English. Rationale: For feasibility reasons, papers written in other languages than English were decided to be excluded.

Studies that met the following criteria were excluded from the review:

- (E1) A study that is not about migration *to services*. Rationale: Studies that support migration to other types of target systems (not to service-based) should be excluded.
- (E2) A study that does not address migration *from existing legacy assets*. Rationale: Studies that do not address migration, that is, reuse of pre-existing assets needs to be excluded.
- (E3) A study that does not specifically propose *a solution for migration*. Rationale: Studies that do not specifically provide a solution for the migration problem must be excluded. For instance, studies presenting challenges on SOA migration are out of scope of this work.

3.1.4. Included and excluded studies. In total, we found 501 publications, whose abstracts contain the keywords defined in the search string. As shown in Figure 4, out of 501 papers, only 170 studies that appeared to be completely irrelevant were excluded based on their title. At the second stage, by applying inclusion/exclusion criteria on the abstracts, 136 studies were included. We found that abstracts could provide little information and consequently it was not always obvious that the study provides a specific approach for SOA migration. Therefore, in this stage, we included all studies that focus on SOA migration. Finally, the full text of studies was reviewed against the inclusion/exclusion criteria and 84 papers were included in our review. Although we identified 84 articles by this search process, some articles were earlier or short versions of other articles. If the same migration approach were included in multiple publications, it could bias the results as a single approach is considered as multiple primary studies. To resolve this issue, we examined the publications written by the same set of authors and affiliations to see if they were reporting the same migration approach. In total, 9 papers were excluded on this basis and the publication that was more complete was included. We further looked for the publications that represent the same approach that none of them is more complete but they are complementary. None of the studies fell under this category. Thus, we ended up with 75 primary studies.

3.1.5. Search result management. The reference details of each study was recorded using JabRef [19]. For each subsequent stage of the selection process, described in the following, a separate JabRef database was established.

3.1.6. Quality assessment of primary studies. We ranked the primary studies considering their level of applicability in practice. This can bring insight into the extent to which the SOA migration approaches proposed in the primary studies are likely to be applicable in practice. We used the three following scales to assess the applicability of the primary studies:

- *High*: The approach is applied in a real-world case study in industrial or organizational settings.

- *Medium*: The proposed approach is explained and discussed using descriptive examples.
- *Low*: The approach is not applied in practice, nor is its applicability in practice exemplified.

3.2. Data extraction

Data from the 75 primary studies was extracted and stored in an extraction form (Table A.I in Appendix). The extraction forms helped us to store details of the primary studies as well as their migration approach. Regarding the migration approach, the summary of the approach, the activities, and their input/output knowledge elements were recorded in the data extraction forms.

3.3. Data analysis method

This study seeks for *categorization* of migration approaches based on the views of activity and knowledge. To this end, we mapped each of the migration approaches on the two views. The question that we faced was how to systematically analyze and map the primary studies in such a way that the meaningful categorizations are determined. We chose coding [20] as our qualitative analysis method. To carry out the mappings systematically, we devised the coding procedure represented in Figure 5.

Codes For creating the codes, we followed the suggestion of Miles and Huberman [20], namely, to have an initial set of codes, called ‘start-list’, that is, refined during the analysis. Our start-list stems from an SOA migration framework (called SOA-MF), described in Section 2.3. For each of the comprising activities of SOA-MF, shown in rounded rectangles in Figure 1, a code was created (e.g., for the activity *Code analysis* the code *ACT-CoAn* was created). Based on the categorization of knowledge presented in Section 4.4.1, we created a code for each of the categories, namely, *problem-related*, *solution-related*, *tacit*, and *explicit*. Table A.II represents the start-list of codes along with their descriptions. The start-list, of course, was refined during the analysis as the new insights occurred during the analysis.

Coding procedure Inspired by the procedure proposed by Lincoln and Guba [21], we devised a coding procedure to systematically codify the primary studies and refine the codes when needed. By codifying the primary studies using the start-list in a step-by-step manner, the coding procedure enables identifying the two-view representation of each study. Furthermore, each step of the procedure guides the relevant refinements to the code as described in the following.

- *Step 1: Filling-in activities (RQ-I)*. This step codifies the activities involved in the migration. Filling in refers to attaching codes to activities and creating new codes when needed. Accordingly, if an activity did not fit any of the existing codes, a relevant code was added. For instance, initially a code representing the *code transformation* activity did not exist, because it was originated from the existing theory on architectural recovery [15] that did not consider transformation at code level. During the early stages of this analysis, we observed that there are some migration approaches that translate the legacy code to Web services. In order to characterize such approaches, a code called ‘code transformation activity’ was added. Eventually, the code transformation activity was added to the transformation process of SOA-MF as well (Figure 8A).
- *Step 2: Filling-in/surfacing knowledge elements (RQ-III)*. The second step codifies the input/output knowledge elements of activities. This way, the knowledge elements that are used and/or produced throughout the migration approach are extracted. Surfacing, as the name implies, accounted for grouping codes into categories and/or the emergence of new categories of knowledge elements.
- *Step 3: Filling-in/surfacing knowledge conversions (RQ-III)*. Once the knowledge elements used in each study are identified, this step identifies the interactions among those knowledge elements. By further surfacing the knowledge conversions, the new categories are identified.

Figure 5. Coding procedure.

- *Step 4: Filling-in/surfacing sequencing of activities (RQ-II).* The fourth step identifies the sequence of activities as well as the categories related to types of sequencing.
- *Step 5: Bridging patterns.* The last step discovers the patterns among the categories in each of the activity view and the knowledge view.

Conducting coding procedure Once the codes and coding procedure were decided, the analysis could be undertaken. We went through the primary studies and applied the steps. By following the first four steps, we were able to identify the two-view migration approach of each of the primary studies. Once all the primary studies were codified, by undertaking step 5, general findings were obtained. In the succeeding texts, we describe in detail how each step was undertaken.

By following step 1 of our coding procedure, we labeled each activity using the relevant codes. As it is evident from the description of the activities in Table A.II, each activity is considered as a transformation of a specific input to a specific output. For instance, an activity (or series of activities) that transforms a *code* to a *legacy element* should be codified as *code analysis*. By codifying the activities of all primary studies, we identified the coverage of activities of each migration approach on SOA-MF. By doing this, we were able to identify the activity view of each primary study.

Following step 2, we coded each identified knowledge element and then mapped it to the categories of the knowledge elements introduced in Section 2.2 (i.e., filling in). By comparing and studying the coded element and consequently identifying new categories, we further classified the knowledge elements (i.e., surfacing). For instance, by comparing and categorizing different types of knowledge describing the system implementation, we identified four main types as illustrated in Figure 14. These new categories are described in Section 4.4.1.

In step 3, we filled in the knowledge conversions. By comparing the knowledge conversions, we identified different categories of conversions. These categories are described in Section 4.4.2.

In step 4, we captured the sequencing of activities by extracting the ordering of knowledge conversions identified in the previous step. As an example, the sequencing of activities shown in Figure 8B (F4(b)) was recognized by identifying the following knowledge conversions: Code → Design Model → Service Model → Service Implementation. In short, we identified two categories of sequencing, which we describe in Section 4.3.

In step 5, we went through the activity and knowledge views of all primary studies and searched for meaningful patterns in types of migration approaches. As a result, we identified a number of interesting observations and lessons learned. This is the topic of Section 6.2.

4. RESULTS

By analyzing the 75 primary studies, we answered our three research questions. In the following, we first present an overview of the primary studies. Then, we answer each of the research questions.

4.1. Overview of the included studies

Table A.III presents the overview of primary studies.

4.1.0.1. Primary studies over time. Figure 6 shows the number of publications per year. The first paper on SOA Migration was published in 2000, although until the early 2003 the topic received fairly little attention. Between 2003 and 2008, there was an increase in the number of publications while this number dropped gradually between 2009 and 2011. Since 2011, however, an increase in the studies can be seen. Interestingly, the curve shown in Figure 6 reveals what we call *SOA Migration Cycle*. This cycle recalls the *Gartner hype cycle* [22] (both illustrated in Figure 7). While the *Gartner Hype Cycle* (thicker line in the figure) presents the maturity level of the emerging technology and its application (here SOA technology), the *SOA Migration Cycle* represents the number of papers published on this topic.

Figure 6. Distribution of primary studies by year.

Figure 7. Gartner Inc. hype cycle and SOA migration cycle (www.gartner.com/technology/research/hype-cycles/).

Using the two cycles, we explain the distribution of the primary studies over years as follows. When SOA faced *peak of inflated expectations*, researchers were ambitious to devise SOA migration approaches. Shortly after, SOA quickly fell into a period of *disillusionment*, marked by a considerable decrease in the number of papers. This is followed by *enlightenment* phase of SOA — when there is increasing awareness about how the SOA can benefit enterprises — where we observe increasing interest on the SOA migration topic.

4.1.0.2. Primary studies resources. Table I gives a breakdown of where the primary studies are published.

4.1.0.3. Quality of the primary studies. As noted, we assessed the quality of primary studies based on the extent to which the SOA migration approaches proposed in the primary studies are likely to be applicable in practice. In this regard, most of the primary studies have included some type of evaluation of their applicability. In total, 14 out of 75 primary studies did not provide any explanation on the applicability of the approaches. Fifteen of the primary studies described the approach with example case studies. Forty-six of the primary studies were applied in real-world case studies and experiments.

Table I. Distribution of primary studies by resources.

Digital library	Number	Percent
IEEE Xplore	36	59%
ACM Digital Library	4	6%
SpringerLink	15	24%
ScienceDirect	4	5%
Wiley InterScience Journal Finder	1	2%
ISI Web of Science	4	6%

4.2. RQ-1-What activities are carried out in SOA migration?

By following step 1 of our coding procedure, we identified the activity view of each primary study. Figure 8 illustrates the schematic forms of activity views. As an example, F4(b) is a schematic form of the activity view shown in Figure 8B. By thoroughly analyzing the activity views, we identified a set of meaningful relationships among the approaches with graphically similar views and their migration objectives and solutions. For instance, the migration approaches that have an activity view similar to the one schematically shown in F1(a) in Figure 8C provide a common solution for migration (i.e., translating or wrapping the legacy system as-a-whole to a service). More precisely, thanks to SOA-MF, the migration approaches that include conceptually similar activities, have graphically similar activity views as well. By considering similar SOA-MF coverage patterns, we clustered different migration approaches into eight distinct families of SOA migration approaches. Figure 8C illustrates the schematic form of distinguished activity views that are dedicated to each family.

The families, as shown in Figure 9, have various sizes. Table A.IV in Appendix, detailing the families, reveals that 28% primary studies fall into F4 family, whereas only 4% of primary studies belong to F7 family. In the remainder of this section, we describe each family in the following way: (1) the *family at a glance* provides a general description of ‘what the activity view of each family implies’, and (2) the *observations* include explanation of the observations related to ‘what migration entails’ in each family.

4.2.0.1. Code transformation family (F1).

Family at a glance The activity view representing this family (simplified in Figure 8C, F1(a)), reflects the following feature: out of the three processes, the migration approach is limited to transformation at system level. Accordingly, the existing legacy code is transformed to service-based implementation.

Observations In this family, migration entails moving the legacy system as-a-whole to a service-oriented platform or technology, without decomposing the existing system. We identified two main categories in this family: (1) transforming the whole code to Web services (e.g., Varga *et al.* [23]),

Figure 8. SOA migration families [5]. (A) SOA-MF, (B) mapping of F4(b) on SOA-MF, and (C) migration families. MF, migration framework.

Figure 9. Distribution of primary studies among families and arch-shaped and bowl-shaped sequencing.

and (2) wrapping the whole application as a Web service [24]. The problem addressed by the first category is to translate a legacy code to a Web service implementation. The second category embraces encapsulating the interfaces of the existing application to a (Web) service interface.

4.2.0.2. Service identification family (F2).

Family at a glance In this family, the transformation process is not covered, meaning that reshaping of the legacy elements to service-based elements is not realized. The reverse engineering process is carried out in all family members, while forward engineering occurs only in some (Figure 8C, F2(c and d)).

