Delft University of Technology
Software Engineering Research Group
Technical Report Series

Realizing Service Migration in Industry - Lessons Learned

Khalid Adam Nasr, Hans-Gerhard Gross and Arie van Deursen

Report TUD-SERG-2011-004

TUD-SERG-2011-004

Published, produced and distributed by:

Software Engineering Research Group
Department of Software Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

ISSN 1872-5392

Software Engineering Research Group Technical Reports:
http://www.se.ewi.tudelft.nl/techreports/

For more information about the Software Engineering Research Group:
http://www.se.ewi.tudelft.nl/

Note: Accepted for publication in the Journal of Software Maintenance and Evolution: Research and Practice , 2011, Wiley-Blackwell

JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE
J. Softw. Maint. Evol.: Res. Pract. 2011; **00**:1–7

Research

Realizing Service Migration in Industry - Lessons Learned

Khalid Adam Nasr[1,2*,†], Hans-Gerhard Gross[2] and Arie van Deursen[2]

[1] *Logica the Netherlands*
[2] *Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands*

SUMMARY

In this paper, we present two descriptive case studies covering the re-engineering and further evolution of adopting service oriented architecture (SOA) in industry. The first case was carried out for a company in the transport sector with an application portfolio of over 700 systems. The second case study was conducted for an organization in the public sector. The goal of both case studies is to identify possible benefits and drawbacks of realizing SOA in large organizations in order to obtain a better perspective on the real, rather than the assumed, benefits of SOA in practice. We describe how the two cases were developed and carried out, and discuss the experiences gained and lessons learned from adopting SOA in the two organizations. Based on these findings, we propose several directions for further research.

KEY WORDS: service oriented architecture. re-engineering. case study. system integration

1. INTRODUCTION

Today, many enterprises worldwide are focused on increasing their business flexibility and simplifying their Information Technology (IT) infrastructure in order to better meet their business objectives. In a world of rapid and constant change, business evolution is forcing IT to innovate constantly. IT systems must be adaptable but also sustainable. The conflicting demands of flexibility, business efficiency, reduced complexity of current infrastructure, and the cutting of software evolution cost bring forth a number of dilemmas that organizations are facing. The Service Oriented Architecture (SOA) paradigm has been positioned as a promising

*Correspondence to: Khalid Adam Nasr, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
†E-mail: k.a.nasr@tudelft.nl

way forward for organizations to address these dilemmas. Through the flexible and proficient design, implementation and operation of systems realized as SOA, organizations are supposedly able to adapt to the demands of rapidly changing environments [23].

However, there is still insufficient solid empirical evidence that indicates to which extent the positive assumptions made about SOA hold true [6, 15]. This paper examines some of the assumed benefits and the potential bottlenecks of implementing SOA. This is based on real-life experience with implementing SOA solutions in the IT industry. The case studies presented highlight the challenges of implementing SOA solutions for the IT industry, and they contribute to developing a research agenda.

This paper adopts the industry-as-a-laboratory approach [25]. Real industry cases can provide and support practical tests and contribute to the further development of SOA technology, and the methods and tools for its implementation. Such empirical research will also point at areas that are still not sufficiently put on the research agenda. The cases discussed in this paper concern projects carried out by Logica in the transport and public sector. Logica is a European IT and business services company, employing 40,000 people across 36 countries. It provides business consulting, systems integration, and IT and business process outsourcing services.

The first case study presents a three-stage process of (1) designing a Proof-of-Concept (PoC), (2) making organizational decisions for a smooth transition to a SOA, and (3) the actual re-engineering and evolution of part of the system towards SOA. The second case study, which benefited from some of the lessons learned in the first case, is also presented in a three-stage process: (1) assessment and start-up, (2) organizational decisions to support the service identification process and its smooth transition, and (3) the implementation of a new engineered sub-system.

The article is structured as follows. Section 2 of this paper reviews related work. Section 3 presents the case study setup. Sections 4 and 5 provide the reports of the two case studies, which discuss the various steps of design and implementation. Section 6 discusses the lessons learned and relevant future research directions. Section 7 concludes this paper by summarizing the key contributions.

2. RELATED WORK

Over the past few years, the benefits and promises of SOA have been quite extensively discussed in the literature [2, 12, 22, 28]. Vendors are busily promoting the hardware, software, tools and services that support SOA implementation [7, 20]. Some researchers have warned businesses not to blindly follow the vendor hype [16, 20]. In fact, an extensive literature survey was not able to derive much empirical research that proves, or disproves, the hypothesized advantages of SOA [37].

Kontogiannis, Lewis and Smith [15] point out that many of the case studies that do provide anecdotal evidence of the business value of SOA are sponsored by vendors. The number of independent industry case studies that document and discuss aspects of SOA implementation in practice, is still rather limited. To the best of our knowledge, these studies include [4, 8, 10, 17, 19, 29, 35, 37, 38]. Therefore, more rigorous empirical research - using the

"industry as a research laboratory approach" proposed by Potts [25] - is needed before hard conclusions can be drawn about the reusability, increased agility and cost reduction that SOA promises.

Based on studies of five different companies that have implemented SOA, Schelp and Aier [29] argue that while SOA appears to contribute to corporate agility, the reuse and cost cutting potential - as claimed by vendors and consultants - is less evident in all cases. In general, investments only pay off in the long run. The authors conclude that the measurement and evaluation of SOA is still "in its infancy"[29].

Costs in software evolution are primarily dominated by software maintenance issues. Some authors argue that maintenance consumes up to 90% of the total software cost [8]. Canfora et al. [4] also looked into the question of cost-effectiveness, specifically in relation to the wrapping technique that the authors present. Their two case studies stress the need to devise methods for supporting the process tasks of reverse engineering and validation, as effort data showed that these were the most expensive tasks.

With their case study into verification and validation as part of a software development project for the US Department of Defense, Gehlot and Pujari [10] show the benefits of a model driven approach for the development of SOA systems. Zdun et al. [37] also focus on a modeling approach in their industrial case study performed in a large telecommunications company. They present a model validation tool that the authors developed, which turned out to be valuable in order to support modeling the process-oriented integration of services.

The case study documented by Cuadrado et al. [8] describes the steps involved in the evolution of an existing medical imaging system, based on medium-sized legacy systems, to SOA principles. While it provides guidelines for practitioners on how to use certain tools, its focus on architecture recovery tasks is less relevant for our paper.

Lewis et al. [19] present a case study of migration of legacy components to services carried out for the US Department of Defense. The study suggested that disciplined analysis is needed to navigate the complex process. Therefore, the authors further developed the Service-Oriented Migration and Reuse Technique (SMART), an approach for collecting the necessary information, and a method for identifying the risks of a SOA migration effort [18].

The case study presented by Kulkarni and Dwivedi [17] concerning a leading US financial institution, openly discusses the risks of "riding on the SOA hype wave". Services that are, in theory, well-designed can, nevertheless, cause usage and maintenance challenges, once implemented. Like in [18] and [19], the authors present a framework for SOA adoption, called "InSOAP". They conclude that "creation of web services in itself does not really help to deliver increased reusability, flexibility and responsiveness to change. This requires a strategic approach towards building a strong service foundation as well as sound engineering principles to realize it" [17].

In their empirical exploratory study, Kokko et al. [14] investigate SOA adoption in nine Finnish organizations by conducting interviews with key staff. SOA adoption challenges pointed out include resistance to change and unexpected extra work compared to traditional approaches.

Sneed [30] describes a pilot project for a large company which entailed the migration of business functions in legacy code to service oriented architecture. For this purpose, the author developed the COB2WEB tool kit, a tool used to adapt the target programs and generate code

to wrap these programs behind the WSDLs interfaces. One of the findings from this study is that in terms of time spent the testing effort required was 2-3 times higher than the wrapping effort. The author emphasizes the need for finding ways to automate the testing of web services using test data from the original components. Canfora and Massimiliano [5] report on a survey of recent research achievements related to SOA testing. They analyze the testing challenges from the perspective of different stakeholders and they present solutions for different levels of testing. Research in the area of SOA testing, which includes [3, 9, 31, 1], is fairly recent and the strategies, methods and techniques proposed are not yet sufficiently mature.

In summary, the specific challenges for SOA that these case studies have pointed out include:

- developing a method that provides a clear framework, based on rigorous analysis, for the design, migration and implementation of SOA [17, 19],
- developing methods and techniques for expensive tasks of the migration process, in particular re-engineering and validation [4],
- examining the actual cost-effectiveness of SOA as expressed in its payback time [4, 29]
- managing organizational issues, such as resistance from staff, often especially senior members who may lack comprehension for web-based technologies [35].

3. CASE STUDY SETUP

Enterprises and public sector organizations around the world are increasingly eager to embrace SOA, hoping to benefit from its acclaimed advantages, which include increased business efficiency through more flexibility as well as a higher Return on Investment (ROI). The specific expectations toward SOA include [23, 20, 7]:

- achieving higher productivity by speeding up the introduction and implementation of new products,
- simplifying the integration process during mergers and acquisitions, and
- achieving cost reductions.

In the literature available to date there is little empirical evidence about the extent to which these expectations are actually met. One of the reasons for this may be that the benefits are aimed at the long term, and hence take a long time to materialize. Another reason will be that increased productivity, flexibility, and reduced costs require a standard or existing system to compare to, which usually is not available.

Rather than trying to measure the supposed benefits, what we set out to do in this paper is take a close look at the risk and cost side of introducing SOA. In particular, we aim at identifying challenges, best practices to mitigate these in ongoing projects, and research directions that can help to provide long-term solutions to some of these challenges.

