State of the Practice in Service Identification for SOA Migration in Industry

Manel Abdellatif[1,2](✉), Geoffrey Hecht[1,4], Hafedh Mili[1], Ghizlane Elboussaidi[3], Naouel Moha[1], Anas Shatnawi[1], Jean Privat[1], and Yann-Gaël Guéhéneuc[2]

[1] Département d'informatique, Université du Québec à Montréal, Montréal, Canada
manel.abdellatif@polymtl.ca
[2] DGIGL, Polytechnique Montréal, Montréal, QC, Canada
[3] Ecole de Technologie Supérieure, Montréal, QC, Canada
[4] DCC, University of Chile, Santiago, Chile

Abstract. The migration of legacy software systems to Service Oriented Architectures (SOA) has become a mainstream trend for modernizing enterprise software systems. A key step in SOA migration is the identification of services in the target application, but it is a challenging one to the extent that the potential services (1) embody reusable functionalities, (2) can be developed in a cost-effective manner, and (3) should be easy to maintain. In this paper, we report on state of the practice of SOA migration in industry. We surveyed 45 practitioners of legacy-to-SOA migration to understand how migration, in general, and service identification (SI), in particular are done. Key findings include: (1) reducing maintenance costs is a key driver in SOA migration, (2) domain knowledge and source code of legacy applications are most often used respectively in a hybrid top-down and bottom-up approach for SI, (3) industrial SI methods focus on *domain* services–as opposed to *technical* services, (4) there is very little automation of SI in industry, and (5) RESTful services and microservices are the most frequent target architectures. We conclude with a set of recommendations and best practices.

1 Introduction

Software maintenance consumes the bulk of IT budgets, as legacy applications become harder to extend, fix, or even *sustain in operation*–with the obsolescence of the hardware and software on which they were built [14]. Rewriting from scratch is seldom an option, for two major reasons: (1) the substantial effort required to rewrite *(tens of) millions* of lines of code, and (2) the amount of valuable domain or procedural knowledge embodied, but otherwise undocumented, in such applications. Hence, application *modernization* remains essential to ease the maintenance of legacy systems and make them more flexible without losing their business values.

The migration of legacy systems to Service-Oriented Architecture (SOA) is one avenue for their modernization. SOA makes it possible to develop complex

C. Pahl et al. (Eds.): ICSOC 2018, LNCS 11236, pp. 634–650, 2018.
https://doi.org/10.1007/978-3-030-03596-9_46

and inter-organizational applications by integrating/orchestrating high-quality and reusable services. The migration of legacy systems to SOA requires identifying services, which is considered the most challenging task of the overall migration process [13]. Service Identification (SI) consists in identifying, in legacy systems or/and from the system domain decomposition, reusable services that may embed valuable business logics. The reusable services must meet a range of expectations concerning their capability, quality of service, and efficiency of use.

Several SI approaches have been proposed in the literature of academic research [4,10,16]. However, these approaches are based on few evidence and out of touch with industry practices due to the little knowledge about the state-of-the-practice of SI as part of *'real'* migration projects. Therefore, in this paper, we want to minimize the gap with industry by understanding industrial practices and identifying best practices of legacy applications migration to SOA in general and SI in particular. Thus, we wanted to answer the following research questions:

- **RQ1.** What kind of systems are being migrated to SOA?
- **RQ2.** Why are such systems being migrated?
- **RQ3.** What approaches are being used for application migration, in general, and SI in particular?

To this end, we surveyed 45 SOA migration practitioners using an online survey, and interviewed eight of them to answer these questions. We identify key findings including: (1) reducing maintenance costs is a key driver in SOA migration, (2) domain knowledge and source code of legacy applications are most often used respectively in a hybrid top-down and bottom-up approach for SI, (3) SI focuses on *domain* services, (4) there is little automation–the process of migration remains *essentially* manual, and (5) RESTful services and microservices are the most frequent target architectures.

This paper is structured as follows. Section 2 presents the related work. Section 3 describes the study design. The results of the online survey are presented in Sect. 4. Section 5 reports the results of the interview sessions, which are discussed in Sect. 6. We conclude in Sect. 7 with recommendations and best practices for SI.

2 Related Work

Seldom are research surveys related to the migration of legacy systems to SOA. Razavian and Lago [15] conducted an industrial survey about legacy-to-SOA migration approaches by targeting seven SOA solution providers. They argued that all industrial migration approaches share the same set of activities as they start by transforming the business models of legacy systems to continue with service design and implementation. Taibi et al. [19] also performed a survey on migration to microservices architectures, filled by 21 practitioners. The survey focus on the migration reasons and report that reducing maintenance cost is the main reasons of legacy-to-microsevices migration. Although these study approaches and results are similar to ours, their focus differ deeply as we cover more in details state of the practice on SI in terms of (1) the methods used, (2) the artifacts used by these methods, (3) the processes of these methods, and (4) the outputs of these processes. We also cover more participants and report best practices for SI.