Observations In this, family migration is limited to the identification of the candidate services in the existing legacy assets. This family uses reengineering activities to identify the elements that are candidates for migration to SOA.

4.2.0.3. Business model transformation family (F3).

Family at a glance In this family, the reverse engineering and the forward engineering processes are not covered. Migration is realized by the transformation process, carried out at concept level.

Observations Based on the types of transformation at concept level, we found two main categories in this family: (1) approaches providing a meta-process for migration (e.g., [25–27]) and (2) approaches with business process reengineering, for example, [28]. The main goal of the meta-process category is to *support the decision regarding ‘how to perform migration’*. The constituent activities of these approaches support decision making on the migration approach itself. Due to the orthogonal view of this category on the migration approach, we recognize this category as a ‘meta-process’. Despite having the same coverage pattern on SOA-MF as the first category, we found that the business process reengineering category of this family reflects a different perspective on SOA migration: *altering the business process of the existing legacy assets to serve as a basis for top-down service development*. The difference among these two categories motivates a refinement to SOA-MF. Such extension is addressed in [8] as orthogonal activities about how to plan migration.

4.2.0.4. Design element transformation family (F4).

In this family, the transformation process only occurs at ‘basic design element’ level (e.g., modules or classes). Similarly, reverse and forward engineering processes, if covered, are also limited to this level.

Observations Migration in this family is limited to reshaping the existing legacy elements to the service-based elements. More precisely, a set of legacy elements, extracted by means of the ‘code analysis’ activity or simply known beforehand, are transferred to a set of services or service-based elements. For instance, a ‘component specification’ is altered to ‘service specification’ (e.g., Li *et al.* [29]), or a ‘module’ is reshaped to a ‘service’ (e.g., Sneed [30]) or ‘segment of code in the persistence layer’ is transformed to a ‘data service’ (e.g., del Castillo *et al.* [31]).

4.2.0.5. Forward engineering family (F5).

Family at a glance This family fully covers the forward engineering process, whereas transformation and reverse engineering processes occur at ‘basic design element’ level.

Observations The main focus of this family is on development of service-based systems starting from the desired business processes. This family uses the reverse engineering only to locate the functionality of services identified in forward engineering process. *As such the migration entails Top–Down service-based development while locating the realization of the required business functionalities and transforming them to services.*

4.2.0.6. Design and composite element transformation family (F6).

Family at a glance The three migration processes occur in the two levels of the ‘basic design element’ and ‘composite design element’, meaning that the members include both design element and composition transformations.

Observations What characterizes this family is having transformation at both level of ‘basic design element’ and ‘composite design element’. This entails altering legacy elements to services (i.e., design element transformation) as well as reshaping the structure and the topology of legacy elements to realize new service compositions (i.e., composition transformation). *Migration, here, embraces recovering and refactoring of the legacy architecture to the service-oriented architecture as well as reshaping the legacy elements to service-based elements.*

4.2.0.7. Pattern-based composition transformation family (F7).

Family at a glance Migration only includes the transformation process at ‘composite design element’ level. This implies that the architecture of the existing system is altered or configured into the service-based architecture.

Observations A common feature in this family is using ‘patterns’ for transforming the existing architecture to service-based architecture. Patterns are inherently reusable solutions that are here used to extract services or facilitate transformations of legacy elements to services. *Migration here entails pattern-based architectural transformation to SOA.*

4.2.0.8. Forward engineering with gap analysis family (F8).

Family at a glance The transformation process, in this family, occurs in the three levels of ‘concept’, ‘composite design element’, and ‘basic design element’. As shown in Figure 8C (F8(a)), the forward engineering process covers the activities of ‘service analysis’ and ‘service design’ whereas the reverse engineering process is not covered.

Observations This family mainly focuses on top–down service development, starting from extraction of the business model of the target system and further designing service compositions and services. What distinguishes this family from pure top–down service development approaches is that at each abstraction level (including concept, composition, and design level) a comparison (a gap analysis) among the new and the pre-existing artifacts occurs. This comparison serves to assess how the desired business services can be realized by exploiting pre-existing capabilities. *The migration here entails Top–Down service development while assessing the reuse opportunities at all abstraction levels.*

4.3. RQ-II-What is the sequencing of the activities?

The previous section described what activities are covered in the primary studies. Here, we focus on the order in which the covered activities are carried out. Following step 4 of the coding procedure, we identified the sequencing of the activities of each migration approach. In doing this, we observed dissimilar sequencings in the approaches that are composed of similar activities. Figure 10 illustrates some examples of different sequencing on the schematic shapes. Considering the graphical representation of the sequencings, we observed two main categories in the primary studies: *arch-shaped* and *bowl-shaped* approaches.

Figure 10. Arch-shaped approaches.

Figure 11. Bowl-shaped approaches.

1. *Arch-shaped approaches.* The sequencing of activities in this category resembles an *arch* (Figure 10), starting from reverse engineering process. Arch-shaped approaches share the following similarity: the As-Is state (characterized by legacy assets) is what drives the migration. Moreover, we found that the arch-shaped approaches aim at renovating the existing legacy system to reconstitute it in the new form of SOA. Consequently, they mainly focus on how to adapt the legacy assets to the SOA environment. To this end, the *reverse engineering* process realizes understanding of the existing system; *transformation* process specifies how to restructure the legacy assets while the *forward engineering* process realizes the restructuring.
2. *Bowl-shaped approaches.* The illustration of the knowledge conversions on the mappings has a shape similar to a bowl (Figure 11). Unlike the arch-shaped approaches, the bowl-shaped ones start from the forward engineering process. The To-Be state (characterized by requirements or properties of service-based system) is the main driver of the migration. Here, the main goal of migration is to facilitate reuse in building new service-based systems. This goal changes the order in which the three processes are carried out. Accordingly, the forward engineering process realizes the service-based development; to do so, the reverse engineering process facilitates identifying reusable legacy assets and the transformation process reshapes the legacy elements to service-based elements. When making compromises, there are some approaches that give priority to To-Be state and as such are driven by characteristics of ideal state.

As shown in Table A.V, 65% of the primary studies are arch-shaped. Considering the families, all the members of F1 and F4 family are arch-shaped, as they are only driven by the As-Is state (Figure 9). Being in minority (35%), bowl-shaped approaches only dominate F5 and F8 families, which are driven by the To-Be state. Figure 12 illustrates a detailed overview of the distribution of primary studies among families and bowl-shaped or arch-shaped sequencing.⁴

4.4. RQ-III-What are the knowledge elements that are used and produced?

By following steps 2 and 3 of our coding procedure, we identified the knowledge view of each primary study. The knowledge view frames two aspects: (i) what knowledge elements are used in the migration,

⁴Because the meta-process category of F3 family addresses SOA migration at meta-process level, the arch-shaped or bowl-shaped categorization is not applicable to the members of this family.

Figure 12. Distribution of primary studies among families and arch-shaped and bowl-shaped sequencing.

and (ii) how those elements are created. By comparing and categorizing the knowledge views, we obtained the *knowledge elements* that are typically used or produced in the different migration approaches, as well as the types of *knowledge conversions* that underlie creating new knowledge. This is the topic of this section in which the typical knowledge elements and knowledge conversions in SOA migration are discussed.

4.4.1. Knowledge elements. Figure 13 schematically represents the generic knowledge elements found in the primary studies. Using the categorization introduced in Section 2.2, we classified the knowledge elements into four categories over the two dimensions of (i) *tacit* versus *explicit* knowledge and (ii) *problem-related* versus *solution-related* knowledge.

Figure 13. Categories of knowledge elements.

In the rest of this section, we will describe each of these categories and their constituent knowledge elements. Figures 14–16 together represent the conceptual models of the knowledge elements and the associations among them. It should be noted that these conceptual models do not cover all possible knowledge elements that can shape the migration, but it only illustrates the ones that were observed in the primary studies.

4.4.1.1. Solution-related explicit knowledge. This category encompasses all externalized knowledge addressing design and implementation of both pre-existing and target systems. This category is itself categorized into code-related and design-related knowledge.

1. *Code-related knowledge.* We found that more than 50% of the migration approaches use knowledge about the implementation of both legacy and target service-based systems, which constitute an important source of knowledge for migration (Table A.VI). We found that this type of knowledge, called *code-related knowledge*, encompasses spectrum of content for *code document* (e.g., O'Brien *et al.* [32]; Rodriguez *et al.* [33]), *code grammar* (e.g., Bodhuin and Tortorella [34]; Sneed *et al.* [35]), *code model* (e.g., Cuadrado *et al.* [36]; Liu *et al.* [37]) and the *code* itself (e.g., Sneed *et al.* [35]).
2. *Design-related knowledge.* The knowledge explaining the design of both the pre-existing system (i.e., As-Is design) and the target service-based system (i.e., To-Be design) is one of the key inputs of migration approaches. Different migration approaches focus on different aspects of software design. For instance, to select the legacy elements to be migrated as a service, some approaches take the functional decomposition of legacy components (components model) as knowledge input whereas some others use the interactions between parts of the legacy system (interaction model). This type of knowledge is design information that is recorded within the *software design descriptions*. The software design is described using different *design elements* (Figure 15).

Figure 14. Code-related knowledge.

Figure 15. Design-related knowledge.

Figure 16. Problem-related knowledge.

- (a) *Design elements*, each addressing a different aspect of design and are composed of various *design entities*. Design entities capture key elements of a software design. Examples of design entities include, but are not limited to, the following: components, connectors, classes, libraries, modules, and data stores. We found the following types of design elements and their constituent design entities (Table A.VII).
- *Functional decomposition model*: we found that decomposition of the software system into elements constitutes the key knowledge element of 29% of the primary studies. These studies use different models such as *component models* (e.g., Chen *et al.* [38]; O'Brien *et al.* [32]) or *service models* (e.g., Lewis and Smith [25]; Chen *et al.* [39]) to describe the decomposition of the software system into elements. They mainly use this knowledge to identify where a specific functionality is located.
 - *Structural model*: describes the internal structure and organization of design by its constituent elements such as classes, interfaces, and their relationships. Twenty-seven percent of the migration approaches (e.g., del Castillo *et al.* [31]; Canfora *et al.* [40]) exploit this type of knowledge to reverse engineer or to localize a functionality in pre-existing systems.
 - *Persistent data model*: concerns data structures, data content, and data types. Examples of such design elements are conceptual data models (e.g., Li *et al.* [29]) or database schemas, del Castillo *et al.* [31]). This type of knowledge is mainly used by the approaches that aim at identification and extraction of data services.
 - *Interaction model*: describes the nature of the interaction between different components and parties. Some migration approaches (e.g., El-Ramly *et al.* [41]; Canfora *et al.* [40]; Cuadrado *et al.* [36]) exploit interaction models to identify candidate legacy elements for migration to SOA.
 - *Pattern*: This type of knowledge addresses reusable design ideas as patterns. Patterns address the design ideas that can be reused in all three migration processes. Examples of these patterns are SOA patterns [42, 43], design patterns [31, 44], and architectural styles [32, 38, 45].
- (b) *Design constraints*, which specify a rule or restriction on design elements or provide characteristics of design elements, also shaped SOA migration approaches. We found that using different design constraints such as coupling or cohesion; the migration approaches specify the characteristic of legacy elements that are suitable to be migrated to services. For instance, the legacy elements that are *loosely coupled* [32, 36, 38, 46, 47], or are *well granular* [38, 39, 48, 49].

4.4.1.2. *Problem-related explicit knowledge*. This category pertains to problem domain knowledge, which has been made explicit mainly in the shape of models. Examples are business processes, business rules, requirements, and cost-benefit calculations. The problem-related knowledge addresses both the problem domain of the target service-based system as well as the problem domain realized

in the pre-existing system. We observed three main types of knowledge representing the problem domain: *requirements*, *quality requirements*, and *problem domain attributes* (Table A.VIII).

Requirements address the functionality and behavior of the domain. Requirements are illustrated using different models, called *problem elements*. The problem elements are codified by two main types of models: structural and behavioral models. As an example, conceptual data models [37, 50], business service blueprints [38], and use cases [28] represent the structural problem decomposition of both pre-existing and target systems. Business processes, workflows, and scenarios represent the behavioral aspect of the problem [49, 51].