Thus, the research questions that we address in this paper are:

1. How can organizations realize the claimed benefits of SOA and which best practices help to minimize the risks and costs?
2. Which factors complicate implementation of and transition to SOA in practice?

3. How can research help to enhance SOA development while further minimizing its risks and costs?

In order to answer these questions we conducted two case studies in two different organizations. For each of these cases, the unit of analysis is a project that involves the first transition to SOA within the organization. The first case study involves a descriptive study in the transport sector. The case describes the subsequent phases of introducing, designing and implementing SOA within the organization. For each phase we include a short description of the actual work undertaken: what was the goal, which stakeholders were involved, what activities were performed and with what result? This descriptive case material is analyzed in order to derive the challenges and complications encountered during the process, as well as the benefits and best practices for the implementation and operation of SOA in this particular case.

In order to expand our findings and confirm or refute our earlier results, we conducted a second case study at a different organization in the public sector, which was also introducing SOA for the first time. Again, the different phases of the project (analysis, start up and implementation) are described, including the steps involved, decisions and activities undertaken by different stakeholders. Lessons learned in the first case study were included and tested in this second case.

Data collection in these case studies followed an approach that can be characterized as participant observation [27]. The first author was involved in implementing both projects as an IT consultant and member of the development team. The data collection thus followed the first degree method, in that the researcher was in direct contact with the subjects and collected data in real time. In order to limit the effects of using only one data source, this study not only followed the above method, but also examined qualitative and quantitative data in relevant project documentation (bid, analysis documents, use cases, planning documents) as data sources. The study includes the insights from other team members (business consultants, architects, developers, project managers) gained through informal discussions as well as formal interviews with customer staff (managers and product owners) and project team members.

The objective of both case studies is to examine which factors complicate implementation of and transition to SOA in practice and to find out through which best practices the advantages of SOA can be maximized and its risks and costs minimized. By gaining a deeper understanding of SOA implementation projects in practice, this study aims to contribute to the improvement of SOA implementation [21]. By evaluating each phase, we provide answers for questions 1 and 2, covering benefits, challenges, and best practices identified in the cases at hand. Question 3, concerning the research directions, is addressed in Section 6, as this requires a broader perspective.

4. SOA IN THE TRANSPORT DOMAIN: CASE STUDY REPORT

This section describes an industry case study about a SOA solution designed and implemented for one of Logica's customers in the transport sector, an organization which, for reasons of

confidentiality, in this paper we will refer to as 'Ndi Moyo'.[†] Ndi Moyo wanted to implement SOA in order to achieve better flexibility and agility, to reduce the complexity of its existing infrastructure and to minimize development and maintenance costs. Therefore a project was initiated to deliver this solution. The project team included the following roles: project manager, business architect, business analysts, SOA architect and developers. This team closely collaborated with Ndi Moyo's IT and management staff.

Integration of systems is a crucial aspect of SOA. Together with Ndi Moyo, the project team therefore started with investigating their existing IT landscape. This process was meant to gain a comprehensive understanding of its many different systems, to try and define specific functional and non-functional requirements, and, moreover, to explore the drivers behind the business initiative.

The investigation showed that Ndi Moyo's application landscape was a mix of many systems. These consisted of packaged and custom-made applications running on heterogeneous platforms using diverse data formats and technologies (J2EE, .NET, and various legacy applications). Over 700 applications were identified, mostly implemented as single monolithic structures and integrated in a complex point-to-point integration approach. The purpose of the overall project was to re-engineer this integration, in order to eventually achieve a well manageable, all-round integration infrastructure. The Enterprise Service Bus (ESB) was to be used as the middleware technology to expose the applications as services.

Based on the first analysis, and because a full transition can only be made incrementally, the decision was made to start off with a pilot project instead of immediately initiating a full-fledged design and implementation of SOA. The pilot led to a second project, which tackled the organizational structure necessary to support the transition to SOA. In the last phase of the project, the first integration solutions were implemented.

4.1. Phase 1: A SOA Proof-of-Concept at Ndi Moyo

The SOA transition project at Ndi Moyo started off with a pilot project for two different reasons. Firstly, as mentioned above, Ndi Moyo's existing application landscape was very complex and diverse and the project team therefore judged it wise to first start with a Proof-of-Concept (PoC). Secondly, the team realized it was important to actively involve the different business units in the process from the very start. This made it possible to explore, together with Ndi Moyo, their different motivations for this project, as well as the possible organizational and personal inhibitors to making it successful. Creating as much ownership as possible is crucial especially when a far-reaching change such as a migration to SOA is at stake [13].

The Logica team suggested to include a web portal solution in the pilot project. Experience had shown that business managers as well as decision makers and staff can more easily relate to visual information (of a web page, which is much more tangible), than to technical concepts. The web portal solution clearly visualizes the possibilities of exposing business functions as services, spanning the whole organization.

[†]Ndi Moyo means 'This is the life' in Chichewa.

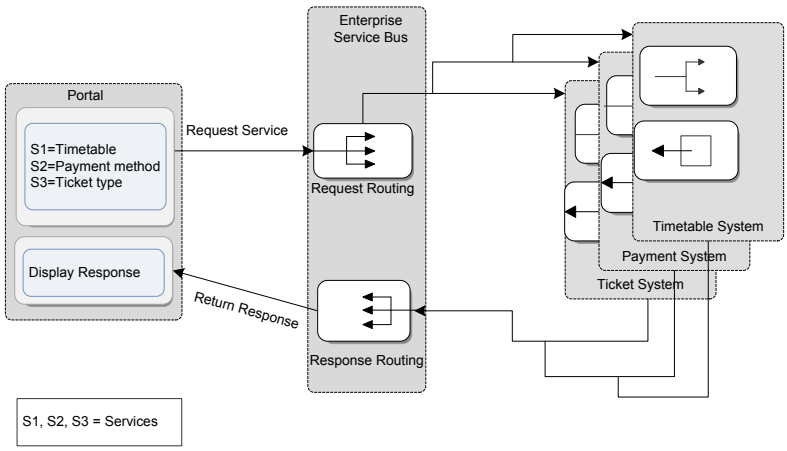

Figure 1. Architecture of the web portal solution

A portal provides authentication features and supports personalization to provide customized content to users. This content is presented as a variety of services. The services could be as simple as providing a corporate news item to staff of a global company. A service user sends an office location name and receives a report back on the latest business news for this specific location. Alternatively, portal users can interact with services by clicking links or submitting forms on the portal page, resulting in requesting services from service providers.

4.1.1. Scenario

We implemented typical usage scenarios in the portal. One example represents the following process: A customer wants to buy a ticket for a journey. He has already consulted Ndi Moyo's online timetable for his preferred date, time and means of travel. To buy his ticket he can choose between various payment methods (credit card, company business card, debit card etc.). He can also choose his preferred way of receiving the ticket (E-ticket, mail, collecting from the customer service desk at Ndi Moyo). This scenario requires the interaction of at least the following company systems: Portal, Timetable system, Payment system, and Ticket system (see Figure 1).

We used the Enterprise Service Bus (ESB) as the middleware technology to integrate these various services and to contain the differences between service implementations. As stated by Papazoglou et al. [23, p. 8] the ESB approach allows loosely coupled systems to take part in the integration and "to break up the integration logic into distinct and easily manageable pieces."

4.1.2. Benefits, challenges and best practices

From a business perspective, the Proof-of-Concept (PoC) examined the potential benefits and capabilities of SOA principles if applied to Ndi Moyo's requirements. The PoC, moreover, verified that the specific technology solutions operated successfully within Ndi Moyo's technical architecture landscape.

A common challenge during SOA transitions is the issue of ownership. In this first stage of the project it was evident that this would indeed become a challenge for Ndi Moyo. There was confusion over who would be responsible for the different services and their governance once these would have been integrated. This issue became a bottleneck in phase 2 which we will cover in Section 4.2.

To start small and design the SOA transition according to incremental steps proved a good practice. Moreover, the pilot portal indeed was a useful way of conveying the notion of SOA and its potential benefits for the business of Ndi Moyo. The selected technology worked and this formed the basis for a go or no-go decision. As it happened, the customer was keen on continuing with the SOA journey. The level of investment made by Ndi Moyo up to this stage was approximately one million Euro. Ninety percent of this investment went to middleware infrastructure, management and licensing; less than ten percent was spent on professional resources (business consultant, senior developer, junior developer, project manager) to design and engineer the PoC. These costs did not include the time spent by Ndi Moyo's personnel. Given that the PoC was specifically meant to convince the various business units of the added value of SOA, it is interesting to note that Ndi Moyo had already invested approximately 900,000 Euro before being assured of the cooperation of its internal organization.

4.2. Phase 2: Setting up the Integration Center of Competence (ICoC)

The second part of the project focused on organizing Ndi Moyo's internal organization to adapt to the SOA paradigm. Ndi Moyo's different business units have distinctive IT systems that provide various business functions spanning the organization. These are used by a wide variety of internal business units as well as by external partners and customers.

The nearly 700 different IT systems utilize a large number of diverse concepts, technologies and protocols. Each business unit can independently make its decision about how to deliver solutions. These solutions mostly concern separate business units without any reference to a centrally manageable and proficient integration infrastructure.

All this meant that in order to achieve a successful and controlled integration of these various systems through the Enterprise Service Bus, and to coordinate and organize the different integration requests put forward by the business units, Ndi Moyo needed one focused central point where all the integration requests were collected. An integration request means that business unit A wants to connect its systems to business unit B in order to provide, consume or exchange each other's services.

An Integration Center of Competence (ICoC) was set up to coordinate the integration requests and to keep an inventory of the application landscape and the different systems' readiness to apply SOA principles. Moreover, Ndi Moyo wanted to get better insight into, and control over, the high costs of software evolution and maintenance that it was facing.

