Pattern-based Process for a Legacy to SOA Modernization Roadmap

FRANK J. FREY, Münster
CARSTEN HENTRICH, PricewaterhouseCoopers AG WPG, Frankfurt/Main
UWE ZDUN, Software Architecture Research Group, University of Vienna

Many organizations still operate their business based on old legacy systems which cause high maintenance costs and are difficult to change. After the hype about service-oriented architectures (SOAs) was gone, they were implemented successfully and helped replacing old legacy systems with flexible services, for example at large companies in the finance or logistics industry. While recent research and the pattern literature cover the technical aspects of SOA and the alignment between business and IT, defining and planning a roadmap for a SOA modernization program, which is essential for initiating such a program and getting the necessary commitment, has not been in the focus of research yet. Thus, we propose a design process for the definition of a high-level roadmap. It considers determining architectural decisions and planning decisions by selecting appropriate patterns and reference architectures. Furthermore we define a heuristic for the roadmap planning activities, because we see a lack of appropriate planning patterns. A case study explains how the roadmap design process has been applied in a large-scale industry project concerning a SOA modernization program of a global logistics provider. The pattern-based process should motivate pursuing further research about roadmap design processes and patterns and provide guidance for industry experts.

Categories and Subject Descriptors: D.2.11 [Software Architectures] Patterns

General Terms: Design

Additional Key Words and Phrases: SOA, architectural decision, patterns, roadmap, business capabilities, software architecture

1 INTRODUCTION

Despite the ongoing trend to organize IT in a service-oriented way, many organizations still have monolithic legacy systems, which are critical for their business operations (Khadka et al., 2012). If these legacy systems cause serious problems, like high maintenance costs or time-lags for introducing new features or products, many organizations start initiatives for replacing old legacy applications and transforming their IT into a Service-Oriented Architecture (SOA) (Maréchaux, 2008). Often the current application architecture entails an inherent complexity which has grown over decades, by organic growth or by mergers and acquisitions, and comes along with a slow reaction to business needs. In the finance industry, for example, core banking functions are often still based on 20+ years-old (mainframe) legacy systems. A recent study (Rasch and Billeb, 2013) from PricewaterhouseCoopers, a leading auditing and consultancy company, points out that a big share of the IT of financial service providers is still based on old legacy systems. The study also claims that there is a strong consolidation need regarding the system architecture and that the objectives and the roadmaps must be clearly defined for the implementation of the modernization programs.

Determining the required work packages, projects and dependencies for a large-scale SOA modernization has an inherent complexity. Consequently, planning and aligning the projects on a roadmap requires a structured approach. Defining roadmaps for SOA modernization programs has not been in the center of research in the recent years, however. In addition to technical aspects of SOAs a lot of research studies

Author's address: Frank J. Frey, Vivaldistr. 6, 48147 Münster, Germany, frank.frey@me.com. Carsten Hentrich, PricewaterhouseCoopers AG WPG, Friedrich-Ebert-Anlage 35-37, 60327 Frankfurt/Main, Germany, carsten.hentrich@de.pwc.com. Uwe Zdun, Software Architecture Research Group, University of Vienna, Vienna, Austria, uwe.zdun@univie.ac.at.

and patterns have been published concerning service identification and business-IT-alignment[1]. For example, the CAPABILITY-BASED SERVICE pattern (Frey et al., 2014) helps identifying SOA services based on a model of business capabilities. Other methods and patterns consider business processes as driving factors. Fareghzadeh, for example, suggests breaking-down business processes for identifying services top-down (Fareghzadeh, 2008). Hentrich and Zdun introduce a comprehensive approach for developing process-driven SOAs, which entails patterns for aligning business and IT (Hentrich and Zdun, 2012). Buckl et al. describe four methodology patterns concerning the planning of application landscapes in their Enterprise Architecture Management (EAM) Pattern Catalog. While these patterns and pattern languages provide guidance to service identification, business-IT-alignment, and technical aspects of SOAs, these patterns do not concern the actual planning process concerning the definition of a modernization roadmap.