Problem domain attributes represents a characteristic or property of the problem domain or migration problem. Examples of the business domain properties include the following: cost of migration [25, 27], feasibility of migration [38, 48], and value of reuse [30, 40].

Quality requirements specify the envisioned quality requirements of the target service-based system as well as those of the pre-existing system. We found very few approaches that address quality requirements in migration. Examples of such quality requirements are as follows: interoperability [52], flexibility [52], sustainability [36], maintainability [36], and configurability [47].

4.4.1.3. Solution-related tacit knowledge. This category includes intangible knowledge of the practitioner, which is exploited to provide solutions at different stages of migration. For instance, in [53], considering patterns of usage execution flows, the architects (out of experience) knows the services that are most reusable. Likewise, [29, 51] exploit tacit knowledge of the architect about the candidate legacy elements for migration. Overall, solution-related tacit knowledge was rarely observed in the primary studies.

4.4.1.4. Problem-related tacit knowledge. This category includes problem domain knowledge about the existing or target system that are not explicitly codified, such as goals, existing assumptions, existing and future concerns, constraints, and existing business rules. A few primary studies rely on problem-related knowledge that remains tacit in stakeholders' minds [25, 27, 28, 49].

4.4.2. Knowledge conversions. Section 4.4.1 addressed the type of knowledge that are used in migration. In this section, we discuss the interactions and conversions between knowledge elements. Each migration approach can be seen as a series of knowledge conversions between *tacit* and *explicit* as well as *solution-related* and *problem-related* knowledge. As an example, the migration approach of Nguyen *et al.* [49] starts with capturing the tacit knowledge of *most beneficial business functions* and representing those business functions in a *business service model* (i.e., conversion of *tacit* to *explicit*). Next, the business service model is transformed to software service model (i.e., conversion of *problem-related* to *solution-related* knowledge).

Section 2.2 introduced 16 categories of knowledge conversions. Out of these 16 conversions, 10 were found in the primary studies. Figure 19 presents those knowledge conversions using solid arrows that are further described in the following. Table A.IX in Appendix details the primary studies covering those categories of knowledge conversions. The knowledge conversions between tacit and explicit are called the following: *socialization* (tacit to tacit), *externalization* (tacit to explicit), *internalization* (explicit to tacit), and *combination* (explicit to explicit).

- *Combination of problem-related knowledge.* This type of conversion is used to explore the problem-space (related to both As-Is and to To-Be states) by combining the problem-related knowledge elements and creating new knowledge elements out of existing ones. For example, Umar and Zordan [27] create cost-benefit calculations of the target system out of the existing ones. Lewis and Smith [25] create service requirements out of goals and critical business processes. While exploration of problem space is crucial for determining the solution for migration, only 23% of the primary studies cover this type of conversion.
- *Externalization of problem-related knowledge.* This conversion relates to making tacit problem-related knowledge explicit. Twenty-nine percent of the primary studies, as a part of their migration approach, address this conversion. Some externalize goals (Umar and Zordan [27]), some business processes [37, 50, 54], and some constraints [36, 52].
- *Combination of problem-related and solution-related knowledge.* This conversion addresses transformation of explicit problem-related knowledge (e.g., business processes) to solution-

related knowledge (e.g., service compositions). This type of conversion was observed frequently in the primary studies.

- *Combination of solution-related knowledge.* This conversion is about transforming a solution-related knowledge to another solution-related knowledge. For instance, within the reverse engineering activity, the existing legacy code or design (i.e., explicit solution-related knowledge) can be used to reconstruct the legacy architecture (e.g., Zhang *et al.* [47]; Liu *et al.* [37]; Canfora *et al.* [40]). Another example of such conversion is when existing architecture is transformed to the target service composition (e.g., Arcelli *et al.* [44]; Kannan and Srivastava [42]; Pahl and Barrett [43]). Not surprisingly, almost all of the primary studies covered this conversion.
- *Externalization of problem-related to solution-related knowledge.* This conversion reflects the transformation of tacit problem-related knowledge to explicit solution-related knowledge. For instance, within the reverse engineering process, tacit problem-related knowledge such as most beneficial business rules can be exploited to extract legacy elements with high reuse potential [30]. Also, this type of conversion occurs in forward engineering process when assumptions about business value of a service are used in candidate service identification [40].
- *Externalization of solution-related knowledge.* This type of knowledge conversion occurs considerably in the SOA migration approaches in practice. As an example, an architect knows the legacy segments that are good candidates for SOA migration and comes up directly with the regarding service design. This conversion was observed in Li *et al.* [29]; Alahmari *et al.* [51].
- *Internalization of problem-related knowledge.* This conversion takes externalized knowledge and makes it into individual tacit knowledge in the form of know-how. We found a few primary studies (i.e., Lavery *et al.* [28]; Lewis and Smith [25]; Umar and Zordan [27]) that embed this conversion in the form of learning the discrepancies and mismatches within existing legacy assets from the explicit, problem-related knowledge (e.g., existing business processes or business rules).
- *Internalization of solution-related to problem-related knowledge.* Explicit solution-related knowledge of the pre-existing legacy system may consequently lead to new ideas, insights, and goals concerning the target SOA environment. Only one study covers this type of conversion [55].
- *Internalization of solution-related knowledge.* Based on experience, a software engineer could arrive directly from an existing solution in the legacy system to a solution in SOA environment. This way, the migration process skips the intermediate conversions between problem-related to solution-related knowledge (within forward engineering), and vice-versa (within reverse engineering). Consequently, the intermediate knowledge is never externalized. This kind of knowledge is thus tacit, because the software engineer ‘just knows’ it fits the problem. Nasr *et al.* [55] and Khadka *et al.* [56] entail this conversion.
- *Combination of solution-related knowledge.* Explicit solution-related knowledge of the pre-existing system such as existing architecture or design models are converted to the problem-related knowledge such as existing business processes. This type of knowledge conversion is associated with business process extraction activity within the reverse engineering [55–58].

5. SOA MIGRATION FRAME OF REFERENCE

The frame of reference for SOA migration we developed depicts the categories identified in this systematic literature review and introduced in Sections 4.2 and 4.4. As shown in Figure 17, this frame of reference structures the categories of (i) the knowledge elements and the conversions among them and (ii) activities covered. The SOA migration frame of reference serves to select existing migration approaches, or even to drive the development of new approaches. In this regard, using this frame of reference one can define the two-view representation of the fitting approach for migration. In such a representation, the knowledge view frames the knowledge elements that are feasible and favorable to be made available while the activity view frames the activities that have to be carried out. Such knowledge and activity views characterize the suitable migration approaches (to be selected or newly defined).

For selecting or defining a migration approach for a certain context, the frame of reference lends itself to a generic method in which first the knowledge view is defined. Next, by providing the link between knowledge and activity views, it helps to arrive from the knowledge view to the activity

Figure 17. Knowledge conversions.

view. What lies in the heart of this link is the *desired knowledge conversions* and the *migration families* able to realize those conversions. Figure 17C shows the mapping of knowledge conversions and the families supporting those conversions. Here follows our reasoning behind this mapping.

The knowledge conversions between problem-related and solution-related knowledge are the key in identifying the required activities for migration. Considering the transformation process, for instance, in case the desired knowledge conversions reside in the solution space (e.g., modules are converted to Web services), then automatic transformations can be a candidate. This makes families supporting this type of transformation (e.g., F1, F4, and F5) suitable candidates for migration. On the other hand, if the transformation embraces the problem to problem conversion (e.g., conversion of legacy business processes to business services), a mapping mechanism (e.g., gap analysis) performed by experts is more suitable. This can be supported only by families covering transformation at top most level (i.e., F3 and F8).

Figure 17C pairs the knowledge conversions and migration families. The schematic forms in this figure represent the activities that realize the conversions. For instance, arrow X (i.e., the generalized form of conversions c, j, e, and h) is paired with activities shown in bold in right hand of Figure 17C. These activities are covered only in Families F2, F5, and F8. It should be noted that the mapping of knowledge conversions and migration families is meant to narrow down the possible migration approaches. In the end, the selection of the approach requires careful consideration of characteristics goals of migration families.

To sum up, the SOA migration frame of reference would facilitate extracting the skeleton of the desired migration approach in the form of the two-view representation. Having the two-view representation at hand one can select and/or define the suitable migration approach that best fits the specific characteristics and needs of a certain case. This, however, should be carried out incrementally. The idea is that practitioners may explore, understand, and define knowledge to be used and the activities to be carried out, incrementally, as in different stages of the project they would focus on a subset of activities and knowledge. This reduces barriers of adoption by making it possible to gain value without needing to make a high investment in defining the complete approach at once. In the next section, we use the frame of reference to select an approach for a real-world migration case.

5.1. Frame of reference application example

This section uses the frame of reference in an example migration case and determines the migration approach that fits the context and goals of that certain case. To provide a realistic example, we took a real-world migration case from earlier work on an industrial SOA migration [8]. In the following, we first introduce the example case and further exemplify the uses of the frame of reference in defining the fitting migration approach for this case.

5.1.1. Example case. For the sake of simplicity, some details of this case are ignored. Our example case is an enterprise that wants to replace a pre-existing system with a new service-based system. Although the ultimate goal is replacing the pre-existing system, yet the enterprise aims at reusing the pre-existing functionality of the legacy system. Consequently, the enterprise targets migration of the legacy system to SOA. In short, there are two main goals for this migration: (g1) to incorporate a set of new goals and requirements and (g2) to achieve flexibility. Moreover, the migration in this enterprise has the following characteristics:

- (c1) The knowledge about the pre-existing system mainly resides in the stakeholders' minds (e.g., maintainer, developer, and architect). As such, the stakeholders know what functionalities (as well as non-functionalities) are available, and where those functionalities are located in the legacy system. Documentations related to architecture description and database schema of the legacy system are still available.
- (c2) The availability of the legacy system is restricted to working hours, as the transactions are executed in batch after working hours.
- (c3) There is a local best practice for understanding the As-Is and To-Be states, that is, gaining understanding through extracting the information architecture (representing the business entities) and the functionality architecture (representing the business functionalities).

5.1.2. Selecting the migration approach of the example case using the frame of reference.

1. **Identifying the knowledge elements.** For defining the migration approach suitable for the example case, first the relevant knowledge elements need to be identified. To this aim, the knowledge elements shown in Figure 17A act as a checklist of relevant types of input knowledge. Using this checklist one could ask himself/herself whether each type of knowledge is available or whether it is desirable and feasible to be further produced. This way the knowledge elements driving the migration are identified. Based on the goals and characteristics explained previously, and by using the knowledge view of the frame of reference, we elicited the knowledge elements that have to be externalized in this migration example. The knowledge elements marked in *white* in Figure 18A exemplify the elements that would be already available in this certain example, whereas, the *gray* ones would indicate the knowledge elements that are currently unavailable.

As noted, to identify the relevant knowledge elements for a certain migration one needs to check the knowledge elements of each of the knowledge categories in Figure 17A (i.e., code-related, design-related, and problem-related). For instance, in this example from the design-related knowledge category in Figure 17A, functional decomposition model is a relevant type of knowledge, which is already available in the form of *architecture descriptions* (see characteristic c1 in the example case). Thus, the *architecture descriptions* are available input knowledge that should be marked white in the knowledge view (Figure 18A). Likewise, persistent data model would be relevant for this example because *database schema* is available, which can facilitate extraction of services of type data. In summary, by acting as a checklist, the knowledge elements of the frame of reference (Figure 17A) aid to identify the input/output knowledge that would drive this example migration.

2. **Identifying the knowledge conversions.** After identifying the available and unavailable knowledge elements, one can decide on how to elicit the unavailable knowledge. If the unavailable knowledge is of type tacit then it might be desirable to either codify it (i.e., externalization) or to make it available (e.g., by keeping track of who knows what) while it still remains tacit in stakeholders' minds (socialization). In addition, if knowledge is

Figure 18. A frame of reference for SOA migration. (A) Knowledge view, (B) activity view, and (C) mapping knowledge conversions to activity view.

unavailable one might decided to create it out of other knowledge (i.e., combination conversion mode). For instance, creating *business services* from *business processes*.