Setting up ICoC was no easy task. First of all, the different business units were apprehensive about the central role that ICoC was going to play regarding IT budgets and decisions. These business units had been operating relatively independently in a decentralized structure, and did not want to give away the responsibility for controlling and managing their business services to a central third party. They were resistant to the idea that ICoC would take over the re-engineering of their systems.

4.2.1. Work process methodology

A work process methodology was developed to guide the decision making on integration, explicitly involving ICoC as well as the relevant business units, as is visualized in Figure 2. The initial request formulated by a business unit contains the vision and scope of the business scenario, the business rules and the software architecture of the system consuming the service. In realizing this first step of the integration request the following roles from the business unit are involved: business consultant, information analyst and software architect. They can consult ICoC for supporting them in this task.

The request is submitted to ICoC for validation. For this, ICoC developed a matrix that allowed easy insight into costs, effort and value of possible integrations as well as comparison between different integrations on these parameters. The selection criteria are based on an assessment of feasibility, reusability and routing. Moreover, validation includes determining whether the requested service already exists in the organization's service portfolio. If not, the appropriate business unit to provide this service is determined, and this business unit will be engaged in the process. If the validation fails, the request is sent back to the requesting business unit for additional information.

If the request passes the validation, ICoC starts the High Level Design (HLD). This includes an impact analysis, business modeling, initial project plan, and estimates for the time and cost for re-engineering, implementation and maintenance. The next step is that the business unit service provider develops a Detail Level Design (DLD), which needs approval from ICoC.

If approval is given, ICoC prepares the proposal which includes the total cost estimate. This needs to go to the business unit service consumer for approval. If approval is given, a detailed project plan is developed by ICoC, which includes interface definitions, supplementary specifications, resources planning as well as a test plan. In addition to the roles already involved, at this point, the release coordinator and project manager are assigned.

The next step is optional. If the request concerns a particularly complex integration, a Proof of Concept (PoC) is developed and tested. If it is a mainstream integration, then this step is omitted. If the PoC is accepted, the actual development starts.

Ndi Moyo invested approximately one million euro for this second phase of setting up and operating the ICoC; note that this covered only the external expertise hired.

4.2.2. Benefits, challenges and best practices

One challenge encountered in this phase was that the great majority of Ndi Moyo's systems were not mature enough to expose their business functions as services without being re-

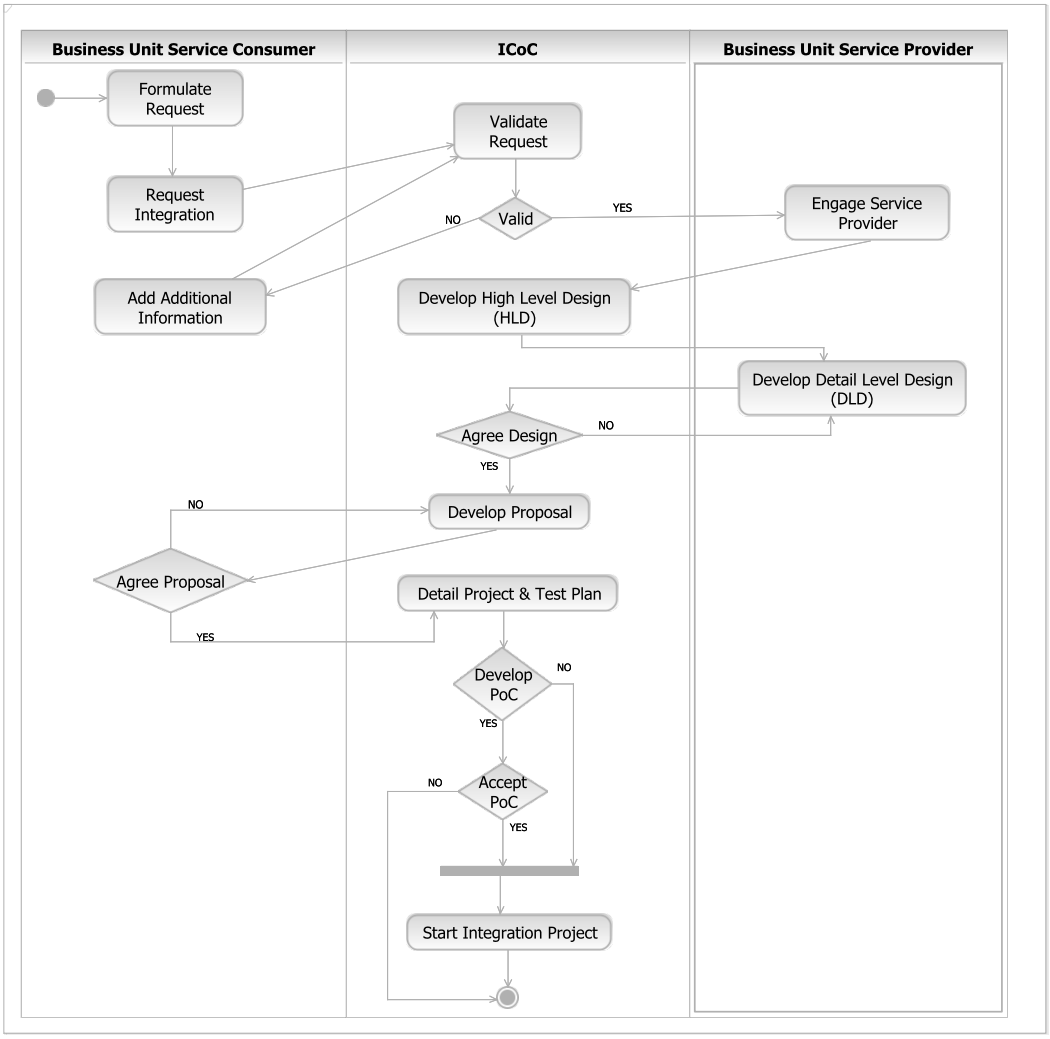

Figure 2. Work methodology

engineered according to SOA principles. In many cases, this meant that effort and money had to be invested in the re-engineering of these systems.

Secondly, a SOA transition is not only a technology challenge but also an organizational one. One tricky issue is how to deal with the resistance to change, and, in particular, changing responsibilities, that business and IT staff often display [14]. Resistance to change is a well-documented phenomenon and many of the methods suggested in the change management literature are applicable to overcome this also in a SOA context. In this project, the method mostly resorted to was creating awareness among the different stakeholders through their active involvement. Workshops and training of personnel to make them more confident about the transition also helped to take away some of the resistance.

Eventually, setting up ICoC as an independent, central coordination and assessment point was a successful way to manage the multifaceted process of this SOA transition. It is crucial that a central unit such as ICoC is an independent body, in order to diffuse the possible distrust between different business units that hamper their willingness to cooperate. The work process methodology that was developed ensured the active engagement of the different business units.

ICoC facilitated a proper coordination of the integration requests from different business units. Moreover, it helped estimating the scope of a particular SOA integration as well as defining how to realize the integration in the most efficient and effective way. The structured methodology allowed Ndi Moyo to make informed go or no-go decisions in an early stage of the proposed integrations. Lastly, for the first time did Ndi Moyo have clear, centralized insight into the actual integration and maintenance costs of services of its different business units.

4.3. Phase 3: Integrating a new service

Based on the positive assessment carried out by Integration Center of Competence (ICoC), Ndi Moyo decided to proceed with the implementation of three integration requests submitted by the business units. These services were required for the portal. The starting point was a preliminary classification of possible services that could be built from components of the existing system, based on the criteria mentioned above. This section briefly discusses one of the implementations.

4.3.1. Scenario

The business function implemented concerns a new "report visitor" service, that was integrated into Ndi Moyo's existing corporate portal. Figure 3 depicts a Unified Modeling Language (UML) sequence diagram for the detailed logic of the service interaction. This service, provided to the portal users, involves two business units. The first business unit is the one responsible for the corporate portal (Portal BU). The second one, the facilities business unit (Facilities BU), is responsible for the visitor registration system. The portal user registers a visitor, or checks for available parking spaces in the visitor registration system. The portal communicates with the back end system of the Facilities BU through the Enterprise Service Bus (ESB). The back end registration system in this example is the service *provider*, providing the business function (register visitor) as a service. This service is provided by an application that is described using the web service definition language (WSDL) and deployed as a Web service. The WSDL