More in general, a number of primary studies have been proposed in the literature about SI. Many of the proposed techniques rely on Business Process Models (BPMs), to identify services within the context of legacy migration [11,17,18]. These techniques decompose processes into tasks and then map these tasks to legacy source code elements to identify candidate services. Other SI techniques use heuristics based on the *technical properties* of services, as reflected in various metrics [1,10,12]. Such techniques often use these metrics to drive clustering and machine learning algorithms that identify software artefact clusters as candidate services. However, they do not always produce good candidate services. Other AI-based techniques use ontologies and Formal Concept Analysis to identify services in legacy systems [2,8,20]. They too, are complex and not ready for industrial applications. Other techniques put service interfaces around *existing* functional components and subsystems [4,6,16,17] but do not *infer* such clusters from finer-grained software artefacts. These so-called *wrapping-based techniques* are suitable for integration problems, but do not solve the maintenance issues.

3 Study Design

The survey presented in this paper was conducted between October 2017 and March 2018 and aimed to investigate the state of the practice in SOA migration, and, in particular, service identification. Our study consisted of four main phases:

A- Preparation of the Online Survey. We created a web-based survey (see https://goo.gl/forms/EE31KeA7R7pUeTYI2) using Google forms. The survey was prepared based on our literature survey of the state-of-the art methods for SI and informal discussions with some subject matter experts. This helped identify the dimensions/aspects of the questionnaire, the individual questions, and the possible answers for each question. Before publishing the survey, we performed a pilot with six potential subjects, three from academia and three from industry, to validate the relevance of the questions, their wording, the coverage of the answers, etc. The six 'testers' went through the questions and suggested minor changes. The final survey contained six sections: (1) participants' professional and demographic data, (2) type of migrated system, and reasons for the migration, (3) general information about SI methods (perception of importance, strategy, inputs, level of automation), (4) detailed technical information about SI (technique/algorithm used, quality metrics considered), (5) Information on the types of services sought and targeted technologies, and (6) Information about the tools used, and the suggested best practices.

B- Selection of Participants. We targeted developers with an industrial experience in SOA migration. Identifying and soliciting such developers was challenging. We relied on (1) information about companies that offer modernization services, (2) online presentations and webinars made by professionals that had the professional's contact information, and (3) search queries on LinkedIn profiles, such as "*legacy migration OR legacy modernization OR SOA architect OR SOA migration OR Cloud migration OR service migration OR service mining*". Once we identified potential participants, we sent them invitations via e-mail, LinkedIn, Facebook, and Twitter. We chose *not* to solicit more than three professionals from any given company to: (1) have an as wide representation as possible, and (2) to not overburden a single organization with our request.

C- Online Survey. We invited 289 professionals to participate, and kindly asked them to forward our invitations to other people in their network who have experience in SOA migration and SI. The survey was completed by 47 people, two of which did not participate in SOA migration projects and whose responses were discarded, leaving us with 45 complete responses.

D- Validation. We assessed the reliability of the answers in the online survey by looking for spurious/facetious answers, contradictions between answers, etc. To be able to validate improbable answers, one question of the survey asked participants if they agreed to be contacted for a follow-up 30-min interview, and 24 out of 45 agreed; however only eight could be interviewed in the end and the results of those interviews are shown in Sect. 5. A two-pass method [5] was used to analyze our transcripts of the individual interviews (see https://goo.gl/ZYv2Ut for sample transcripts). The first pass of the analysis consists of *thematic coding* to identify broad issues related to legacy-to-SOA migration in general and SI in particular. The second pass of analysis was performed using *axial coding* to identify relationships among the identified issues. Major factors were identified using *meta-codes*. The *meta-codes* were then used to identify similar patterns across the data from the multiple interviewees. Overall, the answers were plausible, and the eight detailed interviews confirmed the questionnaire answers, although provided us with more in-depth information.

4 Analysis of the Results of the Online Survey

In this section, we describe the results of our survey. We allowed multiple answers to most questions of the survey, therefore the sum of the computed percentage may exceed 100% in some cases. The given percentages are computed based on the total number of participants who answered a given question.

A- Participants. We reached a total of 45 participants who were involved in legacy-to-SOA migration projects in different capacities: 50% were software architects, 23.7% were directors of technology, and 21% were software engineers. The remaining 5.3% of participants mentioned other positions such as migration specialists, project managers and CEOs. They work in different industries: 64% were in technology and telecommunication, 20% from banking and insurance, 12.8% from health, and 3.2% from education. In terms of experience, 78% had more than 10 years of experience, and this was somewhat reflected in their age distribution: 23% were less than 35 years old, 39% were between 36 and 45, 20.5% were between 46 and 55, and 17.5% were over than 55 years old.