Similar to knowledge elements, the description of knowledge conversions plays the checklist role for identifying knowledge conversions. Consequently, the modes of knowledge conversion (i.e., externalization, combination, internalization, and socialization), covered in the example case are determined.

As an example, for *As-Is and To-Be business processes* (Figure 18A) we need to go from tacit to explicit (i.e., externalization mode). Explicit knowledge here takes the form of business process models. The conversion (b) is covered, because the business processes of both As-Is and To-Be states have to be modeled. In the same vein, *information architecture* and *functionality architecture* have to be externalized too.

To create the output knowledge elements we need to go from explicit to explicit knowledge, that is, to combine knowledge from different sources (i.e., combination mode). For instance, to create *business services*, we need to combine *information architecture* and *functionality architecture* (conversion (a)). Likewise, using the *database schema*, *data flow* and *data model* can be created (i.e., conversion (c)). All in all, knowledge conversions (a) and (c) need to be supported in this migration example (Figure 18B).

3. **Identifying the activity coverage.** As noted, the knowledge conversions pinpoint the generic types of activities that have to be carried out. Considering the knowledge conversions of this example, only family F8 can realize conversions (a), (b), and (c). This suggests migration of type forward engineering with gap analysis being a suitable choice for this example. Being an F8 migration approach implies that *reverse-engineering* process is not covered and understanding about the existing system can be achieved by consulting stakeholders, rather than generating the abstractions from the existing code. Figure 18C illustrates the resulting activity coverage. The *business model transformation* constitutes gap analysis between As-Is and To-Be business processes. Within the *forward-engineering*, process services are analyzed, designed, and implemented. Finally, *legacy element transformation* activity transforms the *legacy functionality* to *service model*.

Taken together, an approach similar to the one presented in [49, 55, 56, 59] appears to be the most fitting one for this certain migration.

6. DISCUSSION

In the following, we discuss the possible threats to validity of our analysis followed by our additional observations.

6.1. Threats to validity

6.1.0.1. Internal validity. Internal validity aims at ensuring that the collected data enables researchers to draw valid conclusions [60]. This validity relates to ‘how’ the research is carried out, and whether the used methods are credible. One of the threats to internal validity of the study is that the review is mainly conducted by two researchers. However, subjective interpretations are mitigated by both following a systematic protocol, checked by senior researchers experienced in software engineering, systematic reviews and SOA, and validating the protocol using a pilot study. Additionally, we explicitly included only publications whose objective is to present a solution for migration. It is possible that a publication proposes also a solution for migration blended with other objectives so that the contribution on migration is not clearly represented. To mitigate this threat, we added some more generalized keywords such as ‘reuse’ in the search terms. To further ensure the unbiased selection of articles, a multistage process was utilized that involved two researchers who documented the reasons for inclusion/exclusion of every step as described in Section 3.3.

Another threat is appraising the quality of published research based on their report on evaluation of the work as journal articles and, in particular, conference papers rarely provide enough detail of the evaluation of their work because of space limitations in journal volumes and conference proceedings. There is therefore a danger that what is being assessed is the quality of reporting rather than the quality of research. To mitigate this threat, we contacted the authors of the primary studies and checked if there exists any publication stating or reporting the application of their approach.

Threat to validity of data extraction process is that the primary studies lack sufficient information for us to be able to document their migration approaches satisfactorily in the data extraction forms. There is therefore a possibility that the extraction process may have resulted in some inaccuracy in data. Nevertheless, we mitigated this threat through consensus meetings. In the data extraction process, each primary study was read by two reviewers. One reviewer acted as the main data extractor, while the other reviewer acted as a checker. Any disagreements were discussed in the data extraction consensus meetings.

Threat to validity of the analysis is in the general applicability of the codes used for characterizing and classifying migration approaches. An assuring factor in this regard is that the start-list of codes is extracted from our SOA-MF published in a service-oriented computing forum, after being peer reviewed by experts in the field [14]. This framework stems from existing theory on reengineering and architectural recovery while it is constantly refined through our coding procedure. This further consolidates its general applicability.

6.1.0.2. External validity. External validity defines to what extent findings from the study can be generalized [60]. Publication bias is a threat to external validity of this study in that the scope of our review is restricted to the scientific domain. The threat here is that very relevant migration approaches originated in industry, if not described in scientific publications, are not covered. Nevertheless, this threat is mitigated by our industrial survey on SOA migration on how SOA migration is performed industrially [8, 61].

6.2. Additional observations

As a co-product of the frame of reference presented so far, our study yields the following complementary findings:

6.2.0.1. Reverse engineering approach to migration. Given the migration families, the number of primary studies in each of those families brings an insight about the current focus of the subject in the field. Families with the main focus on reverse engineering (i.e., F2, F3, and F6) initially attracted a lot of attention. The migration approaches in those families mostly take a reverse engineering approach to define methods, tools, and techniques porting legacy assets to service-oriented ones. The families taking a forward engineering approach (i.e., F5 and F8), however, mostly elicit important knowledge about the legacy assets, and re-engineer them into services in a top-down manner. Interestingly, the families centering on forward engineering are more recent. Accordingly, we conjecture that although academic approaches are still dominated by reverse-engineering perspective, there is an increasing interest in forward-engineering ones. In our opinion, this is in line with the shift in how SOA is perceived: from a technology enabler to a software engineering paradigm [22]. More precisely, in the SOA, hype organizations perceived SOA as a technology enabler that facilitates integrating their existing applications while now SOA is perceived as a paradigm that guides the development of enterprise services from existing assets.

6.2.0.2. As-Is driven migration. Given the two categories of bowl-shaped and arch-shaped approaches, we found that majority of academic approaches are arch-shaped. This implies that understanding As-Is state and modernizing As-Is legacy assets constitutes the common view on migration in the field. This further confirms that in academia the main focus is on modernizing legacy assets, instead of identifying ideal services and reengineering legacy assets to realize those services.

6.2.0.3. Focus on explicit solution-related knowledge. Considering the types of knowledge used and produced, we found that academic approaches mainly focus on explicit solution-related knowledge. This is evident in Figure 19 where most of the identified knowledge elements reside in the explicit solution-related square. This implies that, academic approaches mainly rely on how legacy assets are already implemented or how the services should be designed and implemented, instead of what problems or requirements that the legacy assets already address or To-Be services need to address. In other words, important tacit knowledge is mainly neglected.

6.2.0.4. Focus on externalization of knowledge. Considering the knowledge conversions involving tacit and explicit knowledge, the migration approaches cover mainly explicit to explicit and tacit to explicit knowledge conversions, but not conversions resulting in tacit knowledge. The former conversions capture the idea that knowledge has to be externalized, meaning that the tacit type of knowledge should be converted to the explicit knowledge. This, however, is not the only possible conversion. According to Nonaka and Takeuchi [11], tacit and explicit knowledge should mutually interact. The explicit to tacit conversion, mainly centers on learning or gaining insight from the existing explicit knowledge. Only few approaches cover such conversions (approaches including conversion (g) in Figure 19). Interestingly, knowledge conversions that include learning or gaining insight from existing explicit knowledge are widely used in practice. This highlights the need for further research to support activities that embrace the explicit to tacit knowledge conversion.

6.2.0.5. Lack of focus on quality requirements. One of the main concerns of enterprises carrying out SOA migration should be whether or not the target service-based system can meet the pre-existing or envisioned quality requirements. Migration to SOA can have positive impact on some quality requirements and negative impact on some others. To support the quality requirements, a

Figure 19. Two-view migration approach of the example case. (A) Knowledge view, (B) knowledge conversions, and (C) activity view.

migration approach has to incorporate the identification, analysis, and realization of them. In other words, the quality requirements should drive the migration as well. During the analysis, however, we only found nine studies (see Table A.IX in Appendix) that explicitly address some quality requirements in their migration approach. This shows an important shortcoming because one of the key challenges of migration in practice is how to identify, transform, and develop services that realize the required quality requirements.

6.3. Research challenges on SOA migration

This paper provided an in-depth analysis of how academia perceives SOA migration. While academics mainly take a reverse engineering approach and propose methods, tools, and techniques to port old systems to service-oriented ones, enterprises take a forward-engineering approach, mostly by eliciting important knowledge about the requirements supported by the legacy software and re-engineering them into services that fulfill an ideal SOA [61]. The question is: why this gap between SOA migration approaches by academic and practitioners exists? In our earlier work, we have argued that this is due to the fact that academic approaches do not fit the fundamental problems, goals, strategies, and weaknesses of practice [61]. Having identified the theory-practice gaps, we now turn our attention to the future. Out of the results of case studies [62, 63], industrial surveys [61], and panel with experts [8], we found out that, in practice, there is a need for SOA migration approaches that are simple enough to be pragmatic and have very wide applicability. The best approaches are those that are based on a gradual migration approach. They have to be in-line with the mental models of the practitioner, and eventually offer choices, possibilities, and alternatives for important problems that are typical in

SOA migration. Considering this generic need, we believe the following represents major challenges for industry-relevant SOA migration approaches in the years ahead:

- *Aligning risks, costs, and value with migration approach.* One of the issues that was repeatedly stated by practitioners was the importance of risks and cost management in SOA migration decision making. We found risks and cost management to be in fact one of the main drivers of migration and influential on most decisions. This further confirms a recent interest toward risk-aware, cost-aware and value-aware methods [64, 65] that needs further research.
- *Providing decision-making tools to support selection of migration solutions.* Industry needs tools to support planning and decision making for migration. For this purpose, the practitioners indicated as very beneficial to associate the practices or extensions with typical risks, costs, and pre-requisites. This calls for empirical research studies to identify and isolate practices and associate them with important decision criteria such as risks, costs, and pre-requisites.
- *To-Be-driven migration approaches.* Migration in industrial approaches is mainly driven by To-Be state, thus bowl-shaped [61]. Inadequate support for To-Be driven approaches in academia highlights promising opportunities for research. In particular, future research should focus on how to systematically elicit and capture the migration drivers and how to shape the migration process using those drivers.
- *Legacy understanding without reverse-engineering.* Often, the industrial migration approaches do not use reverse engineering techniques to understand the legacy systems [61]. The required knowledge is elicited from the stakeholders who own the knowledge. Elicitation of the knowledge about the legacy system, however, is crucial for a successful migration. In this regard, research can benefit practice by providing methods, techniques, or guidelines that facilitate elicitation of migration-relevant knowledge from different sources.

7. CONCLUSION

We conducted a systematic literature review, to obtain the categories of SOA migration approaches in the field. The analysis of 75 primary studies resulted in (i) 8 categories of migration approaches with respect to activities carried out, (ii) 21 categories of knowledge elements that shape and drive the migration, and (iii) 10 categories of knowledge conversions used to create knowledge. By consolidating and relating those categories, we defined the *SOA migration frame of reference*. The frame of reference can help practitioners to better understand the fundamentals of existing approaches as well as provide a common ground for developing new approaches better suited for their needs.

Our evidence and discussion support two key conclusions. First, the dominant view on SOA migration relates to using SOA as a technology enabler that facilitates integrating existing applications, rather than a paradigm that guides the development of business and software services from existing assets. We conclude that additional research should be carried out to support the emerging perspective of service orientation as a software engineering paradigm. Our results could be used as a starting point for such research.

Second, from a knowledge management perspective, this article has found typical types of knowledge that shape and drive the migration. We further found that academic approaches mainly focus on knowledge conversions that systematize and capture information mostly in the form of models and make this knowledge available by codifying it (i.e., *codification* knowledge management strategy). In industrial practice, however, knowledge is also transferred by person-to-person communication (i.e., *personalization* knowledge management strategy). In such situations, it is essential for instance to support the knowledge transfer by having a centralized store of information about knowledge sources, like yellow pages of who knows what. Therefore, we conclude that further research is needed to bring the advancements of knowledge management discipline, such as personalization strategies, to SOA migration. In this regard, this paper serves as a first step in this new research area.

APPENDIX

Table A.I. Data extraction form.