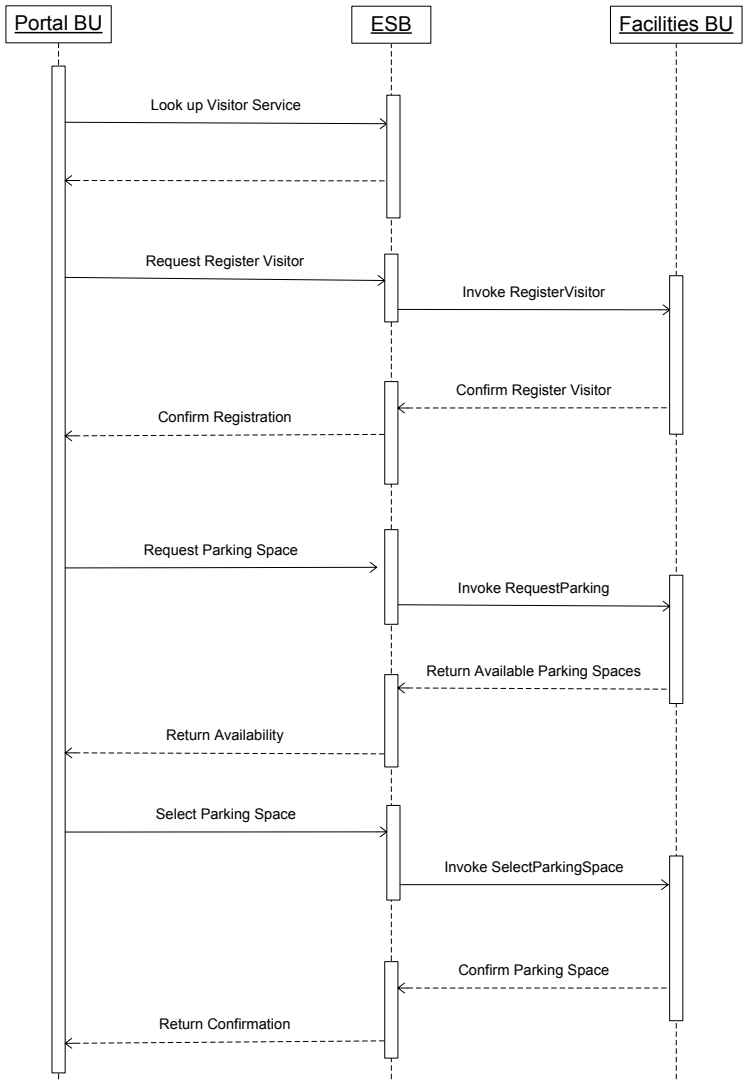

Figure 3. Service provider-consumer sequence

specification defines SOAP bindings for the service. The portal system in this case is the service *requester*, and it consumes the "register visitor" service. The WSDL service definition specifies the provider service interface and implementation that is used to access the target service.

Integrating this new service required the involvement of a software engineer for the portal, a software engineer for the back-end service provider application, an integration engineer for the ESB integration, an IT architect, a tester and a project manager.

4.3.2. Testing the integration

As part of the implementation of this particular integration, an integration test was included. The test descriptions were based on the functional requirements specifications, the acceptance criteria, and the provided scenarios. These tests were performed without back-end systems, but by simulating these systems using stubs. The scope for testing the system was mainly to test the functionality described in the interfaces. Functionality which did not result in any messaging traffic was outside the scope of integration testing, and was assumed to be covered by unit-level testing of the underlying components.

For each test, a separate test data set was available. The test data included the message that was to be sent, a script to send the message, a script to save the result, results and expected results. Messages were simulated using a web service testing tool.

In this case, the service provider developed and tested the integrated service in its development environment. These tests are mainly on the service interface level. The service requester system is simulated. The test is fulfilled when the received data match the expected data defined through the service interface.

The second stage is comprised of running system tests in the test environment where live data and systems are mostly not present (replication from production data and other systems).

The user acceptance testing (UAT) was carried out by Ndi Moyo. This step involves testing the interaction between the systems of various business units in the acceptance environment where a snapshot of the production environment is present.

4.3.3. Benefits, challenges and best practices

A service was identified, developed and deployed, which made the business function "report visitor" available for use by other business units. This successful real-life integration paved the way within the organization to proceed with other service integrations.

By testing this single new integration it was realized that doing the same in a context of many different simultaneous integrations would present serious challenges. There is still a lack of methods and tools for properly testing and managing the effects that one change can have on the overall system of systems. The impact of such a change is likely to be discovered during runtime.

Looking back at all three phases of the Ndi Moyo case, we can conclude that the phased approach applied in this project enabled a thorough analysis as well as good coordination between the various stakeholders and a proper understanding of the well-trained experts. It led to a smooth real-life integration.

5. SOA IN THE PUBLIC DOMAIN: CASE STUDY REPORT

Our second case study involves a project that was carried out for one of Logica's customers in the public sector, an organization which, for reasons of confidentiality, in this paper we will refer to as 'Chinthu'.[‡] The primary goal of the project was to provide new software and hardware for the system responsible for managing the life cycle of smart identification cards (Smart ID system, SmID). The secondary goal of this project was to provide the basis for the organization's full transition to SOA. Chinthu intends to implement SOA to its overall "system of systems" in the near future. Different technologies presently in use by these systems include IBM z/OS mainframe systems (Cobol, CICS, Visual Age, DB2), web application server systems (J2EE) and stand alone systems (.NET, Java). The software of these systems has evolved over the past twenty years, resulting in complex systems connected through a multitude of point-to-point integrations.

The software development methodology used for this project was Rational Unified Process (RUP) because the organization already had its own customized and defined RUP methodology.

The scope of this project was limited to the new SmID and its interfacing systems. First of all, a smart ID system (SmID) had to be designed and developed to replace the existing system (Plain-Old ID system, POID). Chinthu had, for many years, issued plastic ID cards to its citizens and residents. They wanted to change this to using a new, highly secure smart ID card, which contains biometric data (finger print, iris scan, facial photo and signature). This new card allows the government to more comprehensively and securely keep track of its citizens and residents. At the same time, the data stored on the smart card allows cardholders to use a number of e-government services (e.g., apply for new ID card, register newborn child, etc.). Figure 4 illustrates the system context and the interactions between the POID and other systems.

In order to introduce the new SmID, a number of supporting systems also had to be adapted, which included: a) Changing the Card Management System (CMS) to support the new smart ID cards; b) Integrating additional enrollment, self-service and personalization kiosks; c) Integrating the SmID system with the new Biometric Service System (BSS); d) Integrating the SmID with other systems within the organization.

The Logica team responsible for developing the new SmID and adapting the interacting systems, consisted of a project manager, business analysts, architects, software developers and testers. This team closely collaborated with Chinthu's project managers, analysts, architects, developers, staff from the quality assurance department and with its end-users.

5.1. Phase 1: Assessment and start-up

Chinthu had decided that it wanted a new SmID. Moreover, with a view to its long-term goal of a full transition to SOA, the SmID had to support SOA principles for inter-system

[‡]Chinthu means 'identity' in Chichewa.

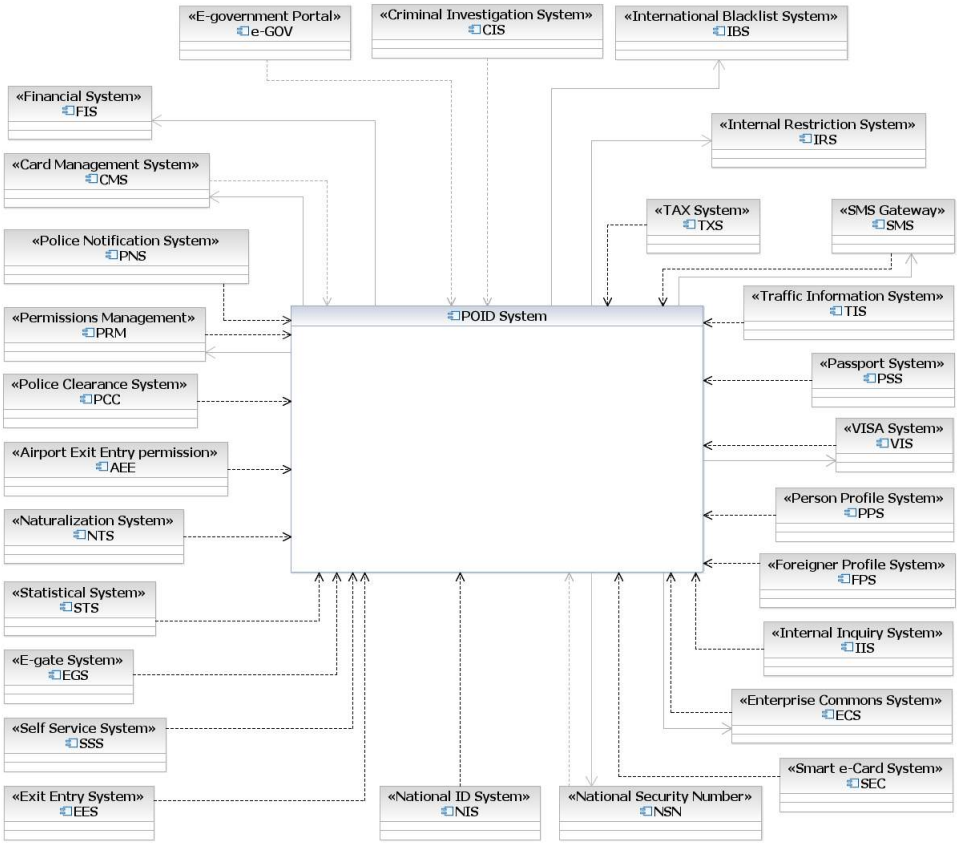

Figure 4. System context diagram

communication. This had three implications. Firstly, the SmID had to be designed to be able to both provide and consume compliant SOA services to and from a large number of other systems within the organization. Just like the existing POID, the new SmID had to guarantee successful interaction with at least twenty-five other systems (e.g., security number system, passport system, police clearance system, financial system). Secondly, the systems which in the existing design accessed the POID and its database directly (POID is a service provider), had to be re-engineered to support SOA principles in order to use the new services provided by the SmID. Thirdly, the external systems providing services currently consumed by the POID (POID is a service consumer) also had to be adapted by the system owners to support the SOA principles and continue providing services to the new SmID.

This multilevel redesign provided a good opportunity to apply the following lessons learned and best practices obtained from the first case study, and test their usefulness for this second case.

5.1.1. Incremental approach

The first case had shown that one big advantage of a phased approach is that it allows transition to happen while business can go on as usual. Therefore, in this second project, the decision was made to start with one system at the time.