In this paper we propose a pattern-based design process for the definition of SOA roadmaps. It is inspired by the architectural design method for software architecture, ArchPad, which combines architectural decision models and patterns (Zimmermann et al., 2008). That is, in ArchPad architectural decisions are used to determine and select appropriate patterns. Further details on the ArchPad design method are described in Section 2. In our opinion ArchPad is focused on the technical aspects of architectural design and it does not, particularly, consider defining and planning service development projects. Both are key aspects for defining a SOA roadmap. Hence, we defined a specific process for designing SOA roadmaps in the context of large-scale SOA modernization programs. Our process, described in Section 3, consists of five stages. It begins with setting the direction for the whole modernization program in Stage 1 up to the multi-project planning of a roadmap. Each stage refers to a different set of design decisions, which answer major design questions and use corresponding patterns or reference architecture. The roadmap process refers to a high-level definition of a SOA roadmap, which is supposed to help defining the scope, as well as discussing and initiating the SOA program in a preliminary phase with stakeholders from business and IT. The proposed design process has been applied in an industry project concerning an SOA modernization program for a global logistics provider. This case study is described in Section 4. Finally, Section 5 concludes with a summary and outlook on further work.

2 COMBINING ARCHITECTURAL PATTERNS AND DECISIONS

Recent research about software architecture considers architectural (design) decisions as a vital element of software architecture. Jansen and Bosch, for example, propose “to view a software architecture as a set of explicit architectural design decisions” or rather the results of preceding architectural design decisions[2] (Jansen and Bosch, 2005). As a working definition, we consider an architectural decision as a selection among alternative choices concerning the design of an architectural building block[3] of architecture (i.e. referring to enterprise architecture as well as the architecture of a piece of software).

Following publications discuss how architectural decisions (AD) are supported by patterns or can be combined with patterns. For example, (Harrison et al., 2007) introduced this notion, (Zimmermann et al., 2008) included it in the aforementioned ArchPad design method, and That et al. proposed documenting architectural decisions with the help of formalized patterns (That et al., 2012).

The ArchPad design method (Zimmermann et al., 2008) considers the design process as a sequence of stages, which require architectural decisions on different levels and from different stakeholders. Figure 1 shows the process. It indicates that executive decisions are required for setting the primary objectives and directions in the first stage, called “Forming”. The following stages require decisions about the conceptual

[1] Please refer to Khadka et al. (Khadka et al., 2012) for a comprehensive literature review about legacy to SOA evolution.

[2] Jansen and Bosch define an architectural design decision as a “description of the set of architectural additions, subtractions and modifications to the software architecture, the rationale, and the design rules, design constraints and additional requirements that (partially) realize one or more requirements on a given architecture” (Jansen and Bosch, 2005).

[3] Please refer to The Open Group Architecture Framework (TOGAF, 2011) concerncing the definition and usage of an architectural building block.

and technology architecture as well as about the selection of appropriate vendors (and their products). For each stage ADs are expressed by selecting a set of suitable architectural patterns. An architectural pattern refers to “recurring solutions that solve problems at the architectural design level, and provide a common vocabulary in order to facilitate communication” (Avgeriou & Zdun, 2005).

For example, the first stage (“Forming”) requires selecting appropriate business patterns, while the second stage (“Storming”) refers to architectural patterns on a conceptual level. According to Zimmermann et al. defining ADs by selecting appropriate patterns brings the following benefits: Each AD refers to a pattern. Hence, the description of the AD can be shortened because it refers to description of the pattern. Instead of a comprehensive description of the context, forces, problems and solutions, it is supposed to be sufficient describing how the pattern is adopted for this particular AD. Furthermore domain-specific decision models may guide practitioners through the whole design process, including the pattern selection.