Field	Description
1. Study identifier	[Provide a unique ID]
2. Title	[Title of the primary study]
3. Reviewer name	[Name of the reviewer conducting the data extraction]
4. Data source	[The name of the database where the primary study was found (i.e., IEEE Xplore, ACM Digital Library, ISI Web of Knowledge, SpringerLink, ScienceDirect, and Wiley InterScience Journal Finder)]
5. Type of article	[The type of article (workshop, conference, book chapter, or journal)]
6. Summary of migration approach	[Enter the summary of the migration approach]
7. Goal of migration	[Enter the goal of performing the migration]
8. Activities with inputs/outputs	[List the set of activities carried out in migration together with their inputs and outputs]
9. Sequencing of activities	[Provide the sequence in which each of the above-mentioned activities are carried out]
10. Usage of approach	[Is the approach applied in a case study? (Yes/No) If yes: describe whether the case study is a real-world case study or an example one.]

Table A.II. Codes start-list.

Category	Code	Name	Description
<i>Activity</i>	ACT-CoAn	Code analysis	The activity that analyzes the legacy code and extracts its comprising elements, called legacy elements (e.g., components, modules, or segments of code).
	ACT-ArchRec	Architectural recovery	The activity that recovers the inter-dependencies and compositions of legacy elements.
	ACT-BMRec	Business model recovery	The activity that extracts the business domain models (e.g., business process models, use case models, business rules) from legacy elements.
	ACT-DETr	Design element transformation	The activity that transforms the legacy elements to services.
	ACT-ComTr	Composition transformation	The activity that transforms the legacy architecture to service compositions.
	ACT-MPTr	Business model transformation	The activity that transforms the legacy business models to new ones to be used in the service-based system.
	ACT-SrvAn	Service analysis	The activity that identifies candidate services from business domain models.
	ACT-SrvDsg	Service design	The activity that focuses on design of software services and service compositions.
<i>Knowledge</i>	ACT-SrvImpl	Service implementation	The activity that realizes the implementation of software services.
	KN-Sol	Solution-related knowledge	The knowledge that addresses the solution to a specific problem domain.
	KN-Prob	Problem-related knowledge	The knowledge presenting the analysis of the problem, the problem decomposition, and the world in which it is located (problem space).
	KN-Tacit	Tacit knowledge	The knowledge that remains implicit in stakeholder mind.
	KN-Explicit	Explicit knowledge	The knowledge that is codified in documents and artifacts.

Table A.III. Overview of primary studies.

Study	Year	Applicability
S1 [66]	2008	Low
S2 [44]	2008	Low

(Continues)

Table A.III. (Continued)

Study	Year	Applicability
S3 [67]	2008	High
S4 [34]	2003	Low
S5 [40, 68, 69]	2006 and 2008	High
S6 [31]	2009	High
S7 [70, 71]	2007	Low
S8 [72]	2004	Medium
S9 [39]	2005	Medium
S10 [38]	2009	Medium
S11 [50]	2007	High
S12 [36]	2008	High
S13 [41]	2009	High
S14 [45]	2009	High
S15 [73]	2005	Medium
S16 [52]	2008	High
S17 [74]	2008	Medium
S18 [75]	2008	Medium
S19 [76]	2009	Medium
S20 [26]	2007	Low
S21 [77]	2008	High
S22 [42]	2008	High
S23 [28]	2004	High
S24 [78]	2004	Low
S25 [25, 79, 80]	2005, 2006, and 2007	High
S26 [29]	2007	Medium
S27 [81]	2008	High
S28 [48]	2009	Low
S29 [82]	2004	Low
S30 [37]	2008	Low
S31 [83]	2008	High
S32 [84]	2008	High
S33 [49, 85]	2009	Medium
S34 [32]	2005	High
S35 [43]	2006	High
S36 [46]	2008	High
S37 [30]	2006	High
S38 [27]	2009	High
S39 [23]	2004	Medium
S40 [86]	2005	Medium
S41 [47, 87]	2004 and 2006	Medium
S42 [88]	2008	High
S43 [89]	2000	High
S44 [90]	2007	High
S45 [51]	2010	High
S46 [91]	2010	High
S47 [92]	2010	High
S48 [24]	2010	High
S49 [93]	2009	Medium
S50 [94]	2009	Low
S51 [95]	2009	High
S52 [96]	2010	Low
S53 [97]	2009	Medium
S54 [98]	2010	High
S55 [99]	2010	High
S56 [100]	2009	High
S57 [54]	2010	High
S58 [101]	2010	Low
S59 [102]	2010	High
S60 [53, 103]	2010	High
S61 [104]	2010	Low

(Continues)

Table A.III. (Continued)

Study	Year	Applicability
S62 [105]	2011	High
S63 [106, 107]	2011 and 2012	High
S64 [108]	2011	High
S65 [109]	2011	Low
S66 [57, 110]	2011 and 2013	High
S67 [58]	2012	High
S68 [111]	2012	High
S69 [55]	2013	High
S70 [59]	2013	High
S71 [112]	2013	High
S72 [33]	2013	High
S73 [56, 113]	2013	High
S74 [35]	2013	High
S75 [114]	2013	Medium

Table A.IV. Overview of SOA migration families.

Family	Member	Number	Percent
F1	Bodhuin and Tortorella [34], Liu <i>et al.</i> [82], Varga <i>et al.</i> [23], Lee <i>et al.</i> [78], Zhang <i>et al.</i> [88], Huang [76], Iocola [90], Balis <i>et al.</i> [67], Chenghao <i>et al.</i> [24], Koschel <i>et al.</i> [95], Delsing <i>et al.</i> [106], and Hsieh <i>et al.</i> [111]	12	17%
F2	O'Brien <i>et al.</i> [32], Chen <i>et al.</i> [38], Sindhgatta and Ponnalagu [46], Li and Tahvildari [81], Ilk <i>et al.</i> [77], Alahmari <i>et al.</i> [51], Bao <i>et al.</i> [91], Chung <i>et al.</i> [93], Nakamura <i>et al.</i> [100], Zhang <i>et al.</i> [53], Zhou <i>et al.</i> [104], and Sneed <i>et al.</i> [58]	12	17%
F3	Lavery <i>et al.</i> [28], Lewis and Smith [25], Hutchinson <i>et al.</i> [26], Umar and Zordan [27], and Chung <i>et al.</i> [50]	5	7%
F4	Ahn <i>et al.</i> [66], del Castillo <i>et al.</i> [31], El-Ramly <i>et al.</i> [41], Gonzalez <i>et al.</i> [45], Guo <i>et al.</i> [73], Haidar and Abdallah [52], Huang <i>et al.</i> [75], Li <i>et al.</i> [29, 48], Sneed [30], Zou and Kontogiannis [89], Lucia <i>et al.</i> [83], Matos [84], Bissyande <i>et al.</i> [92], Djelloul <i>et al.</i> [94], Almonaies <i>et al.</i> [105], Yu <i>et al.</i> [109], Baghdadi and Al-Bulushi [112], Sneed <i>et al.</i> [35], Xu <i>et al.</i> [86], and Bissyande <i>et al.</i> [92]	21	28%
F5	Cetin <i>et al.</i> [70], Chen <i>et al.</i> [39], Koufi <i>et al.</i> [96], Mulcahy <i>et al.</i> [99], Nasr <i>et al.</i> [54], Fuhr <i>et al.</i> [57], Rodriguez <i>et al.</i> [33], and Sosa <i>et al.</i> [114]	8	10%
F6	Canfora <i>et al.</i> [40], Cuadrado <i>et al.</i> [36], Heckel <i>et al.</i> [74], Liu <i>et al.</i> [37], Zhang <i>et al.</i> [47], Marchetto and Ricca [97], Millham [98], Tibermacine and Kerdoudi [101], Wang <i>et al.</i> [102], Delsing <i>et al.</i> [106], and Sosa <i>et al.</i> [114]	10	14%
F7	Arcelli <i>et al.</i> [44], Kannan and Srivastava [42], and Pahl and Barrett [43]	3	4%
F8	Nguyen <i>et al.</i> [49], Nasr <i>et al.</i> [55], and Khadka <i>et al.</i> [56]	4	5%

Table A.V. Overview of sequencing of activities in primary studies.

Category	Member	Number	Percent
Arch-shaped approaches	Bodhuin and Tortorella [34], Huang [76], Iocola [90], Lee <i>et al.</i> [78], Liu <i>et al.</i> [82], Varga <i>et al.</i> [23], Zhang <i>et al.</i> [88], Li and Tahvildari [81], O'Brien <i>et al.</i> [32], Sneed [30], Ahn <i>et al.</i> [66], del Castillo <i>et al.</i> [31], El-Ramly <i>et al.</i> [41], Gonzalez <i>et al.</i> [45], Guo <i>et al.</i> [73], Haidar and Abdallah [52], Huang <i>et al.</i> [75], Li <i>et al.</i> [29, 48], Zou and Kontogiannis [89], Cuadrado <i>et al.</i> [36], Liu <i>et al.</i> [37], Arcelli <i>et al.</i> [44], Pahl and Barrett [43], Lucia <i>et al.</i> [83], Matos [84], Balis <i>et al.</i> [67], Alahmari <i>et al.</i> [51], Bissyande <i>et al.</i> [92], Chenghao <i>et al.</i> [24], Chung <i>et al.</i> [93], Djelloul <i>et al.</i> [94], Koschel	48	65%

(Continues)

Table A.V. (Continued)

Category	Member	Number	Percent
Bowl-shaped approaches	<i>et al.</i> [95], Marchetto and Ricca [97], Millham [98], Nakamura <i>et al.</i> [100], Tibermacine and Kerdoudi [101], Zhang <i>et al.</i> [53], Sneed <i>et al.</i> [58], Hsieh <i>et al.</i> [111], Baghdadi and Al-Bulushi [112], Sneed <i>et al.</i> [35], Sosa <i>et al.</i> [114], Almonaies <i>et al.</i> [105], Delsing <i>et al.</i> [106], Sonntag <i>et al.</i> [108], and Yu <i>et al.</i> [109]	24	35%
	Chen <i>et al.</i> [38], Ilk <i>et al.</i> [77], Sindhgatta and Ponnalagu [46], Cetin <i>et al.</i> [70], Lavery <i>et al.</i> [28], Chung <i>et al.</i> [50], Chen <i>et al.</i> [39], Canfora <i>et al.</i> [40], Zhang <i>et al.</i> [47], Heckel <i>et al.</i> [74], Kannan and Srivastava [42], Nguyen <i>et al.</i> [49], Bao <i>et al.</i> [91], Koufi <i>et al.</i> [96], Mulcahy <i>et al.</i> [99], Nasr <i>et al.</i> [54], Wang <i>et al.</i> [102], Zhang <i>et al.</i> [53], Zhou <i>et al.</i> [104], Nasr <i>et al.</i> [55], Khadka <i>et al.</i> [56], Rodriguez <i>et al.</i> [33], Khadka <i>et al.</i> [56], and Fuhr <i>et al.</i> [57]		

Table A.VI. Code-related knowledge elements.