The reason for choosing to start with the POID was the organization's wish to have a more secure ID system operational as soon as possible. Because the SmID that was to replace the POID had to be built from scratch, this provided a good opportunity to design it according to SOA principles from the start. As said, the SmID is a system that needs to interact with many other internal and external systems. As a second step, therefore, the business functions of these other systems relevant to the SmID functionalities, were re-engineered to support SOA principles. The organization's intent was that, if these first steps would succeed, the remaining functions of the other systems would also be re-engineered to support SOA. In effect, the SmID served as a pilot for this organization's full transition to SOA.

5.1.2. Commitment to business drivers

Complex transitions in an organization's IT landscape, such as a migration to SOA, not only pose technological challenges, but also challenges related to staff commitment. The technical IT personnel may not share the support for the business drivers as formulated by the decision-makers. Key people that are needed for the successful migration to SOA may resist change, or be insufficiently pro-active.

The first case study showed that a SOA transition is indeed not a simple exercise to perform within a running organization. One condition for success is to have the support and commitment of the management and decision-makers, as well as of the different business units and IT staff. In the first project, this was achieved by starting with a Proof of Concept (PoC) and actively involving the various staff through workshops and training. In this second project, Chinthu's higher management already had a clear understanding of the drivers behind the transition. This was achieved during the bid phase. Because of their clear commitment to go ahead with SOA, there was no need to do a PoC. However, the organization's IT personnel had insufficient understanding of the need for, and benefits of, a transition towards SOA. They had a lack of knowledge of what SOA entails at a higher conceptual level. Moreover, when confronted with the planned changes, they showed clear resistance. Initially, they did not want to participate or share their knowledge of the current systems, which was needed to conduct a good analysis of the requirements for the SmID. They admitted to being worried that their expertise and professional experience with legacy systems may become redundant due to the introduction of SOA.

We learned from the first project that the role of the technical IT personnel in the transition process must not be underestimated. Their intimate knowledge of the organization and its systems is of the utmost importance for a smooth transition process. This was even more

urgent in the second project, where there was a lack of up to date and reliable documentation about the existing systems. Because of the positive experience with providing workshops for IT and business personnel in the first project, the same strategy was used this time. Workshops were organized for key technical IT personnel of various departments of the organization. The objective was to make them familiar with the technical issues involved with a transition to SOA as well as with the organizational benefits. It was made clear that legacy systems can be integrated in a SOA. Moreover, in informal meetings with certain staff members, the team explained that the transition to SOA may introduce new opportunities for them. As a result of these workshops and discussions, the IT staff became visibly more involved in the process. Their initial resistance changed into a cooperative attitude. This facilitated the remainder of the analysis process.

5.1.3. Implementing a middleware

In the first case study, the organization invested a great amount of money for implementing a middleware for its SOA migration in an early stage of the process. Ninety percent of its investment was spent on implementing middleware technology (see 4.1.2). Eventually, it turned out that this middleware was too advanced for what was really needed, which meant unnecessary costs. In order to avoid these, the second project adopted a different approach. The team advised Chinthu to not underestimate the effort and expenses needed to adapt applications and therefore not put all its attention and investment towards the middleware infrastructure. Therefore, the team this time started with the service identification and classification (see 5.2.1) and the adaption of applications. Starting by specifying the interactions and exchange between applications, while at the same time planning for the appropriate middleware solution proved to be a better and more cost-effective method. It allowed the organization to adopt a middleware for its SOA in a way most appropriate for its business needs.

5.1.4. Challenges, benefits and best practices

The positive experience of using an incremental approach in the first case was confirmed in this second project. It allowed the organization to adopt SOA at the pace that best suits its business goals and budget. Choosing the system that serves as the pilot is context specific and depends mostly on the immediate business goals. It is, however, crucial to have a very good understanding of not only the system itself, but also its interactions with interfacing systems. This project confirmed that to gain this understanding, the knowledge of the organization's IT personnel is indispensable. Therefore, their cooperation and commitment need to be assured. If IT staff show resistance to change, focused workshops can help to get them on board. Finally, a lesson learned from the first case which helped to identify a new best practice, was to not invest too much in the middleware infrastructure in the first stage of the transition. Starting with a minimal middleware, which can be expanded if necessary in due course, avoids unnecessary upfront investments.

5.2. Phase 2: Coordination and identification of candidate services

The second phase of the project continued with the business analysis process, but not before taking care of an important organizational issue.

5.2.1. Coordination

The first case study had shown the importance of setting up a central point of coordination, which takes responsibility for service definition, application integration, SOA strategy, project solution reviews, and the overall governing process (see 4.2). Given Chinthu's growth and future development plans, the project team persuaded its management to set up a similar body. The so-called System Integration Organization (SIO) was formed during the second phase of the project. It was supposed to assume responsibility for governing the identified services and defining the SOA roadmap and strategy for the organization. Given the organization's instant requirements, the SIO had to focus on integration architecture.

Initially, however, the SIO did not perform as expected. The staff assigned to the unit did not perform their tasks, giving priority to other projects they were involved in. This caused delay. As soon as this became clear, the higher management made sure that these staff members were able to spend more time on the SIO by taking them off other projects. This improved the situation considerably.

5.2.2. Service Identification

Different approaches exist to design SOAs and specifically to identify services. SOMA (Service Oriented Modeling and Architecture) is one of the approaches that is increasingly popular for this purpose. It was not used in this case because the different stakeholders involved in decision-making did not choose to adopt SOMA. In such decisions, the project implementers can advise organizations, but have no control over the final decisions where SOA methodology is concerned.

The team gathered information from staff of all different business departments (those that exchange information and services with the POID/SmID) to identify functionality that could be turned into services, with a view to optimizing their business-orientation. Involved in this process were the business-process analysts, business architects and business designers, who have an intimate knowledge of the business processes and goals, and the software architects and software designers. Together they provided the first building blocks of Chinthu's service portfolio, detailing the names, functionalities and owners of services as well as their potential users. Such a service portfolio had never been put together before for this organization.

At first the team, and other ongoing projects within the organization, had started off randomly identifying and defining services, and governing their delivery. However, it soon became clear that this could lead to service design inconsistency and redundancy in service functionality. In response, the teams started working according to a clearly structured process, led by the SIO. As a result, the identification phase took considerably longer than planned. However, it proved a good trade off, because the structured approach avoided having to rectify things in a later stage.

The project team then proceeded by categorizing the candidate services on the basis of their functionalities. Each candidate service contains business functions which can be invoked by the clients of the service. Each service was grouped into one of the following two categories: *a) Business service:* A service encapsulating business transactional functionalities. This may be a single or a composite service. Invoking a composite service results in triggering other services while these internal invocations are abstracted from other systems (consumers) providing a consistent view; *b) Data service:* A service that provides a standard and collective view of important data entities or master data. This layering of services provided the basis for the new design of the SmID.

5.2.3. Benefits, challenges and best practices

SOA transitions typically involve many different departments or business units. To structure the process and ensure that the new services are business-driven, setting up a central coordination unit is advisable, as was learned in the first case. A challenge in this project was that the SIO staff were insufficiently dedicated to their coordination role, because they had too many other responsibilities to attend to. A best practice identified is that the management should ensure that a committed coordination team can concentrate on its central role in the SOA transition process.

Another best practice is to invest time in conducting a well-structured and centrally managed service identification analysis, involving the SIO. The time investment will pay off because it minimizes the risk of service functionality redundancies and inconsistencies, and maximizes the business-orientation of the services.

5.3. Phase 3: System implementation

After the service identification was finished, design and implementation could start. The existing POID needed to be replaced by the SmID because the design and technology used for the POID did not permit an efficient and cost effective way of integrating it with other systems. This made it difficult for the organization to introduce new business functionalities and enhance its system maintenance effort. Consequently, a new SmID, based on SOA principles, was designed and implemented. This was expected to support more effective inter-system integration and improve the business flow to increase overall business efficiency.

5.3.1. New system design

The SmID was designed using Java Enterprise Edition (JEE) technology and was provided with a web user interface to provide business functions to the SmID end users. In addition to that, the SmID was also meant to provide these same business functions to other systems, for instance, the self service terminals in which people can access the same services without being helped at a counter. Furthermore, the SmID needed to provide data services, which used to be accessed through its database by other systems within the organization.

The SmID architecture thus had to distinguish between two types of services: business services and data services. The business services were incorporated and exposed through the