B- Types of Legacy Systems. The results show that the legacy systems included mainframe applications, transactional applications, ERP systems, monolithic client-server applications, software-analysis tools, and visualization tools; 13% of these were less than 5 year old, 18% were between 5–10 year old, and 69% were more than 10 years old. In terms of size, 62% of the systems were deemed large, 36% were medium size, and 2% were deemed small. Cobol (52.6%) and Java (57%) were the two most prominent languages for legacy systems. Figure 1 shows the many other languages used in the migrated applications.

Finding 1: *Practitioners migrate different types of old legacy systems implemented mainly in Cobol and Java.*

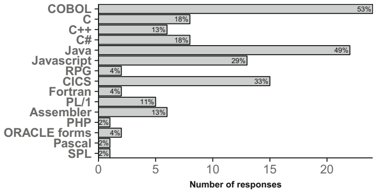

Fig. 1. Used languages in legacy systems

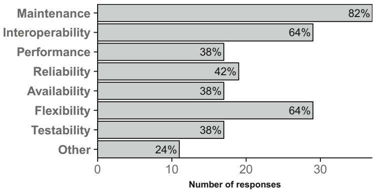

Fig. 2. Reasons for migration

C- Motivations for Legacy-to-SOA Migration. We asked about the motivations behind the migration of legacy systems to SOA. We provided a list of reasons for the migration as shown in Fig. 2. The most prevalent motivation was to reduce maintenance costs (82%). Practitioners reported during the interviews that the cost involved in maintaining legacy systems can be high due to (1) the poor/outdated documentation of these systems; (2) the obsolete/old programming languages used to implement these systems; (3) the decay and difficulty to understand the architectures, designs, and implementations of these legacy systems; and, (4) the lack of developers with the skills necessary to maintain these systems. The second most significant motivation to migrate legacy systems was to improve their flexibility (64%). We have been told during the interviews that practitioners have difficulties with legacy systems because they do not allow companies to have the flexibility required to carry out day-to-day tasks for evolving systems to meet new business requirements. The improvement of the interoperability of the legacy systems with the migration to SOA was the third most significant motivation (64%). During the interviews, practitioners told us that SOA eases the interoperability of heterogeneous systems by exploiting the pervasive infrastructure of the network. Thus, it offers the possibility to continue using and reusing the business capabilities provided by legacy systems in new, modern systems [9]. Improving system availability and testability as well as improving performance were other motivations of industrial legacy-to-SOA migration projects (38% each). Participants also mentioned other business and technical reasons for migrating legacy systems to SOA, such as improving business agility, having new user interfaces, and embracing new technologies.

Finding 2: *Reducing maintenance costs, improving the flexibility and interoperability of legacy systems are the main motivations to migrate legacy systems.*

D- Importance of Identifying Reusable Services from Legacy Systems. We asked about the importance of identifying *reusable* services in the source code of legacy systems during the migration process: 87% of the participants qualified it as important while only 13% thought that it is not. We explain this agreement by the benefits of software reuse, which (1) *increases software productivity* by shortening software-development time, (2) *reduces software development costs* by avoiding the reimplementation of existing services, (3) *reduces maintenance costs* because the reused services were functional and have been well-tested, and (4) reduces the risk of introducing new failures into the process of enhancing or creating new business services. We explain the 13% of disagreement as some participants undertook top-down migrations rather than bottom-up or mixed migrations and the former does not require identifying services in source code.

Finding 3: *Identifying services in legacy applications is an important step in legacy-to-SOA migration.*

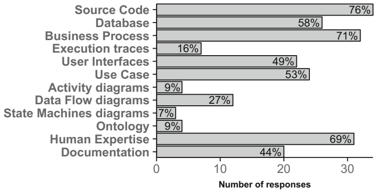

Fig. 3. Used inputs for SI in industry

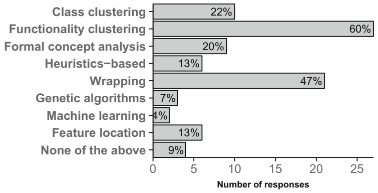

Fig. 4. Used techniques for SI in industry

E- Inputs of SI. Through a literature review, we identified several types of inputs used for SI. We listed these inputs in our survey and asked participants on which inputs they relied to identify services. Figure 3 shows that the most used inputs were source code, business process models, databases, and human knowledge. 76% of the participants relied on the recovery of the business logic of legacy systems through the analyses of the source code to identify services with high business value. 71% relied on the mapping of business processes with the legacy source code to extract reusable services through human expertise. These artifacts may help software engineers to have a better understanding of the legacy systems. Finally, participants rarely relied on ontologies, activity diagrams, state machine diagrams, and execution traces to identify services. This observation may be due to their unavailability or complexity to establish especially since our practitioners deal with large systems.