Knowledge	Primary studies	Number	Percent
Code document	O'Brien <i>et al.</i> [32] and Rodriguez <i>et al.</i> [33]	Code Itself	3%
Code grammar	Bodhuin and Tortorella [34], Li and Tahvildar (2008), Huang <i>et al.</i> [75], Zhang <i>et al.</i> [47], Chenghao <i>et al.</i> [24], Koschel <i>et al.</i> [95], Tibermacine and Kerdoudi [101], Almonaies <i>et al.</i> [105], Delsing <i>et al.</i> [106], Sonntag <i>et al.</i> [108], Baghdadi and Al-Bulushi [112], and Sneed <i>et al.</i> [35]	Code Itself	16%
Code model	Lee <i>et al.</i> [78], O'Brien <i>et al.</i> [32], Sindhgatta and Ponnalagu [46], Huang <i>et al.</i> [75], del Castillo <i>et al.</i> [31], Chen <i>et al.</i> [39], Cuadrado <i>et al.</i> [36], and Liu <i>et al.</i> [37]	Code Itself	15%
Code itself	Bodhuin and Tortorella [34], Liu <i>et al.</i> [82], Varga <i>et al.</i> [23], Lee <i>et al.</i> [78], Zhang <i>et al.</i> [88], Huang [76], Iocola [90], O'Brien <i>et al.</i> [32], Chen <i>et al.</i> [38], Sindhgatta and Ponnalagu [46], Li and Tahvildari [81], Ilk <i>et al.</i> [77], del Castillo <i>et al.</i> [31], Chen <i>et al.</i> [39], Cuadrado <i>et al.</i> [36], Liu <i>et al.</i> [37], Ahn <i>et al.</i> [66], El-Ramly <i>et al.</i> [41], Gonzalez <i>et al.</i> [45], Guo <i>et al.</i> [73], Haidar and Abdallah [52], Huang <i>et al.</i> [75], Li <i>et al.</i> [29], Sneed [30], Zou and Kontogiannis [89], Lucia <i>et al.</i> [83], Matos [84], Li <i>et al.</i> [48], Bissyande <i>et al.</i> [92], Djelloul <i>et al.</i> [94], Almonaies <i>et al.</i> [105], Delsing <i>et al.</i> [106], Sonntag <i>et al.</i> [108], Yu <i>et al.</i> [109], Fuhr <i>et al.</i> [57], Sneed <i>et al.</i> [58], Hsieh <i>et al.</i> [111], Nasr <i>et al.</i> [55], Khadka <i>et al.</i> [56], Baghdadi and Al-Bulushi [112], Rodriguez <i>et al.</i> [33], Khadka <i>et al.</i> [56], Sneed <i>et al.</i> [35], and Sosa <i>et al.</i> [114]	Code Itself	59%

Table A.VII. Design-related knowledge elements.

Category	Subcategory	Members	Number	Percent
Design element	Functional decomposition model	Chen <i>et al.</i> [38], Sindhgatta and Ponnalagu [46], Huang <i>et al.</i> [75], Zou and Kontogiannis [89], Lewis and Smith [25], Chen <i>et al.</i> [39], O'Brien <i>et al.</i> [32], Cuadrado <i>et al.</i> [36], Zhang <i>et al.</i> [47], Canfora <i>et al.</i> [40], Kannan and Srivastava [42], Pahl and Barrett [43], Alahmari <i>et al.</i> [51], Koufi <i>et al.</i> [96], Marchetto and Ricca [97], Millham [98], Mulcahy <i>et al.</i> [99], Nasr <i>et al.</i> [54], Wang <i>et al.</i> [102], Sneed <i>et al.</i> [58], Khadka <i>et al.</i> [56], and Sosa <i>et al.</i> [114]	22	29%

(Continues)

Table A.VII. (Continued)

Category	Subcategory	Members	Number	Percent
	Structural model	Li and Tahvildari [81], Guo <i>et al.</i> [73], del Castillo <i>et al.</i> [31], Canfora <i>et al.</i> [40], Zhang <i>et al.</i> [47], Alahmari <i>et al.</i> [51], Bao <i>et al.</i> [91], Bissyande <i>et al.</i> [92], Chung <i>et al.</i> [93], Koufi <i>et al.</i> [96], Nakamura <i>et al.</i> [100], Nasr <i>et al.</i> [54], Tibermacine and Kerdoudi [101], Wang <i>et al.</i> [102], Zhang <i>et al.</i> [53], Zhou <i>et al.</i> [104], Sonntag <i>et al.</i> [108], Hsieh <i>et al.</i> [111], Khadka <i>et al.</i> [56], Sneed <i>et al.</i> [35], and Sosa <i>et al.</i> [114]	20	27%
	Persistent data model	Li <i>et al.</i> [29], del Castillo <i>et al.</i> [31], Djelloul <i>et al.</i> [94], Hsieh <i>et al.</i> [111], Khadka <i>et al.</i> [56], and Sneed <i>et al.</i> [35]	6	8%
	Interaction model	El-Ramly <i>et al.</i> [41], Canfora <i>et al.</i> [40], Cuadrado <i>et al.</i> [36], Alahmari <i>et al.</i> [51], Bao <i>et al.</i> [91], Chung <i>et al.</i> [93], Mulcahy <i>et al.</i> [99], Nakamura <i>et al.</i> [100], and Nasr <i>et al.</i> [54]	9	12%
	Pattern	Kannan and Srivastava [42], Pahl and Barrett [43], del Castillo <i>et al.</i> [31], Arcelli <i>et al.</i> [44], Chen <i>et al.</i> [38], O'Brien <i>et al.</i> [32], Gonzalez <i>et al.</i> [45], Chung <i>et al.</i> [93], Zhang <i>et al.</i> [53], and Sosa <i>et al.</i> [114]	10	13%
	Coupling	Chen <i>et al.</i> [38], O'Brien <i>et al.</i> [32], Sindhgatta and Ponnalagu [46], Cuadrado <i>et al.</i> [36], Zhang <i>et al.</i> [47], Marchetto and Ricca [97], Nakamura <i>et al.</i> [100], Yu <i>et al.</i> [109], Fuhr <i>et al.</i> [57], and Nasr <i>et al.</i> [55]	10	13%
	Reusability	Chen <i>et al.</i> [38], O'Brien <i>et al.</i> [32], Sneed [30], Kannan and Srivastava [42], Alahmari <i>et al.</i> [51], Fuhr <i>et al.</i> [57], and Nasr <i>et al.</i> [55]	7	9%
	Cohesion	Canfora <i>et al.</i> [40], Nguyen <i>et al.</i> [49], Nakamura <i>et al.</i> [100], and Nasr <i>et al.</i> [55]	4	5%
	Statelessness	Chen <i>et al.</i> [38]	1	1%
	Modularization	Li and Tahvildari [81], Zhang <i>et al.</i> [47], Yu <i>et al.</i> [109], Fuhr <i>et al.</i> [57], Sneed <i>et al.</i> [58], and Sosa <i>et al.</i> [114]	6	8%
	Composability	Kannan and Srivastava [42]	1	1%

Table A.VIII. Problem-related knowledge elements.

Category	Subcategory	Members	Number	Percent
Requirements	Structural model	Chung <i>et al.</i> [50], Liu <i>et al.</i> [37], Zhang <i>et al.</i> [47], Chen <i>et al.</i> [38], Lavery <i>et al.</i> [28], Bao <i>et al.</i> [91], Chung <i>et al.</i> [93], Djelloul <i>et al.</i> [94], Koufi <i>et al.</i> [96], Marchetto and Ricca [97], Mulcahy <i>et al.</i> [99], Nasr <i>et al.</i> [54], Wang <i>et al.</i> [102], Zhou <i>et al.</i> [104], Zhang <i>et al.</i> [53], Delsing <i>et al.</i> [106], Fuhr <i>et al.</i> [57], Sneed <i>et al.</i> [58], Nasr <i>et al.</i> [55], Khadka <i>et al.</i> [56], Rodriguez <i>et al.</i> [33], and Sosa <i>et al.</i> [114]	21	28%
	Behavioral model	Lavery <i>et al.</i> [28], Kannan and Srivastava [42], Nguyen <i>et al.</i> [49], Alahmari <i>et al.</i> [51], Koufi <i>et al.</i> [96], Fuhr <i>et al.</i> [57], Nasr <i>et al.</i> [55], Khadka <i>et al.</i> [56], and Sosa <i>et al.</i> [114]	8	11%
Problem domain attribute		Lewis and Smith [25], Umar and Zordan [27], Chen <i>et al.</i> [38], Li <i>et al.</i> [48], Sneed [30], Canfora <i>et al.</i> [40], Zhang <i>et al.</i> [53], Fuhr <i>et al.</i> [57], Nasr <i>et al.</i> [55], and Khadka <i>et al.</i> [56]	10	13%
Quality requirements			9	12%

(Continues)

Table A.VIII. (Continued)

Category	Subcategory	Members	Number	Percent
		Haidar and Abdallah [52], Cuadrado <i>et al.</i> [36], Zhang <i>et al.</i> [47], Alahmari <i>et al.</i> [51], Yu <i>et al.</i> [109], Fuhr <i>et al.</i> [57], Nasr <i>et al.</i> [55], Khadka <i>et al.</i> [56], and Rodriguez <i>et al.</i> [33]		

Table A.IX. Knowledge conversions.

Conversion type	Members	Number	Percent
a	Lavery <i>et al.</i> [28], Lewis and Smith [25], Hutchinson <i>et al.</i> [26], Umar and Zordan [27], Chung <i>et al.</i> [50], Alahmari <i>et al.</i> [51], Bao <i>et al.</i> [91], Koufi <i>et al.</i> [96], Millham [98], Nasr <i>et al.</i> [54], Zhang <i>et al.</i> [53], Fuhr <i>et al.</i> [57], Nasr <i>et al.</i> [55], Khadka <i>et al.</i> [56], Rodriguez <i>et al.</i> [33], Khadka <i>et al.</i> [56], and Sosa <i>et al.</i> [114]	17	23%
b	Lewis and Smith [25], Umar and Zordan [27], Li <i>et al.</i> [48], Sneed [30], Canfora <i>et al.</i> [40], Chung <i>et al.</i> [50], Liu <i>et al.</i> [37], Zhang <i>et al.</i> [47], Chen <i>et al.</i> [38], Lavery <i>et al.</i> [28], Kannan and Srivastava [42], Nguyen <i>et al.</i> [49], Haidar and Abdallah [52], Cuadrado <i>et al.</i> [36], Alahmari <i>et al.</i> [51], Nasr <i>et al.</i> [54], Zhang <i>et al.</i> [53], Fuhr <i>et al.</i> [57], Nasr <i>et al.</i> [55], Khadka <i>et al.</i> [56], and Sosa <i>et al.</i> [114]	22	29%
c	Chen <i>et al.</i> [38], Ilk <i>et al.</i> [77], Sindhgatta and Ponnalagu [46], Cetin <i>et al.</i> [70], Chen <i>et al.</i> [39], Zhang <i>et al.</i> [47], Heckel <i>et al.</i> [74], Kannan and Srivastava [42], Nguyen <i>et al.</i> [49], Chung <i>et al.</i> [93], Koufi <i>et al.</i> [96], Millham [98], Mulcahy <i>et al.</i> [99], Wang <i>et al.</i> [102], Zhou <i>et al.</i> [104], Almonaies <i>et al.</i> [105], Yu <i>et al.</i> [109], Fuhr <i>et al.</i> [57], Hsieh <i>et al.</i> [111], Nasr <i>et al.</i> [55], Khadka <i>et al.</i> [56], Baghdadi and Al-Bulushi [112], Rodriguez <i>et al.</i> [33], Khadka <i>et al.</i> [56], Sneed <i>et al.</i> [35], and Sosa <i>et al.</i> [114]	26	35%
d	Zhang <i>et al.</i> [47], Liu <i>et al.</i> [37], Canfora <i>et al.</i> [40], Arcelli <i>et al.</i> [44], Kannan and Srivastava [42], Pahl and Barrett [43], Bao <i>et al.</i> [91], Bissyande <i>et al.</i> [92], Chenghao <i>et al.</i> [24], Chung <i>et al.</i> [93], Djelloul <i>et al.</i> [94], Koschel <i>et al.</i> [95], Koufi <i>et al.</i> [96], Millham [98], Marchetto and Ricca [97], Mulcahy <i>et al.</i> [99], Nakamura <i>et al.</i> [100], Nasr <i>et al.</i> [54], Tibermacine and Kerdoudi [101], Wang <i>et al.</i> [102], Zhang <i>et al.</i> [53], Zhou <i>et al.</i> [104], Almonaies <i>et al.</i> [105], Delsing <i>et al.</i> [106], Sonntag <i>et al.</i> [108], Yu <i>et al.</i> [109], Fuhr <i>et al.</i> [57], Sneed <i>et al.</i> [58], Hsieh <i>et al.</i> [111], Nasr <i>et al.</i> [55], Khadka <i>et al.</i> [56], Baghdadi and Al-Bulushi [112], Rodriguez <i>et al.</i> [33], Khadka <i>et al.</i> [56], Sneed <i>et al.</i> [35], and Sosa <i>et al.</i> [114]	71	95%
e	Sneed [30], Canfora <i>et al.</i> [40], Zhang <i>et al.</i> [53], and Nasr <i>et al.</i> [55]	3	4%
f	Li <i>et al.</i> [29], Alahmari <i>et al.</i> [51], Fuhr <i>et al.</i> [57], Nasr <i>et al.</i> [55], Khadka <i>et al.</i> [56], Rodriguez <i>et al.</i> [33], and Khadka <i>et al.</i> [56]	7	9%
g	Lavery <i>et al.</i> [28], Lewis and Smith [25], Umar and Zordan [27], Nasr <i>et al.</i> [55], and Khadka <i>et al.</i> [56]	5	7%
h	Nasr <i>et al.</i> [55]	1	1%
i	Nasr <i>et al.</i> [55] and Khadka <i>et al.</i> [56]	2	3%
j	Fuhr <i>et al.</i> [57], Sneed <i>et al.</i> [58], Nasr <i>et al.</i> [55], and Khadka <i>et al.</i> [56]	4	5%

REFERENCES

1. Papazoglou MP, Traverso P, Dustdar S, Leymann F. Service-oriented computing: a research roadmap. *International Journal of Cooperative Information Systems* 2008; **17**(2):223–255.
2. Lago P, Razavian M. A pragmatic approach for analysis and design of service inventories. In: Jmaiel, M., Pallis, G. (Eds.), *ICSOC 2011 Workshop Proceedings*. Vol. 7721. Springer-Verlag, heidelberg, 2012; 44–53.
3. Almonaies A, Cordy JR, Dean TR. Legacy system evolution towards service-oriented architecture. In: *International Workshop on SOA Migration and Evolution*. 2010; 53–62.