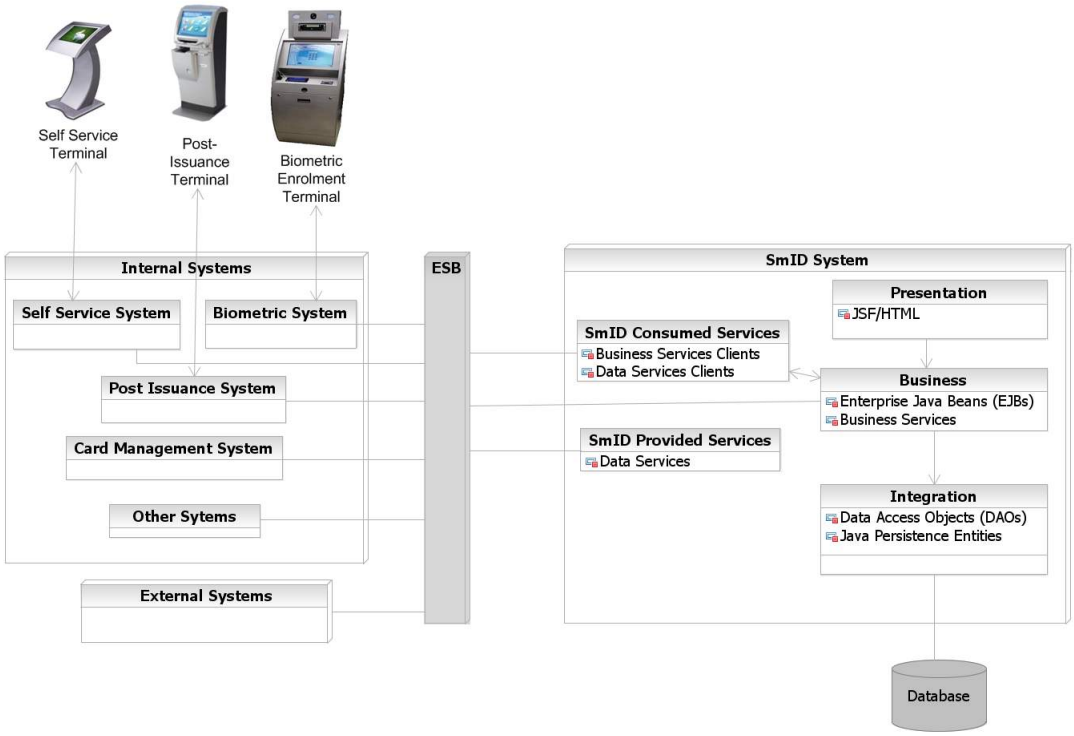

Figure 5. New system architecture

core business layer of the application, while the data services were delivered and packaged separately. This was done, firstly, to avoid jeopardizing the delivery of the core business functions through the SmID and its user interface, and secondly, to avoid interdependency and tight coupling between the provided data services and the SmID core application.

The new SmID is a distributed application classified into a set of layers. Figure 5 illustrates the design and presents the SmID application in its context. The SmID system itself (shown at the right) consists of five separate layers. The rationale behind the layering is that it separates various system responsibilities from one another, which enhances both system development and future maintenance.

- The Presentation layer deals with the presentation logic and is responsible for the screen rendering and the dialogue with the end user.
- The Business layer is where core business functions, transactions, and functionality takes place. The presentation layer requests data and business transactions from this layer.

Other internal and external systems also interact with this layer requesting business functions. This Business layer in turn interacts with the integration layer.
- The Integration layer is responsible for access to the relational database.
- The SmID Consumed Services layer represents the exposed SmID business functions and is responsible for the interaction with other internal systems and external business partner systems. Components in this layer require business logic, thus this layer interacts with the business layer.
- The SmID Provided Services layer is where the data services provided by the SmID are located.

5.3.2. Implementation

The POID was replaced with the SmID to make it possible for those systems that access the POID application and database directly, to be migrated to use the new services provided by SmID. Moreover, the SmID allowed the use of the new services identified and provided by other systems using Chinthu's new SOA strategy for inter-system dataflow and communication.

After the candidate services and their operations were agreed with the different system owners and the SIO staff, their specifications and requirements were detailed in service specification documents. These documents were then handed over to the development teams of the different systems involved, who were responsible for delivering these services.

Complications were caused by the fact that the project had no control over when the other development teams would deliver their services. These teams had shared their delivery planning with the project, but in practice their planning as well as the new services themselves changed repeatedly, due to modifications in operation names, location and functionality. Such changes were not consistently communicated to the project. As a result, developers were often unable to continue with the implementation of the use cases assigned to them, because certain steps in the use cases involved interaction with the other systems. As a result, the developer had to develop their own stubs to proceed with their work. This resulted in a couple of weeks delay.

On the other hand, the fact that a rigorous and comprehensive analysis had been carried out during the identification phase, helped to deal with the high number of change requests that came from the business as the project evolved (due to changing legal frameworks, the introduction of new business products, and changes to business processes). The improvements in coherence of the new system and its interfacing systems, made that the project under study spent considerably less time than estimated on adding the new functionalities to the system.

5.3.3. Scenario

This scenario is an example of how the SmID was implemented and integrated with other systems. The example involves one of the business functions, "apply for ID card", provided by the SmID (see Figure 6) and performed by an end-user. The user enters the applicant's social security number on the main screen for applying for an ID card and submits the request. The SmID system decides whether the applicant's request complies with the specific business rules that are defined for this function. If the request passes certain checks, the system will continue to the payment step, after which it will issue a card for the applicant.

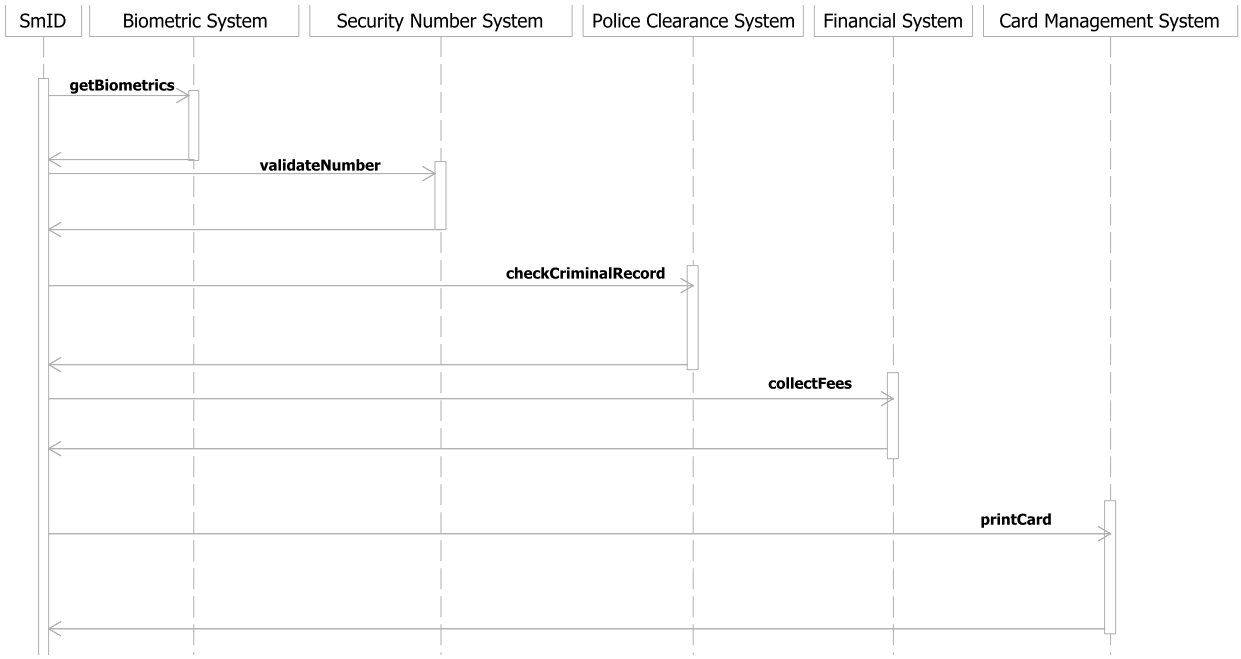

Figure 6. Scenario

This scenario involves several interactions between the SmID and other internal systems. The *Biometric System* retrieves the biometric details of the applicant which are needed for the ID card; the *Security Number System* checks whether the security number is valid; the *Police Clearance System* checks whether the applicant has a criminal record; the *Financial System* collects the administrative and card fees; and finally the *Card Management System* prints the ID card. Information exchange, inquiry and checks are executed during this interaction. This functionality is made available through the specified provided services. The SmID depends on these services in order to complete this internal business function.

5.3.4. Benefits, challenges and best practices

Challenges encountered during the implementation phase were caused by the fact that the project had no control over the delivery of services provided by other systems with which the new SmID was to interact, nor did it control the fact that these services kept changing. This was resolved by using stubs. A best practice that was identified was to separate the core SmID functionality from the data services by means of a layering in the design. This ensured the

continued delivery of the core business functions to the end users through the user interface, and avoided tight coupling between the provided data services and the SmID core application.

6. DISCUSSION

Based on the lessons learned from the two case studies presented, and from the existing literature, in this section, we revisit the research questions posed in section 3. We also put forward some urgent questions for further research.

6.1. Revisiting research questions

Without wanting to repeat all the above mentioned challenges, benefits and best practices, we summarize some of the most important lessons learned throughout the implemented case studies.

6.1.1. How can organizations realize the claimed benefits of SOA and which best practices help to minimize the risks and costs?

Proponents of SOA suggest that a migration to SOA will help organizations to cut costs, increase flexibility and achieve higher productivity by speeding up the introduction of new products and services (see section 3). The case studies presented in this paper do not allow us to draw final conclusions on all these points. The reason is that a full migration to SOA throughout the organization had not been completed at Ndi Moyo or Chinthu at the time of writing this paper. Whether the two organizations have benefited from higher flexibility and/or productivity since they started the SOA process can therefore not yet be ascertained. Assessments as to whether they will enjoy a reduction of total costs for re-engineering and integrating their systems, in order to meet rapidly changing business demands, cannot be made yet. However, as a result of the SOA process, Ndi Moyo and Chinthu have gained better insight into, and control over, the expected costs of software evolution and maintenance. Furthermore, in the second case, as a result of designing and implementing the new system according to SOA principles, the organization now has, for the first time, a proper and comprehensive overview of this internal system and its many interactions with other systems. Moreover, a first indication of the flexibility of this system was given by the fact that new requirements and change requests were easily integrated. This flexibility may bring down the running and maintenance costs of the system in the long run.

While achieving these first positive results it became clear that SOA projects are at risk of experiencing delay, or even failure, due to the fact that many different stakeholders are involved in the implementation of these projects. The parties responsible for the service consumer and provider applications depend on each other for delivering their part of the solution. This mutual dependency can cause delays in delivering the overall solution, as was experienced in the second case study. Therefore, establishing efficient coordination between all stakeholders needs major attention from the very start of any SOA project. However, to tackle this risk at its root, methodologies and techniques should be developed that reduce the mutual technical

dependencies of the different service oriented applications. This was beyond the scope of the two case studies and is an issue to be taken up by future research.