Finding 4: *Many software artifacts can be used for SI. Practitioners mostly used source code, business process models, databases, and human expertise. There is a very low interest in relying on ontologies, activity diagrams, state machine diagrams, and execution traces to identify services.*

F- Directions of SI. We asked participants about their choices of direction for identifying services. We proposed three directions: (1) Top-down: starting from domain-specific conceptual models, like business concepts and process models, to identify services, which are then specified and implemented through a forward engineering process; (2) Bottom-up: starting by analyzing the existing legacy system artifacts and identifying services from reusable legacy code; and (3) Mixed: starting both from domain-specific conceptual models and the analyses of the legacy system to identify services. We found that 53% of the participants use a mixed direction to identify services. Participants used almost equally top-down and bottom-up directions with 23% and 24% each. We explain these observations as follows: (1) practitioners relied on source code and business process models as reported in *Finding 4*, (2) practitioners also relied on extracting the business logic of legacy systems because documentation was not always available, (3) practitioners prioritized reuse and avoided development from scratch to reduce time and costs, and (4) practitioners faced limitations due to the lack of legacy experts/knowledge, unavailability of up-to-date documentation, program comprehension, and challenges of reverse-engineering legacy systems.

Finding 5: *Practitioners highly rely on a mixed direction to identify services during legacy-to-SOA migration process.*

G- Techniques for SI. We asked about the techniques that they used to identify services. As depicted in Fig. 4, we found that 60% of the participants relied on clustering functionalities of the legacy systems and exposing these clusters as services. 47% of them relied on some black-box techniques, like wrapping, because they either consider the migration as an integration problem or did not want to modify the core functionalities of the legacy systems because it provided useful services. We observed a low interest in using machine-learning techniques, formal-concept analysis, or meta-heuristic algorithms to identify reusable services. Using these techniques may be challenging for practitioners because they are dealing with large systems to migrate and so the knowledge required to establish these techniques could be time consuming and may not lead to optimal results. Also these techniques are researched by academics and not mainly by professionals (see Sect. 2). Finally, 9% of the participants mentioned that they did not use any techniques and performed SI manually.

Finding 6: *Functionality clustering and wrapping are the most used techniques of SI in industry.*

H- Analyses Types for SI. We asked about the types of analyses that they performed for SI (static, dynamic, textual, and–or historical analyses). We observed that 87% of the participants relied on static analyses of the source code of the legacy systems to identify services. 43% of them reported that they relied on runtime analyses. Participants also relied on textual analyses for the identification processes. Textual analyses include elements such as features identification techniques, natural language processing, legacy documentation analysis, etc. Only 18% of the participants reported that they relied on historical analyses (analyses of different versions of the legacy system) to extract candidate services, which may be due to (1) the unavailability of several versions of the legacy system and (2) the difficulty to study the evolution of a legacy system to gather valuable information to identify reusable services.

Finding 7: *Practitioners mostly relied on static analyses of the source code of their legacy systems for SI.*

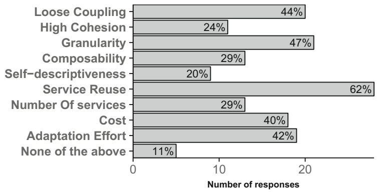

Fig. 5. Desired services quality criteria for SI in industry

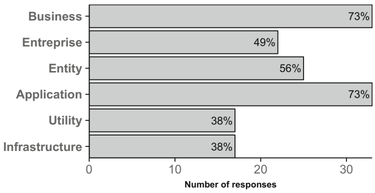

Fig. 6. Types of the migrated services

I- Services Quality Criteria. We asked the participants about the quality metrics/criteria that they sought during SI. We identified the quality criteria, listed in Fig. 5. Service reusability was the most sought quality criteria by the participants (62%), followed by service granularity (47%), and loose coupling (44%). Reusability was defined by participants as both a measure of the amount of source code reused in the services and the amount of services reused in the systems. Costs and the adaptation effort were also considered by the participants during the identification process (40% and 42% respectively). However, they did not consider self-descriptiveness, high cohesion, composability, and the total numbers of services when identifying services.

Finding 8: *Only few service quality criteria are desired by practitioners in the SI process: reusability, granularity, and loose coupling.*

J- Types of the Identified Services. We provided practitioners with a taxonomy classifying service types into domain-specific (business) services versus domain-neutral (technical) services. The provided domain-specific services are: (1) business services, enterprise services, application services and entity services. The technical services are utility services and infrastructure services. We report the results in Fig. 6. As domain-specific services represent the business core functionalities of SOA, they were the most targeted services (i.e., business and application services) during the SI processes compared to technical services. Utility and infrastructure services were the less targeted services because they are SOA-specific services and utility services are relatively easy to implement.

Finding 9: *SI is a business-driven process that prioritized the identification of domain-specific services rather than technical services.*

K- Service Technologies. We asked the participants about the services technologies that they targeted during migration. We found that 75% of them use REST services, 60% use SOAP and only 4.5% use Service Component Architecture (SCA). Surprisingly, half of the participants reported that they focused on identifying microservices in legacy systems. While there is no precise definition of this architectural style, microservices are gaining interest among organizations, especially with the growth of the Cloud and DevOps paradigms [7].