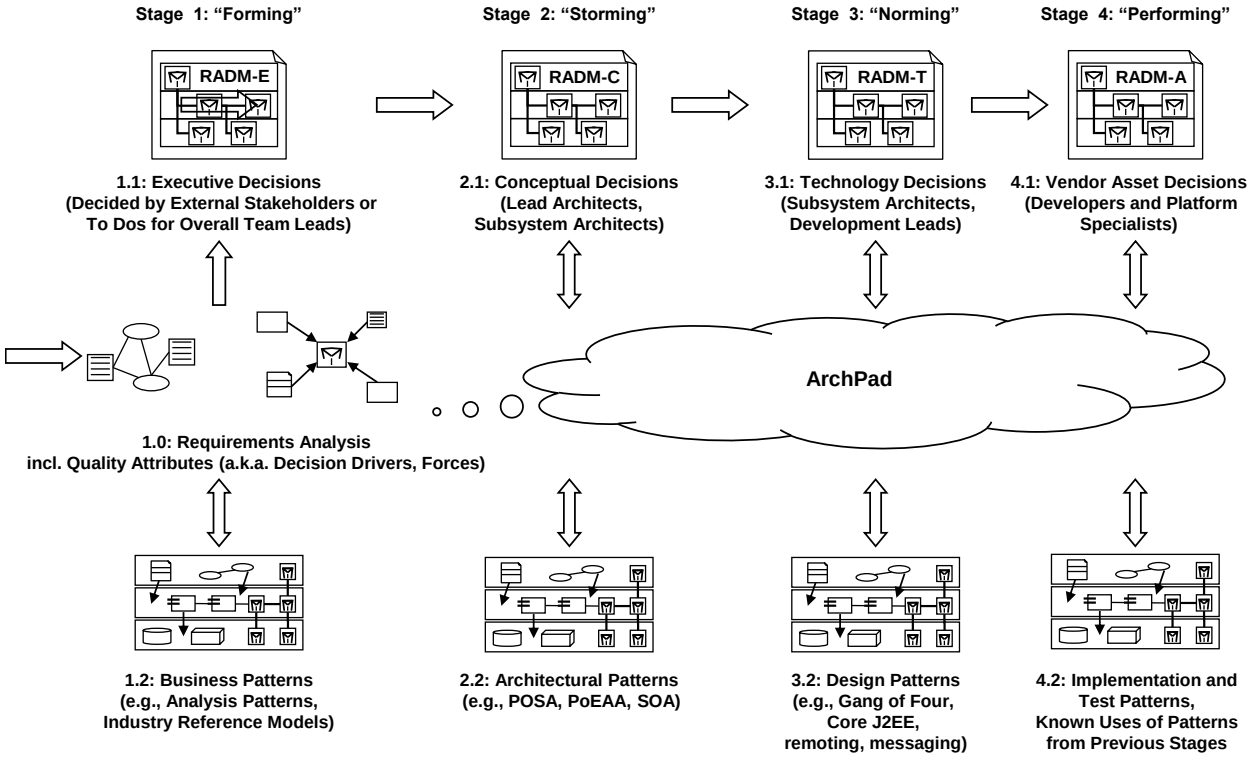

Figure 1: ArchPad Design Method for Software Architecture (from Zimmermann et al., 2008)

3 DESIGN PROCESS FOR SOA ROADMAP DEFINITION

3.1 Overview

We propose a process for the definition of a legacy to SOA modernization roadmap following the ArchPad design method. According to the ArchPad design method each decision refers to a selection of an appropriate pattern or reference architecture, considering different stages of a design process. The ArchPad method organize Stages top-down, beginning with executive decisions, then conceptual and technical decisions up to vendor asset decisions. Applying the ArchPad method, the results are focused on technical aspects, concerning architecture design and implementation of software. Organizational matters, like identifying and planning required development projects, are not covered by ArchPad.

Our proposed process follows a top-down approach similar to ArchPad, but it is more specific, as it focusses on the purpose of legacy to SOA modernization. ArchPad does not particularly address the concerns of consolidating and replacing outdated applications by services. Furthermore the outcome of our

design process is focused on the definition of a high-level roadmap rather the technical implementation, which is a major concern of ArchPad's final stage ("Performing").

Thus, we propose an adapted design process with focus on the roadmap definition as major concern and final outcome of the process. Consequently, this design process does not only focus on ADs but also on decisions about organizational aspects, like project timelines and project organization. Nevertheless, we follow Zimmermann et al.'s concept for ArchPad that decisions "capture selected design options with their strengths and weaknesses, as well as justifications for the selections (Zimmermann et al., 2008). While ArchPad considers the selection of patterns for documenting a design decision, we also consider the selection of a reference architecture for our SOA roadmap design process. Industry-specific reference architectures can be particularly helpful (and speed up the process) for identifying relevant functionality and corresponding business services as well as a set of architectural patterns relevant for a specific domain.