The following best practices were derived from the two case studies.

Firstly, both case studies showed that it is highly recommendable to take an incremental approach when adopting SOA. In the first case of Ndi Moyo, the transition was carried out in three phases, while in the second case of Chinthu it was also decided to start with one system at the time. This strategy gave the various stakeholders the opportunity to get acquainted with the implications of the SOA transition. The proof-of- concept (PoC) phase in case one, which involved the design of a visually appealing portal, helped to make the potential benefits of SOA tangible for decision makers, most of whom do not have a technical background.

Allowing enough time at the beginning of a project to introduce the SOA concept to all stakeholders involved, will pay off in the long-term because a SOA transition requires the continued shared commitment of the whole organization. This strategy was also successfully used in the second case study, where participatory workshops resulted in a cooperative attitude from IT and management staff. Conducting a PoC also has an important technical function: it verifies whether the specific technology solutions can operate successfully within an organization's IT landscape. The most important benefits of a phased approach are that it enables organizations to spread and control the required investments for the SOA transition and that it allows this transition to happen incrementally while core business can go on as usual.

Secondly, both case studies showed that setting up a central coordination body - ICoC in the first case, SIO in the second case - is beneficial to the SOA transition process. In the first case, ICoC developed a structured methodology to estimate the scope and costs of individual service integrations. This allowed Ndi Moyo to make informed go or no-go decisions at an early stage of the proposed integrations. The second case study presented in this paper similarly benefited from setting up a central coordination unit (SIO), but only after the management intervened to make sure that the staff assigned to SIO was given sufficient time to perform its coordinating role.

The first case study showed that it is crucial that a unit such as ICoC is an independent body. If business units that have to work with the coordinating body are not assured of this, they may not want to give away some of the responsibility for controlling and managing their business services to that third party. The work process methodology that ICoC devised actively involved the business units in various steps of decision making. This helped to foster trust in, and ownership of, the SOA transition.

Thirdly, the second case study showed that it is recommendable to invest sufficient time in the initial analysis and identification phase. Conducting a well-structured and centrally managed service identification analysis, involving the coordination unit, will pay off because it minimizes the risk of service functionality redundancies and inconsistencies. Moreover, it is of great benefit during the implementation phase, when implementing new functionalities can be handled efficiently and does not require high resource costs of senior personnel. In the second case of Chinthu this was particularly significant because there happened to be a high number of change requests from the business as the project evolved. Thanks to the fact that the new system and its interfacing systems were now loosely coupled, these change requests were dealt with in less time than anticipated.

Fourthly, the second case study showed that it is wise for organizations not to dedicate the majority of their IT budget to the infrastructure (middleware) from the very start. Instead, as was done in the second case, focusing on both applications and infrastructure helps to avoid unnecessary upfront investments, and ensures a better tailored infrastructure.

Lastly, a best practice derived from the second case is to use a strategy of layering in the design of SOA based applications. Usually applications provide business functions to a specific group of end users. In order to guarantee undisrupted provision of these functions, separating core functionality from new services is a method system designers of SOA applications might well consider.

6.1.2. Which factors complicate implementation of and transition to SOA in practice?

An important lesson learned from both case studies is that implementing SOA is not only a complicated technical endeavor, but also presents challenging issues of organizational structure and capacity, including the question of ownership. Many people have an inclination to resist change. In the second case, the IT staff of various departments initially were reluctant to participate and share their knowledge of the current systems, which was needed for the analysis of the planned transition. They were worried that their expertise may become redundant due to the introduction of SOA.

Therefore, it is very important to involve, from the very start of a SOA transition, not only all different business units, but also the technical IT staff who will, eventually, be responsible for the integration. The efforts needed to get people on board and to ensure sufficient training of personnel for the use of tools and development should not be underestimated [6].

Secondly, the second case showed that a practical challenge that organizations may face is that a majority of their systems are not mature enough to expose their business functions as services without first being re-engineered according to SOA principles. If this turns out to be the case, organizations will have to make considerable investments for re-engineering these systems.

Thirdly, a complication that derives from the general risk of SOA projects mentioned above (i.e. the mutual dependency of stakeholders) is that delays in development and delivery can happen because the consuming and providing systems have no control over each other. During SOA transitions one typically deals with systems that both provide and consume services to and from other systems. The second case, in particular, showed that delays and unreported changes can hinder development progress.

Fourthly, the rationale of migrating to SOA is to be able to manage multiple integrations at the same time. However, this presents testing challenges. Because, while testing single new integrations is not very complicated, doing so for multiple integrations at the same time is daunting. Methods and tools for testing the effects that one change can have on the overall system of systems are not yet available. This can jeopardize the benefits of SOA, especially when negative impacts of changes are discovered during runtime.

6.2. Research Directions

We fully support the directions for further research as suggested in recent literature [15, 16, 22, 24]. In this paper we emphasize three important areas for future research that follow from the challenges identified and the lessons learned through our case studies.

6.2.1. How is impact measured?

Most of the SOA transitions, migrations and implementations that are carried out today claim to be successful, and vendors publicize this. The experiences reported in this paper also suggests that the SOA transitions performed were successful, because they proved technically feasible and the clients were "happy" and decided to continue with the further implementation. However, we need to ask ourselves what so-called "satisfactory" or "acceptable" results really mean? We believe that a much clearer distinction should be made between outcome and impact. A SOA transition can be (technically) successful, but is that necessarily an indication of achieved business goals?

Measuring ROI in software engineering is a delicate problem in general, discussed, for example, in the realm of software process improvement by Van Solingen [33]. As for the specific case of SOA, it has by now achieved a certain level of technical maturity, which requires similar investments in analyzing its true benefits for business. Much more rigorous research should be done, and tested in real-life situations, to establish whether organizations that made the transition to SOA are really better off, in the medium and long term, on the various aspects that SOA promised to deliver (increased flexibility, reduced costs, increased productivity, etc). While there were strong indications that flexibility and cost reduction were indeed achieved in the second case, models for quantifying these benefits should be developed. The same applies to the issue of increased productivity. Independent researchers and organizations should collaborate to develop this important research direction.

6.2.2. Automated runtime engineering

The industry reports showed that an actual transition towards SOA is not a simple and straightforward exercise. The fact that SOA no longer deals with autonomous systems, but with systems of systems that together display a much higher degree of complexity, means that some of the traditional engineering methods for testing and the composition of services may no longer be suitable. SOA challenges us to execute and perform more of the engineering activities in an automated way and during runtime. New process models and product models need to be developed to enable this.

Tools and methodologies to reduce time and human resources have to be developed. Global costs and risks of SOA migration processes need to be assessed through case studies, in a similar way that this has been done for migrating legacy systems to more modern systems [26].

One area to which this applies in particular, and which is still under-explored, is testing. While SOA realizes the technological foundations for runtime reconfiguration and maintenance, adequate methods, techniques and tools for dealing with runtime integration and testing

are still lacking. We are currently involved in an ongoing research project that investigates the primary industry challenges of runtime integration and testing. We believe that that runtime evolution and testing of SOA can only be implemented successfully if, especially, the communication between stakeholders, and the test-isolation and test-awareness of services are improved [11].

6.2.3. Empirical research

Finally, we urge that more case studies are conducted in order to achieve progress in the understanding and improvement of SOA development and implementation. This requires more collaboration between industry and independent research [15, 32]. If researchers are closely involved with industry, this allows them first-hand observation and investigation of the problems that occur during the SOA development and migration. This will help to identify the most urgent practical bottlenecks that need to be addressed in academic research, as well as empirically establish successes and best practices which can help future research.

6.3. Threats to Validity

The main threats to the validity of the results of a descriptive case study are concerned with repeatability (reliability validity) and generalizability (external validity) [36]. With respect to reliability, the confidential nature of the case is a clear threat. This is inherent to industrial SOA implementations, and hard to avoid. Moreover, the level of control in these kind of case studies - especially because it involves a third party - will always be lower than in an independent experiment [34]. The potential threat to internal validity due to the first author's involvement in these projects is countered by the fact that as a third party implementer of the solutions, he had no direct business interest in its success or failure.

With respect to external validity, it is clear that more empirical studies will have to be designed and executed to extend the validity of the findings of both case studies presented here. The fact that the second case study gave the opportunity to test some of the lessons learned in the first case study, is a first step.

A strong point of industrial case studies like these ones is that they avoid scalability problems [34]. It can be safely said that organizations of similar scale as the ones studied here, will present similar challenges and results. Moreover, the results are unlikely to be domain related - as we found that similar challenges and benefits applied to the transport (commercial) and public sector organizations studied here.

A possible threat to the external validity of the first case study is the fact that it is based on only one actual new SOA integration. In reality, several such integrations were executed, and these did not raise new challenges. Nevertheless, it is likely that if all 700 applications were to be integrated, this would lead to unexpected problems, either organizational (because of the number of people involved) or technical. As indicated before, testing in such a complex context may also pose new challenges.

7. CONCLUSIONS

In this paper, we report on our experiences with conducting two transitions from non-SOA multi-point integration to re-engineered SOA infrastructure, in both the transport and logistics domain, and the public sector. In particular, we described the first case, that involves a proof-of-concept web portal implementation, and the setting up of an Integration Center of Competence, plus the integration of a new service into the existing application portfolio. In the second case, we have applied some of the lessons learned from the first case, and presented extended analysis and specification phases, the setting up of a System Integration Organization, and the implementation of a complete sub-system as SOA. We consider the following items as our key contributions:

- The experimental design of our case studies, permitting us and others to repeat similar case studies, as here case two built on the findings of case one;
- The reports of the cases conducted;
- The identification within these cases of (1) actually achieved benefits; (2) best practices that helped to achieve the success; (3) challenges that require further attention;
- The identification of research directions addressing the challenges encountered.