Finding 10: *Restful services are the most targeted service technology in legacy-to-SOA migration.*

L- Automation of SI. We asked the participants about the degree of automation of their SI techniques as well as the tools they used to this end (for the lack of space, we report the list of tools in https://goo.gl/ZYv2Ut). We found that many different tools are being used as well as manual analyses and in-house tools put together for the migration of particular legacy systems. However, not one set of tools supports adequately the migration of systems to SOA. We also found that the majority of the techniques used by the participants to identify services are either semi-automatic (51%) or manual (42.3%). Only three participants (6.7%) mentioned the use of tools to identify automatically reusable services. We highly believe based the reported tools that these fully-automatic approaches deal with re-engineering tasks as well as wrapping techniques that automatically expose legacy systems functionalities as services.

Finding 11: *There is a lack of automation of SI techniques in industry but input from human experts is essential to annotate/qualify intermediate or final results of SI.*

M- Threats to the Validity.

Construct validity threats refers to the extent to which operationalizations of a construct (in our case the survey and interview questions and terminology) do actually measure what the theory claims. To minimize this threat, we used both open and closed questions in the survey and tried to minimize the ambiguities through our pilot study as we mentioned in Sect. 3.

Internal validity acquiescence bias is a kind of response bias where respondents have a tendency to agree with all the questions in the survey or to indicate a positive connotation. It is sometimes referred to the tendency of a respondent to agree with a statement when in doubt. We mitigate this threat by doing interview sessions to validate the survey answers. We also eliminated responses where participants selected all the possible choices for all the questions. We also mitigate this threats by checking the responses to questions that are related to each others (e.g. the used input for SI and the identification direction, etc.). Finally, we decided not to have incentives for participating in our survey to minimize social desirability bias.

External validity the survey participants might not be representative of the general population of software developers migrating legacy systems to SOA. Thus, the generalizability of our survey might be limited. The mitigation of this threat to validity is very challenging because (1) we are targeting practitioners with very specific technical skills; and (2) professionals are in general not eager to communicate the details of their in-house tools and techniques of modernization approaches. To mitigate this risk, we advertised our survey through various channels (e.g., LinkedIn, Twitter, Facebook and email) and targeted professionals from different legacy modernization companies. Also, to the best of our knowledge, our sample size is one of the largest such size among many papers in empirical software engineering in general and modernization in particular (see Sect. 2). Participants could freely decide whether to participate in the study or not (self-selection). They were informed about the topic of the survey, the estimated time to complete the survey, the research purpose of the study and the guarantee of the anonymity of their identity and that of their answers.

5 Interview Sessions

Eight participants among the 45 agreed to carry phone interviews. Table 1 describes their profiles. The initial purpose of the interviews was to ask the participants to elaborate on some of their answers or resolve contradictions among their answers. However, the interviews often ended with discussions on issues not addressed in the survey. The interviews also allowed participants to rectify some of their answers and for our part to obtain presentations and white papers about their migrations. We now summarize salient facts gathered from the interviews in terms of the adopted migration strategies and directions of SI.

Table 1. Information about the participants in the interview sessions

Participant	Profession	Years of experience	Country
P1	Technical solution architect	25 years	Germany
P2	Legacy modernization and enterprise IT architect	18 years	India
P3	Mainframe modernization specialist	33 years	USA
P4	Legacy and data Center senior consultant	30 years	Italy
P5	Software modernization expert	15 years	Canada
P6	IT architect	20 years	Canada
P7	Director of technology	12 years	Canada
P8	Software engineer	7 years	France

5.1 Migration Strategies

We asked the interviewed participants about their adopted migration strategies to migrate legacy software systems to SOA. We identified three strategies: *rehosting*, *legacy system re-architecture*, and *rehosting followed by re-architecture.*

A- Rehosting (P1, P3, P6) consists of moving a legacy system with minimal changes from one platform, typically legacy mainframes, to more modern alternatives such as Linux, Unix, or Windows in two ways: (1) by running emulators or virtual machines of the source platform on the target platforms (e.g., a VMS or AS400 emulator/virtual machine on Linux) or (2) by rewriting the parts of the systems that interface with the target platforms. The business logic and data of the legacy systems remain unchanged on the new platform. Rehosting is done when the hardware or software platforms become too costly to support–or are no longer supported–by the manufacturer/vendor. The systems can be *wrapped* within services once they are integrated on the new platforms.

B- Legacy systems re-architecture (P1, P2, P5) is a migration strategy in three steps applied each on three different layers: the *application-code layer*, which contains the legacy code in Cobol, PL1, etc.; the *information layer*, which gathers data access through files, databases etc.; and, the *business-process layer* which describes the business logic of the system. The three migration steps are: (1) *Legacy system discovery and migration planning*, it focuses on cataloging and understanding all the assets in the legacy systems, *"we are importing the code in our toolset. We are looking for dependencies and capturing business processes. We just take a look if everything is complete"* said P1; (2) *Design*, this phase consists in designing the new system, a "*future case analysis repository*" that contains enhancements to the legacy business processes and all the modernized data and future SOA model are stored in the *information layer*; and (3) *Target system development and test*, this phase *"is a very classic software development phase just to develop and test the new SOA based system"* said P1.