4. Khadka R, Saeidi A, Idu A, Hage J, Jansen S. Legacy to SOA evolution: a systematic literature review. *Migrating to SOA and Cloud Environments: Challenges in Service Oriented Architecture and Cloud Computing Environments*, Ionita AD, Lewis G, Litoiu M (eds.). IGI Global.
5. Razavian M, Lago P. A frame of reference for SOA migration. In *Towards a Service-Based Internet, Third European Conference, ServiceWave*. Berlin / Heidelberg: Springer, 2010; 150–162.
6. Tilley SR, Smith D. Perspectives on legacy system reengineering. Tech. rep., Reengineering Center Software Engineering Institute Carnegie Mellon University, 1995.
7. Kruchten P. *Documentation of Software Architecture from a Knowledge Management Perspective Design Representation*. Springer: Berlin Heidelberg, 2009; 39–57.
8. Razavian M, Lago P. A lean and mean strategy for migration to services. In: *WICSA/ECSA 2012 Proceedings Companion Volume*. ACM, 2012; 61–68.
9. Bisbal J, Lawless D, Wu B, Grimson J. Legacy information systems: issues and directions. *IEEE Software* 1999; **16**(5):103–111.
10. Chikofsky EJ, Cross JHC II. Reverse engineering and design recovery: a taxonomy. *IEEE Software* 1990; **7**(1):13–17.
11. Nonaka I, Takeuchi H. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press: New York, 1995.
12. Jackson M. *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison-Wesley Longman Publishing Co., Inc: Boston, MA, USA, 2001.
13. Nuseibeh B. Weaving together requirements and architectures. *Computer* 2001; **34**(3):115–119.
14. Razavian M, Lago P. Towards a conceptual framework for legacy to SOA migration. In: *Fifth International Workshop on Engineering Service-Oriented Applications (WESOA'09)*. 2010; pp. 445–455.
15. Kazman R, Woods SG, Carrière SJ. Requirements for integrating software architecture and reengineering models: CORUM II. In: *Working Conference on Reverse Engineering*. IEEE, 1998; 154–163.
16. Kitchenham B. Guidelines for performing systematic literature reviews in software engineering, 2007.
17. Dyba T, Dingsøyr T, Hanssen GK. Applying systematic reviews to diverse study types: an experience report. In: *1st Int. Symposium on Empirical Software Engineering and Measurement. ESEM 2007*. IEEE, 2007; 225–234.
18. Gu Q, Lago P. Exploring service-oriented system engineering challenges: a systematic literature review. *Service Oriented Computing and Applications* 2009; **3**(3):171–188.
19. Alver M, Batada N, Baylac M, Brix K, Gardey G, D'haese C, Nagel R, Oezbeck C, Reitmayr E, Rudert A, *et al.* JabRef reference manager, 2003.
20. Miles MB, Huberman M. *Qualitative Data Analysis: An Expanded Sourcebook* (2nd edn). Sage Publications: Thousand Oaks, California, USA, 1994.
21. Lincoln YS, Guba EG. *Naturalistic Inquiry*, vol. **75**. Sage Publications, 1985.
22. Scholler DGR. Hype cycle for application architecture. Tech. Rep. G00200962, Gartner, Inc, 2012.
23. Varga LZ, Hajnal Á, Werner Z. An agent based approach for migrating Web services to semantic Web services. *Artificial Intelligence: Methodology, Systems, and Applications*, 2004; 371–380.
24. Chenghao G, Min W, Xiaoming Z. A wrapping approach and tool for migrating legacy components to Web services. In: *Networking and Distributed Computing (ICNDC)*, 2010 First International Conference on. 2010; 94–98.
25. Lewis G, Smith DB. Developing realistic approaches for the migration of legacy components to service-oriented architecture environments. *Trends in Enterprise Application Architecture*. Springer-Verlag: Berlin, Germany, 2007; 226–240.
26. Hutchinson J, Kotonya G, Walkerdine J, Sawyer P, Dobson G, Onditi V. Evolving existing systems to service-oriented architectures: perspective and challenges. In: *Web Services, ICWS 2007*. IEEE International Conference on. 2007; 896–903.
27. Umar A, Zordan A. Reengineering for service oriented architectures: a strategic decision model for integration versus migration. *Journal of Systems and Software* 2009; **82**(3):448–462.
28. Lavery J, Boldyreff C, Ling B, Allison C. Modelling the evolution of legacy systems to Web-based systems. *Journal of Software Maintenance and Evolution: Research and Practice* 2004; **16**(1-2):5–30. DOI:10.1002/smr.282.
29. Li S-H, Yen DC, Chang C-C. Migrating legacy information systems to Web services architecture. *Journal of Database Management* 2007; **18**(4):1–25.
30. Sneed HM. Integrating legacy software into a service oriented architecture. In: *CSMR'06: Proceedings of the Conference on Software Maintenance and Reengineering*. IEEE, 2006; 3–14.
31. del Castillo RP, García-Rodríguez I, Caballero I. PRECISO: a reengineering process and a tool for database modernisation through Web services. In: *SAC'09: ACM symposium on Applied Computing*. 2009; 2126–2133.
32. O'Brien L, Smith D, Lewis G. Supporting migration to services using software architecture reconstruction. *Software Technology and Engineering Practice* 2005; **17**:81–91.
33. Rodríguez JM, Crasso M, Mateos C, Zunino A, Campo M. Bottom-up and top-down Cobol system migration to web services. *Internet Computing, IEEE* 2013; **17**(2):44–51.
34. Bodhuin T, Tortorella M. Using grid technologies for Web-enabling legacy systems. In: *Workshop on Software Technology and Engineering Practice*. IEEE, 2003; 186–195.
35. Sneed HM, Verhoef C, Sneed SH. Reusing existing object-oriented code as Web services in a SOA. In: *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*, 2013 IEEE 7th International Symposium on. IEEE, 2013; 31–39.

36. Cuadrado F, Garcia B, Dueas JC, Parada HA. A case study on software evolution towards service-oriented architecture. In: *Advanced Information Networking and Applications*, AINAW 08. 2008; 1399–1404.
37. Liu Y, Wang Q, Zhuang M, Zhu Y. Reengineering legacy systems with RESTful Web service. *Computer Software and Applications COMPSAC '08*. 32nd Annual IEEE International, IEEE: Turku, 2008; 785–790.
38. Chen F, Zhang Z, Li J, Kang J, Yang H. Service identification via ontology mapping. *Computer Software and Applications Conference*. COMPSAC'09. 33rd Annual IEEE International vol. 1, 2009; pp. 486–491.
39. Chen F, Li S, Yang H, Wang C-H, Cheng-Chung Chu W. Feature analysis for service-oriented reengineering. In: *Software Engineering Conference, APSEC'05*. 12th Asia-Pacific. 2005; 8.
40. Canfora G, Fasolino AR, Frattolillo G, Tramontana P. A wrapping approach for migrating legacy system interactive functionalities to service oriented architectures. *Journal of Systems and Software* 2008; **81**(4):463–480.
41. El-Ramly M, Stroulia E, Samir H. “Legacy systems interaction reengineering”. *Human-Centered Software Engineering*, Springer: London, 2009; 316–333.
42. Kannan K, Srivastava B. Promoting reuse via extraction of domain concepts and service abstractions from design diagrams. In: *Services Computing*, 2008; 265–272.
43. Pahl C, Barrett R. Layered patterns in modelling and transformation of service-based software architectures. In: *Software Architecture, Third European Workshop*. 2006; 144–158.
44. Arcelli F, Tosi C, Zanoni M. Can design pattern detection be useful for legacy system migration towards SOA? In: *workshop on Systems development in SOA environments*. 2008; 63–68.
45. Gonzalez MAC, Penalvo FJG, Guerrero MJC, Forment MA. Adapting LMS architecture to the SOA: an architectural approach. In: *Internet and Web Applications and Services, ICIW'09*. 2009; 322–327.
46. Sindhgatta R, Ponnalagu K. Locating components realizing services in existing systems. In: *Services Computing*. 2008; 127–134.
47. Zhang Z, Yang H, Chu WC. Extracting reusable object-oriented legacy code segments with combined formal concept analysis and slicing techniques for service integration. In: *Quality Software*. 2006; 385–392.
48. Li Z, Anming X, Naiyue Z, Jianbin H, Zhong C. A SOA modernization method based on tollgate model. In: *Information Engineering and Electronic Commerce*, 2009. IEEC'09. International Symposium on. 2009; 285–289.
49. Nguyen DK, van den Heuvel WJ, Papazoglou MP, de Castro V, Marcos E. *GAMBUSE: A Gap Analysis Methodology for Engineering SOA-based Applications*. Springer-Verlag: Springer Berlin Heidelberg, Ch. GAMBUSE: A, 2009; 293–318.
50. Chung S, An JBC, Davalos S. Service-oriented software reengineering: SoSR. In: *System Sciences, HICSS 2007*.
51. Alahmari S, Zaluska E, De Roure D. A service identification framework for legacy system migration into SOA. In: *Services Computing (SCC)*, 2010 IEEE International Conference on. 2010; 614–617.
52. Haidar AN, Abdallah AE. Composition and customization of Web services using wrappers: a formal approach based on CSP. In: *Software Engineering Workshop, SEW'08*. 2008; 187–194.
53. Zhang Z, Zhou D-D, Yang H-J, Zhong S-C. A service composition approach based on sequence mining for migrating e-learning legacy system to SOA. *International Journal of Automation and Computing* 2010; **7**(4):584–595.
54. Nasr KA, Gross H, van Deursen A. Adopting and evaluating service oriented architecture in industry. In: *Software Maintenance and Reengineering (CSMR)*, 2010 14th European Conference on. 2010; 11–20.
55. Nasr KA, Gross H-G, van Deursen A. Realizing service migration in industry - lessons learned. *Journal of Software: Evolution and Process* 2013; **25**(6):639–661.
56. Khadka R, Saeidi A, Jansen S, Hage J. A structured legacy to SOA migration process and its evaluation in practice. In: *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*, 2013 IEEE 7th International Symposium on. IEEE, 2013; 2–11.
57. Fuhr A, Horn T, Riediger V, Winter A. Model-driven software migration into service-oriented architectures. *Computer Science-Research and Development* 2013; **28**(1):65–84.
58. Sneed HM, Schedl S, Sneed SH. Linking legacy services to the business process model. In: *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*, 2012 IEEE 6th International Workshop on. IEEE, 2012; pp. 17–26.
59. Khadka R, Saeidi A, Jansen S, Hage J, Haas GP. Migrating a large scale legacy application to SOA: challenges and lessons learned. In: *Reverse Engineering (WCRE)*, 2013 20th Working Conference on. IEEE, 2013; 425–432.
60. Creswell JW. *Research Design: Qualitative, Quantitative, and Mixed Method approaches*. Sage Publications: Thousand Oaks, California, USA, 2003.
61. Razavian M, Lago P. A survey of SOA migration in industry. In: *International Conference on Service Oriented Computing, ICSOC*, vol. 215483. Springer: Berlin / Heidelberg, 2011; 618–626.
62. Razavian M, Lago P. A viewpoint for dealing with change in migration to services. In: *2012 Joint Working Conference on Software Architecture & 6th European Conference on Software Architecture*. IEEE, 2012; 201–205.
63. Razavian M, Lago P. A lean and mean strategy: a data migration industrial study. *Journal of Software: Evolution and Process* 2014; **26**(2):141–171.
64. Poort ER, van Vliet H. Architecting as a risk- and cost management discipline. In: *Working IEEE/IFIP Conference on Software Architecture (WICSA)*. 2011; 2–11.
65. Brown N, Cai Y, Guo Y, Kazman R, Kim M, Kruchten P, Lim E, MacCormack A, Nord R, Ozkaya I, Sangwan R, Seaman C, Sullivan K, Zazworka N. Managing technical debt in software-reliant systems. In: *FSE/SDP workshop on Future of software engineering research*. ACM, 2010; 47–52.