Overall, we conclude that SOA has beneficial effects on an organization's IT landscape, in particular in terms of improved flexibility and, thus, productivity of the participating systems, if the transition is carried out carefully and all stakeholders are involved properly. We have proposed a number of "best practices" that help alleviate the transition to SOA. However, through the short-term nature of the two cases presented, conclusions on the overall return of investment on migrating to SOA are difficult to draw based on our experience with conducting the transitions.

Several avenues for future research have been sketched in Section 6.2. One issue in particular that our future research will focus on is to develop methods and techniques that can reduce the mutual dependency between stakeholders and applications, which currently complicates SOA. Another, related challenge that we will address in the near future is the development of test methods targeting SOA-specific faults, related to, e.g., service publication, discovery, binding, and composition.

REFERENCES

1. Cesare Bartolini, Antonia Bertolino, Sebastian Elbaum, and Eda Marchetti. Whitening soa testing. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, ESEC/FSE '09, pages 161–170, New York, NY, USA, 2009. ACM.
2. Norbert Bieberstein, Sanjay Bose, Marc Fiammante, Keith Jones, and Rawn Shah. *Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.
3. Gerardo Canfora and Massimiliano Di Penta. Testing services and service-centric systems: Challenges and opportunities. *IT Professional*, 8(2):10–17, 2006.
4. Gerardo Canfora, Anna Rita Fasolino, Gianni Frattolillo, and Porfirio Tramontana. A wrapping approach for migrating legacy system interactive functionalities to service oriented architectures. *Journal of Systems*

and Software, 81(4):463 – 480, 2008. Selected papers from the 10th Conference on Software Maintenance and Reengineering (CSMR 2006).
5. Gerardo Canfora and Massimiliano Penta. Software engineering. chapter Service-Oriented Architectures Testing: A Survey, pages 78–105. Springer-Verlag, Berlin, Heidelberg, 2009.
6. S. Cetin, N. Ilker Altintas, H. Oguztuzun, A.H. Dogru, O. Tufekci, and S. Suloglu. Legacy migration to service-oriented computing with mashups. In *Software Engineering Advances, 2007. ICSEA 2007. International Conference on*, pages 21–21. IEEE Computer Society, Aug. 2007.
7. K. Channabasavaiah, K. Holley, and E. M.Tuggle. Migrating to a service-oriented architecture. *IBM White paper*, 2004. `http://www.ibm.com/developerworks/library/ws-migratesoa/`.
8. Félix Cuadrado, Boni García, Juan C. Dueñas, and Hugo A. Parada. A case study on software evolution towards service-oriented architecture. In *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops*, pages 1399–1404, Washington, DC, USA, 2008. IEEE Computer Society.
9. Schahram Dustdar and Stephan Haslinger. Testing of service-oriented architectures – a practical approach. In *Object-Oriented and Internet-Based Technologies, Net.ObjectDays*, volume 3263 of *Lecture Notes in Computer Science*, pages 97–109. Springer, 2004.
10. V. Gehlot and G. Pujari. A case study in defining colored petri nets based model driven development of enterprise service oriented architectures. In *System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on*, pages 1 –9, 2009.
11. Michaela Greiler, Hans-Gerhard Gross, and Khalid Adam Nasr. Runtime integration and testing for highly dynamic service oriented ict solutions an industry challenges report. In *TAIC-PART '09: Proceedings of the Testing: Academic & Industrial Conference on Practice And Research Techniques*. IEEE, 2009.
12. M.N. Huhns and M.P. Singh. Service-oriented computing: key concepts and principles. *Internet Computing, IEEE*, 9(1):75–81, Jan-Feb 2005.
13. Mira Kajko-Mattsson, Grace A. Lewis, and Dennis B. Smith. A framework for roles for development, evolution and maintenance of soa-based systems. In *SDSOA '07: Proceedings of the International Workshop on Systems Development in SOA Environments*, page 7, Washington, DC, USA, 2007. IEEE Computer Society.
14. Timo Kokko, Jari Antikainen, and Tarja Systä. Adopting SOA - experiences from nine finnish organizations. In *CSMR '09: Proceedings of the 2009 European Conference on Software Maintenance and Reengineering*, pages 129–138, Washington, DC, USA, 2009. IEEE Computer Society.
15. Kostas Kontogiannis, Grace A. Lewis, and Dennis B. Smith. A research agenda for service-oriented architecture. In *SDSOA '08: Proceedings of the 2nd international workshop on Systems development in SOA environments*, pages 1–6, New York, NY, USA, 2008. ACM.
16. Kostas Kontogiannis, Grace A. Lewis, Dennis B. Smith, Marin Litoiu, Hausi Muller, Stefan Schuster, and Eleni Stroulia. The landscape of service-oriented systems: A research perspective. In *SDSOA '07: Proceedings of the International Workshop on Systems Development in SOA Environments*, page 1, Washington, DC, USA, 2007. IEEE Computer Society.
17. N. Kulkarni and V. Dwivedi. The role of service granularity in a successful soa realization a case study. In *Services - Part I, 2008. IEEE Congress on*, pages 423–430, July 2008.
18. G. Lewis, E. Morris, and D. Smith. Service-oriented migration and reuse technique (SMART). In *Software Technology and Engineering Practice (STEP), 2005. 13th IEEE International Workshop on*, pages 222–229. IEEE Computer Society, 2005.
19. G. Lewis, E. Morris, and D. Smith. Analyzing the reuse potential of migrating legacy components to a service-oriented architecture. *Software Maintenance and Reengineering, 2006. CSMR 2006. Proceedings of the 10th European Conference on*, pages 9–23, 2006.
20. Grace A. Lewis, Edwin Morris, Soumya Simanta, and Lutz Wrage. Common misconceptions about service-oriented architecture. In *ICCBSS '07: Proceedings of the Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems*, pages 123–130, Washington, DC, USA, 2007. IEEE Computer Society.
21. M. Lormans, H. van Dijk, A. van Deursen, E. Nöcker, and A. de Zeeuw. Managing evolving requirements in an outsourcing context: an industrial experience report. In *Software Evolution, 2004. Proceedings. 7th International Workshop on Principles of*, pages 149–158, Sept. 2004.
22. Zaigham Mahmood. The promise and limitations of service oriented architecture. *International Journal of Computers*, 1(3):74–78, 2007.
23. Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, Frank Leymann, and Bernd J. Krämer. 05462 service-oriented computing: A research roadmap. In Francisco Cubera, Bernd J. Krämer, and Michael P. Papazoglou, editors, *Service Oriented Computing (SOC)*, number 05462 in Dagstuhl Seminar Proceedings,

Dagstuhl, Germany, 2006. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.
24. Mike P. Papazoglou and Willem-Jan Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16:389–415, July 2007.
25. Colin Potts. Software engineering research revisited. *IEEE Software*, 10(5):19–28, September 1993.
26. Maseud Rahgozar and Farhad Oroumchian. An effective strategy for legacy systems evolution. *Journal of Software Maintenance*, 15(5):325–344, 2003.
27. Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.
28. Derek T. Sanders, J. A. Hamilton, Jr., and Richard A. MacDonald. Supporting a service-oriented architecture. In *SpringSim '08: Proceedings of the 2008 Spring simulation multiconference*, pages 325–334, San Diego, CA, USA, 2008. The Society for Computer Simulation, International.
29. J. Schelp and S. Aier. SOA and EA - sustainable contributions for increasing corporate agility. In *42nd Hawaii International Conference on System Sciences (HICSS'09)*, pages 1–8. IEEE Computer Society, 2009.
30. Harry Sneed. A pilot project for migrating cobol code to web services. *International Journal on Software Tools for Technology Transfer (STTT)*, 11:441–451, 2009. 10.1007/s10009-009-0128-z.
31. Wei-Tek Tsai, Xinyu Zhou, Yinong Chen, and Xiaoying Bai. On testing and evaluating service-oriented software. *Computer*, 41(8):40–46, 2008.
32. Arie van Deursen, Paul Klint, and Chris Verhoef. Research issues in the renovation of legacy systems. In J.-P. Finance, editor, *Fundamental Approaches to Software Engineering '99*, Lecture Notes in Computer Science, pages 1–23. Springer, 1999.
33. Rini van Solingen. Measuring the ROI of software process improvement. *IEEE Software*, 21(3):32–38, 2004.
34. Claes Wohlin, Per Runeson, Martin Host, Magnus C. Ohlsson, Bjorn Regnell, and Anders Wesslen. *Experimentation in software engineering: An introduction*. Kluwer Academic Publishers, 2000.
35. I. Wong-Bushby, R. Egan, and C. Isaacson. A case study in soa and re-architecture at company abc. In *System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on*, pages 1–8. IEEE Computer Society, 2006. Track 8.
36. R. K. Yin. *Case Study Research: Design and Methods*. SAGE Publications Inc, 3d edition, 2003.
37. Uwe Zdun, Carsten Hentrich, and Schahram Dustdar. Modeling process-driven and service-oriented architectures using patterns and pattern primitives. *ACM Trans. Web*, 1, September 2007.
38. Jia Zhang, Jen-Yao Chung, and Carl K. Chang. Migration to web services oriented architecture: a case study. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1624–1628, New York, NY, USA, 2004. ACM.

TUD-SERG-2011-004
ISSN 1872-5392