C- Legacy systems re-hosting and re-architecture (P1, P4, P7, P8) aims to build new SOAs that yield the business values of the legacy systems while minimizing costs related to legacy hardware and ensuring a progressive and incremental replacement of the legacy code. This migration strategy is mainly used to "*minimize disruption while ensuring business continuity*" said P8. It avoids the "*big-bang*" migration strategy by (1) re-hosting the legacy systems to modern platforms to minimize hardware costs, (2) creating wrappers to hide the internal legacy functionalities, and (3) replacing progressively the legacy code.

5.2 Directions of SI

We detail in this section the adopted directions of SI by our interviewees.

A- Bottom-up strategies (P1, P3, P4, P5) consist of identifying artifacts of the legacy code that implement reusable business functions to be repackaged as services: *"Through the bottom-up SI strategy we want to reuse the existing legacy code certainly, but not the architecture. Most of legacy systems that we deal with have about 25 millions lines of code. If we want to write them again, it can take* *years"* said P5. The artifacts used by bottom-up approaches include the source code, data flow analyses, legacy system interfaces, databases, documentations, and human expertise. Reverse-engineering tools were used to understand the legacy systems and extract their business logic, especially when there is a lack of documentation and experts. Several interviewees reported using both in-house and open-source tools to reverse-engineer systems. For example P5 used an in-house tool based on the Knowledge Discovery Model (KDM) to obtain call and data-flow graphs of COBOL systems. P5 relies on functionality clustering and pattern matching to identify reusable services: *"We are searching for patterns and we are looking for business rules or business logic that match with these patterns and heuristics. We are doing data flow analysis with slicing to identify reusable business functions that can be grouped and deployed as services"*. Many interviewees (P1, P3, P4, and P5) also relied on techniques for detecting code clones to identify reusable services. *"What we also do in many cases is looking for duplicate code pattern because in many cases business rules are duplicated, you need to decide what to take out of this"* said P4.

B- Top-down strategies (P7) starts from the analysis of domain-specific conceptual models and requirements to specify the services of the targeted SOA. P7 recommended to use this strategy when (1) legacy source code is not available, (2) legacy source code is not reusable, (3) cost of re-engineering and integrating legacy systems is high, and (4) organizations are mature enough in terms of business processes. P7 reported that they adopted a semi-automatic top-down strategy for SI to migrate a legacy banking system to SOA. They used BPMN process models of the banking legacy system as input. They begun by identifying the entity-services and the application services. They then moved to higher-level services, such as task-centric services, and finally developed an orchestration layer that represented business services. This strategy is based on the analysis of "*information*" used in each activity of the business processes. P7 explained that *"information could be a document, reports, windows, screens, an entity etc. that is required in the execution of an activity"*. To identify entity services from business processes, *key information* manipulated in the business process models was identified. An *information* is considered as *key* by P7, if it meets at least one of the following conditions: (1) its number of occurrences exceeds a given threshold and (2) it is related to a highly solicited activities.

C- Mixed strategies (P1, P2, P3, P4, P5, P6, P8) rely on reverse-engineering techniques to document the legacy systems, extract the business logics, and identify reusable pieces of code that can be exposed as services. They also rely on forward-engineering techniques to define the business processes of the target SOAs and to design and implement the services. P2 said: *"Sometimes if the source code is available and documentation is not, we use some parser based tools to reverse-engineer these applications. These tools will create some documentation from the code and then that documentation is used to do the forward engineering and complete the targeted SOA road map"*. He also argued that *"through this documentation we create use cases for forward engineering to complete the identification, the design and the implementation of the services"*. P1 said *"we document everything in our system and then at the very end we identify the business rules mark them in the code and you can extract them afterwards [...] we have a list of business processes and core code description and we also document this, and based on this we are creating our service-oriented material"*.

D- Final choice of the identified services is a manual process driven by subject-matter experts. P5 said: *"We make proposition about the services that we identify and ask the customer if it makes sense. Sometimes at technical level we have better knowledge than the customer but not from business process level"*.

5.3 Threats to Validity

Internal Validity. Social desirability is a bias that leads any respondent to deny undesirable traits and report traits that are socially desirable. To minimize this threat, we did not put any incentives for the participants to participate in the interviews. We also guaranteed the interviewees their anonymity and emphasized that all the reported information will be only for research purposes.

External Validity. Information from our interviews is not generalizable as the number of the interviewees is a bit on the low end for software engineering studies. However, it is acceptable given that it is unquestionably difficult to find interviewees in legacy-to-SOA migration domain. We only sought to obtain a better understanding of the results of the online survey. Also, Table 1 shows that our interviewees are experts in legacy-to-SOA migration and, thus, that our sample is still reliable because we are dealing with subject-matter experts.