66. Ahn K, Park S, Kim H, Kim J, Park J. The design of SOA-based business collaboration system. In: *Networked Computing and Advanced Information Management*. 2008; 707–712.
67. Balis B, Bubak M, Wegiel M. LGF: a flexible framework for exposing legacy codes as services. *Future Generation Computer Systems* 2008; **24**(7):711–719.
68. Canfora G, Fasolino AR, Frattolillo G, Tramontana P. A flexible wrapper for the migration of interactive legacy system to Web services. In: *Software Maintenance and Reengineering, CSMR 2006*; 344–346.
69. Canfora G, Fasolino AR, Frattolillo G, Tramontana P. Migrating interactive legacy systems to Web services. In: *Software Maintenance and Reengineering, CSMR 2006*; 36–46.
70. Cetin S, Altintas NI, Oguztuzun H, Dogru AH, Tufekci O, Suloglu S. A mashup-based strategy for migration to service-oriented computing. In: *Pervasive Services, IEEE International Conference on*. 2007; 169–172.
71. Cetin S, Ilker Altintas N, Oguztuzun H, Dogru AH, Tufekci O, Suloglu S. Legacy migration to service-oriented computing with mashups. In: *Software Engineering Advances, ICSEA 2007*; 21–31.
72. Chen F, Yang H, Guo H, Xu B. Agentification for Web services. In: *Computer Software and Applications Conference, COMPSAC 2004. Proceedings of the 28th Annual International*. 2004; 514–519.
73. Guo H, Guo C, Chen F, Yang H. Wrapping client-server application to Web services for Internet computing. In: *Parallel and Distributed Computing, Applications and Technologies, PDCAT; 2005*; 366–370.
74. Heckel R, Correia R, Matos C, El-Ramly M, Koutsoukos G, Andrade L. *Software Evolution*. Springer: Berlin Heidelberg, 2008; 139–170.
75. Huang H, Tan H, Zhu J, Zhao W. A lightweight approach to partially reuse existing component-based system in service-oriented environment. *High Confidence Software Reuse in Large Systems*, Hong M (ed.). Springer: Berlin Heidelberg 2008; **5030**:245–256.
76. Huang G. Study and design of enterprise information-based system based on SOA. In: *Computer Science & Education, ICCSE'09*. 2009; **5030**:933–936.
77. Ilk N, Zhao JL, Hofmann P. On reuse of source code components in modernizing enterprise systems. *Advanced Management of Information for Globalized Enterprises* 2008; **5030**:1–5.
78. Lee R, Kim H-K, Yang HS. An architecture model for dynamically converting components into Web services. In: *Software Engineering Conference, 2004. 11th Asia-Pacific*. 2004; 648–654.
79. Lewis G, Morris E, Smith D. Service-oriented migration and reuse technique (SMART). In: *Software Technology and Engineering Practice, 2005. 13th IEEE International Workshop on*. 2005; 222–229.
80. Lewis G, Morris E, Smith D. Analyzing the reuse potential of migrating legacy components to a service-oriented architecture. In: *Software Maintenance and Reengineering, 2006. CSMR 2006. Proceedings of the 10th European Conference on*. 2006; 9–23.
81. Li S, Tahvildari L. E-BUS: a toolkit for extracting business services from java software systems. In: *ICSE Companion 08: Companion of the 30th international conference on Software engineering*. 2008; 961–962.
82. Liu R, Chen F, Yang H, Chu WC, Lai Y-B. Agent-based Web services evolution for pervasive computing. In: *Software Engineering Conference, 2004. 11th Asia-Pacific*. 2004; 726–731.
83. Lucia AD, Francese R, Scanniello G, Tortora G. Developing legacy system migration methods and tools for technology transfer. *Software: Practice and Experience* 2008; **38**(13):1333–1364.
84. Matos C. Service extraction from legacy systems. *Graph Transformations*, Ehrig H, Heckel R, Rozenberg G, Taentzer G (eds.). (Lecture Notes in Computer Science, vol. **5214**), Springer: Berlin Heidelberg, 2008; 505–507.
85. Nguyen DK, van den Heuvel W-J, Papazoglou MP, de Castro V, Marcos E. Gap analysis methodology for business service engineering. In: *Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on*. 2009; 215–220.
86. Xu Y, Duan Q, Yang H. Web-service-oriented customer relationship management system evolution. In: *Software Technology and Engineering Practice, 2005. 13th IEEE International Workshop on*. 2005; 39–48.
87. Zhang Z, Yang H. Incubating services in legacy systems for architectural migration. In: *Software Engineering Conference, 2004. 11th Asia-Pacific*. 2004; 196–203.
88. Zhang B, Bao L, Zhou R, Hu S, Chen P. A black-Box strategy to migrate GUI-based legacy systems to Web services. In: *Service-Oriented System Engineering*. 2008; 25–31.
89. Zou Y, Kontogiannis K. Web-based specification and integration of legacy services. *CASCON'00: Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research*, 2000.
90. Iocola P. When legacy meets SOA: achieving business agility by integrating New technology with existing software asset. In: *Systems Conference, 2007 1st Annual IEEE*. 2007; 1–8.
91. Bao L, Yin C, He W, Ge J, Chen P. Extracting reusable services from legacy object-oriented systems. In: *software maintenance (ICSM), 2010 IEEE International Conference on*. 2010; 1–5.
92. Bissyande T, Reveillere L, Bromberg Y-D, Lawall J, Muller G. Bridging the gap between legacy services and Web services. In *Middleware 2010. Vol. 6452 of Lecture Notes in Computer Science*, Gupta I, Mascolo C (eds.). Springer: Berlin / Heidelberg, Ch. 14, 2010; 273–292.
93. Chung S, Won D, Baeg S-H, Park S. Service-oriented reverse reengineering: SWIH model-driven re-documentation and candidate services identification. In: *Service-Oriented Computing and Applications (SOCA), 2009 IEEE International Conference on*. 2009; 1–6.
94. Djelloul B, Mimoun M, El Kader MA. Towards reengineering Web applications to Web services. *International Arab Journal of Information Technology* 2009; **6**(4):359–364.

95. Koschel A, Kleiner C, Astrova I. Mainframe application modernization based on service-oriented architecture. In: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD'09. Computation World:2009; 298–301.
96. Koufi V, Malamateniou F, Papakonstantinou D, Vassilacopoulos G. Using ESB and BPEL for evolving healthcare systems towards pervasive, grid-enabled SOA. In *Information Systems Development*, Papadopoulos GA, Wojtkowski W, Wojtkowski G, Wrycza S, Zupancic J (eds.). Springer, US, Ch. 18, 2010; 167–175.
97. Marchetto A, Ricca F. From objects to services: toward a stepwise migration approach for Java applications. *International Journal on Software Tools for Technology Transfer (STTT)* 2009; **11**(6):427–440.
98. Millham R. Migration of a legacy procedural system to service-oriented computing using feature analysis. In: Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on. 2010; 538–543.
99. Mulcahy JJ, Huang S, Veghte AB. Leveraging service-oriented architecture to extend a legacy commerce system. In: Systems Conference, 2010 4th Annual IEEE. 2010; 243–248.
100. Nakamura M, Igaki H, Kimura T, Matsumoto KI. Extracting service candidates from procedural programs based on process dependency analysis. In: Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific. 2009; 484–491.
101. Tibermacine C, Kerdoudi M. From Web components to Web services: opening development for third parties. In *Software Architecture*, vol. **6285** of Lecture Notes in Computer Science, Babar M, Gorton I (eds.). Springer: Berlin / Heidelberg, 2010; 480–484.
102. Wang T, Zhou X, Wei J, Zhang W. Towards runtime plug-and-play software. In: Quality Software (QSIC), 2010 10th International Conference on. 2010; 365–368.
103. Zhang Z, Yang H, Zhou D, Zhong S. A SOA based approach to user-oriented system migration. In: Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on. 2010; 1486–1491.
104. Zhou N, Zhang L-J, Chee Y-M, Chen L. Legacy asset analysis and integration in model-driven SOA solution. In: Services Computing (SCC), 2010 IEEE International Conference on. 2010; 554–561.
105. Almonaies AA, Alalfi MH, Cordy JR, Dean TR. Towards a framework for migrating Web applications to Web services. In: proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research. CASCON'11. IBM Corp., Riverton, NJ, USA, 2011; 229–241.
106. Delsing J, Eliasson J, Kyusakov R, Colombo AW, Jammes F, Nessaether J, Karnouskos S, Diedrich C. A migration approach towards a SOA-based next generation process control and monitoring. In: IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society. IEEE Computer Society, 2011; 4472–4477.
107. Delsing J, Rosenqvist F, Carlsson O, Colombo AW, Bangemann T. Migration of industrial process control systems into service oriented architecture. In: IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society. 2012; 5786–5792.
108. Sonntag M, Hotta S, Karastoyanova D, Molnar D, Schmauder S. *Using Services and Service Compositions to Enable the Distributed Execution of Legacy Simulation Applications*, vol. **6994**. Springer: Berlin Heidelberg, 2011; 242–253.
109. Yu D, Wang J, Hu B, Liu J, Zhang X, He K, Zhang L-J. A practical architecture of cloudification of legacy applications. In: Services (SERVICES), 2011 IEEE World Congress on. Vol. 0. IEEE Computer Society, Los Alamitos, CA, USA, 2011; 17–24.
110. Zillmann C, Winter A, Herget A, Teppe W, Theurer M, Fuhr A, Horn T, Riediger V, Erdmenger U, Kaiser U *et al.*. The SOAMIG process model in industrial applications. In: Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on. IEEE Computer Society, 2011; 339–342.
111. Hsieh S-H, Hsieh S-L, Cheng P-H, Lai F. E-health and healthcare enterprise information system leveraging service-oriented architecture. *Telemedicine and e-Health* 2012; **18**(3):205–212.
112. Baghdadi Y, Al-Bulushi W. A guidance process to modernize legacy applications for SOA. *Service Oriented Computing and Applications*. Springer-Verlag: London, 2013; 1–18.
113. Khadka R, Reijnders G, Saeidi A, Jansen S, Hage J. A method engineering based legacy to SOA migration method. In: Software Maintenance (ICSM), 2011 27th IEEE International Conference on. IEEE, 2011; 163–172.
114. Sosa E, Clemente PJ, Conejero JM, Rodríguez-Echeverría R. A model-driven process to modernize legacy web applications based on service oriented architectures. In: Web Systems Evolution (WSE), 2013 15th IEEE International Symposium on. IEEE, 2013; 61–70.