As a working definition for reference architecture we consider Kruchten's definition[4] (Kruchten, 2000) and adapt it as follows:

A reference architecture is, in essence, a predefined architectural pattern, or set of patterns, designed, and proven for use in particular business and technical domains, together with supporting artifacts to enable their instantiations (i.e. solution architecture) in specific contexts.

Our proposed process for the SOA roadmap definition, depicted in Figure 2, contains the following five stages and corresponding decisions:

1. Setting Direction
2. SOA Architecture Definition
3. Service Identification
4. Service Design
5. SOA Roadmap Planning

The first two Stages are based on the first stages of ArchPad, but consider specific executive decisions and conceptual directives concerning the context of legacy to SOA modernization. The Stages 3-5 deviate from ArchPad in several respects, relating to identifying and designing services on a high-level and leading to the definition of the roadmap definition rather than the technical implementation.

[4] Kruchten defines a reference architecture as "in essence, a predefined architectural pattern, or set of patterns, possibly partially or completely instantiated, designed, and proven for use in particular business and technical contexts, together with supporting artifacts to enable their use. Often, these artifacts are harvested from previous projects." (Kruchten, 2000). From our point of view, a reference architecture has the character of a template or blueprint, which needs additional activities to be instantiated. Thus, for our working definition we excluded the part "possibly partially or completely instantiated" of Kruchten' definition. Instead, we explicitly mention the support for enabling the instantiation (instead of use) for particular contexts. Furthermore we use the term 'domains' (instead of contexts) referring to domain-driven design introduced by Eric Evans (Evans, 2003).