Conclusion Validity. The information from our interviews also show some threats to the validity of our conclusion because some interviewees contradict each other or, for one interviewee, change their answers to the survey. However, this threat is acceptable because we use these interviews with the purpose to mitigate and discuss the answers to the survey.

6 Discussions

After analysing the survey and interview data, we highlight the following facts.

Importance of Service Identification From Legacy Systems. We observed that SI is an important step in the overall legacy-to-SOA migration process for most practitioners, especially when it comes to the context of SI from legacy systems. As emphasized by P4 "It is important because we are able to identify the reusable of the code. SI is considered as the main helmet to measure the impact of the migration [...] you need to understand the migration cost which is in many cases too expensive, you have to cut the cost by identifying reusable pieces of the legacy code in a cost-effective way". Thus, the agreement about the importance of identifying reusable services in industry can be explained by the benefits of software reuse. However it should be noted that SI is not always fine-grained as mentioned by P6 "We basically wrap the legacy system and expose all its functionalities as services".

Business-Value Driven Service Identification. We notice that not all service quality criteria are equally targeted by practitioners. Unlike academia, efforts in industrial SI strategies are made to deal with business constraints such as the recovering of the business logic of legacy systems and extracting reusable functionalities with high business value. There are big investments by practitioners to preserve the business logic of legacy systems rather than to care about service quality constraints. As it is stated by P2, SI is mainly driven by the customers business needs: "Our customers do not really focus on these features. I am not saying that these quality criteria are not necessary but because of the business constraints, considering service quality metrics become a lower priority comparing to timing to finish the project and return in investment issues". Also, technical constraints may hinder the consideration of quality metrics: targeting quality metrics while identifying reusable pieces of code that can be exposed as services may not be suitable for all legacy technologies like mainframe legacy systems for example: "For banking mainframes systems it is not easy to use that kind of approach since we are dealing with routines" stated P4.

Automation and Experts Feedbacks. The full automation of SI process is not the primary focus of practitioners. It is even the case of big modernization companies as it is stated by P1 "In our SI methodology we are not doing everything automatic, automation is about 70% of all the migration project". However, there is automation in wrapping and reverse engineering techniques to document and extract the business logic of legacy systems when the documentation is absent. Feedback loop with business analysts and customers is considered essential by practitioners to decide about the pertinence of a candidate identified service. Practitioners also do not take the risk to try to fully automate the SI process as it is a challenging problem with unpredictable results, time consuming and needs a lot of research investments.

Gap Between Academia and Industry. None of the interviewed practitioners mentioned the use of research papers or academic resources for their migration projects. From the point of view of practitioners, "academics do not see the larger picture of the real industrial problems and challenges they are facing" as stated by P2. The lack of cost-effective academic SI technique and the lack of validation on real enterprise-scale systems is a problem that hinders knowledge transfer between academia and industry.

7 Conclusion and Recommendations

We presented a state of the practice of SI in industry to support the migration of legacy software systems to SOA. We surveyed 45 industrial practitioners and interviewed eight of them to collect, analyze, and report their experiences with the migration of legacy systems. Our results showed that reducing maintenance costs and improving the flexibility and interoperability of legacy systems are the main motivations to migrate these systems to SOA. They also showed that SI is perceived by practitioners as an important step for the migration, in particular to identify reusable code in the legacy systems. In addition, they showed that SI is a process driven by business value rather than quality criteria, even though some practitioners consider some quality criteria (mainly reusability, granularity, and loose coupling). Finally, our results showed that SI remains a manual process in which human experts' feedbacks is essential.

We drew several lessons from these results, which we summarize as follows.

Service Identification is a Business-Value Driven Process. When identifying services we must focus on the functional clusters that implement *useful* and *reusable* business functions. We must not focus on *technical/architectural* properties, as many academic techniques do (e.g., [1,2,11,12]). While the research literature identifies many service types, SI must first and foremost focus on identifying domain services, i.e., entity, business, and process services that have business values.

A Deep Understanding of the Domain and a Great Familiarity with the Legacy Systems are Necessary. Because SI is driven by business values, we must have a deep understanding of the *domain*, including its *main entities* and *processes*. We must also be familiar with the legacy systems in which to identify services. While the research literature assumes that the SI techniques are independent of the legacy experts, they should allow incorporating seamlessly knowledge from experts who are familiar with the systems.

The Input Must be Source Code and Production Data. According to the old military adage, if the terrain differs from the map, trust the terrain. With legacy systems, documentation (the map) may be absent or awfully out of date. The source code (the terrain) is the only reliable source of information about what the *current* system does. Production data, as stored for example in databases or JCL scripts, also contain valuable, up-to-date information about the systems. Therefore, while the research literature has been recently studying Q&A forums and other similar documentations, it should also strive to reconcile and improve the analyses of different up-to-date sources of information.

The Output Must be High-Value, Coarse-Grained Services. Regardless of the targeted SOA technology, be it SOAP, RESTful, or microservices, the output of any SI technique must be high-value, coarse grained *domain services*. While the research literature has been concerned by the implementation (and quality thereof) of services, it should seek to define, assess, and optimize the business values of the identified services.

The Process Must Follow a (Proven) Methodology. Migration projects are complex endeavors, regardless of the source and target technologies. There is value in adopting or adapting an existing *SOA migration methodology* because such methodologies prescribe *processes*, *deliverables*, and *quality metrics* to guide the migration. While the research literature proposes techniques, the participants recommended using existing methodologies including Oracle's *OUM Methodology*, IBM's *Service-Oriented Modelling and Architecture* (SOMA) methodology [3], and devising SI techniques as *parts* of these methodologies.

In future work, we plan to build an exhaustive catalogue of best practices for SI. We also want to identify issues that the research community can address to facilitate knowledge transfer between academia and industry in the context of legacy-to-SOA migration.

Acknowledgements. The authors would like to thank all the practitioners who replied to us. This work was supported by the Fonds Québécois de Recherches - Nature et Technologies. The second author is also supported by the Proyecto FONDECYT de Postdoctorado N3180561.

References

1. Adjoyan, S., Seriai, A., Shatnawi, A.: Service identification based on quality metrics object-oriented legacy system migration towards SOA. In: SEKE, pp. 1–6 (2014)
2. Amiri, M.J., Parsa, S., Lajevardi, A.M.: Multifaceted service identification: process, requirement and data. ComSIS **13**, 335–358 (2016)
3. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., Holley, K.: SOMA: a method for developing service-oriented solutions. IBM Syst. J. **47**(3), 377–396 (2008)
4. Canfora, G., Fasolino, A.R., Frattolillo, G., Tramontana, P.: Migrating interactive legacy systems to web services. In: CSMR, p. 10 (2006)
5. Charmaz, K., Belgrave, L.: Qualitative interviewing and grounded theory analysis. In: The SAGE Handbook of Interview Research, pp. 347–365 (2012)
6. Chenghao, G., Min, W., Xiaoming, Z.: A wrapping approach and tool for migrating legacy components to web services. In: ICNDC, pp. 94–98 (2010)
7. Di Francesco, P., Malavolta, I., Lago, P.: Research on architecting microservices: trends, focus, and potential for industrial adoption. In: ICSA, pp. 21–30 (2017)
8. Djeloul, M.: Locating services in legacy software: information retrieval techniques, ontology and FCA based approach. WSEAS Trans. Comput. (Greece) (2012)
9. Erl, T.: SOA Principles of Service Design. Prentice Hall PTR, Upper Saddle River (2007)
10. Gysel, M., Kölbener, L., Giersche, W., Zimmermann, O.: Service cutter: a systematic approach to service decomposition. In: ESOCC, pp. 185–200 (2016)
11. Huergo, R.S., Pires, P.F., Delicato, F.C.: A method to identify services using master data and artifact-centric modeling approach. In: ACM SAC, pp. 1225–1230 (2014)
12. Jain, H., Zhao, H., Chinta, N.R.: A spanning tree based approach to identifying web services. Int. J. Web Serv. Res. **1**(1), 1 (2004)
13. Khadka, R., Saeidi, A., Jansen, S., Hage, J.: A structured legacy to SOA migration process and its evaluation in practice. In: MESOCA, pp. 2–11 (2013)
14. Lewis, G., Morris, E., O'Brien, L., Smith, D., Wrage, L.: Smart: the service-oriented migration and reuse technique. Technical report, DTIC Document (2005)
15. Razavian, M., Lago, P.: A survey of SOA migration in industry. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 618–626. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25535-9_48
16. Rodríguez-Echeverría, R., Maclas, F., Pavón, V.M., Conejero, J.M., Sánchez-Figueroa, F.: Generating a REST service layer from a legacy system. In: Escalona, M.J., Aragón, G., Linger, H., Lang, M., Barry, C., Schneider, C. (eds.) Information System Development, pp. 433–444. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07215-9_35

17. Sneed, H.M., Verhoef, C., Sneed, S.H.: Reusing existing object-oriented code as web services in a SOA. In: MESOCA, pp. 31–39. IEEE (2013)
18. Souza, E., Moreira, A., De Faveri, C.: An approach to align business and it perspectives during the SOA services identification. In: ICCSA, pp. 1–7 (2017)
19. Taibi, D., Lenarduzzi, V., Pahl, C.: Processes, motivations, and issues for migrating to microservices architectures: an empirical investigation. IEEE Cloud Comput. **4**(5), 22–32 (2017)
20. Zhang, Z., Yang, H., Chu, W.C.: Extracting reusable object-oriented legacy code segments with combined formal concept analysis and slicing techniques for service integration. In: QRS, pp. 385–392 (2006)