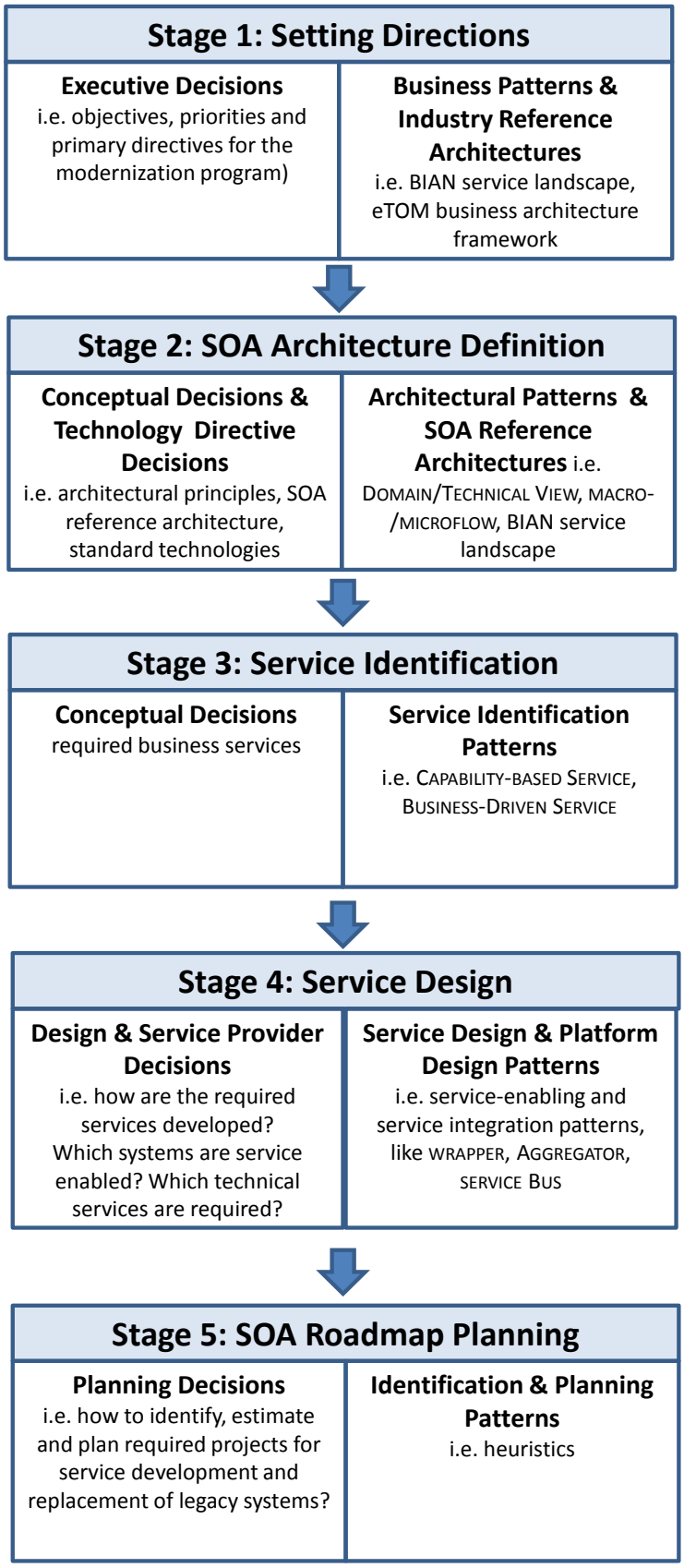

Figure 2: Design Process for SOA Roadmap Definition

Table 1 provides an overview of relevant decisions and exemplary decision alternatives for each stage. It is inspired by the Reusable Architectural Decision Models (RADM) for SOA from the ArchPad paper (Zimmermann et al., 2008). But in this case decision alternatives refer to patterns, reference architectures as well as heuristics. The listed decisions and decision alternative are based on SOA literature, for example, concerning process-driven SOA (Hentrich and Zdun, 2012) or TOGAF's reference architecture for SOA (Open Group, 2011). Furthermore the authors brought in their practical experience from industry SOA projects for identifying relevant decisions and decision alternatives.

Table 1: Pattern Categories per Stage

Stage	Decision Type	Decision	Decision Alternatives (Exemplary)
Stage 1: Setting Directions	Executive Decisions (ED)	ED1: business objectives & priorities	i.e. reduction of IT maintenance costs; new functionality for process innovation or new markets / products
		ED2: IT transformation paradigm	i.e. evolutionary modernization vs. disruptive change; industry reference architecture BIAN (banking); industry reference architecture eTOM (telco)
Stage 2: SOA Architecture Definition	Conceptual Decisions (CD)	CD1: SOA reference architecture	SOA reference architecture (i.e. including definition of SOA layers, architectural principles, service types, architectural building blocks), i.e. Open Group's SOA reference architecture (Open Group, 2011).
		CD2: process / service integration paradigm	i.e. process-driven SOA / process integration patterns, i.e. i.e. DOMAIN-/TECHNICAL-VIEW, MACRO-/MICROFLOW (Hentrich and Zdun, 2012) event-driven architecture patterns, i.e. EVENT-BASED PROCESS INSTANTIATOR, EVENT-BASED ACTIVITY (Köllmann and Hentrich, 2007)
		CD3: routing paradigm	routing patterns, i.e. DYNAMIC ROUTER, AGGREGATOR (Ciurana, 2008)
		CD4: interaction paradigm	interaction patterns, i.e. FIRE EVENT ACTIVITY, ASYNCHRONOUS SUB-PROCESS SERVICE, MULTIPLE ASYNCHRONOUS RESULTS SERVICE (Hentrich and Zdun, 2008)
		CD5: Data transformation paradigm	data / business object transformation patterns, i.e. INTEGRATED BUSINESS OBJECT MODEL (Hentrich and Zdun, 2012)
	Technology Directive Decisions (TDD)	TDD1: high-level technology stack	i.e. basic technologies, like Java / JEE, .NET platform
Stage 3: Service Identification	Conceptual Decisions (CD)	CD6: service identification paradigm	top-down approach, i.e. BUSINESS-DRIVEN SERVICE (Hentrich & Zdun, 2006) or CAPABILITY-BASED SERVICE (Frey et al., 2014) or bottom-up approach; service models of reference architecture, i.e. BIAN, eTOM
Stage 4: Service Design	Design Decisions (DD)	DD1: service design paradigm	i.e. build new service components vs. patterns concerning re-use of existing applications, like the WRAPPER (Ciurana, 2008) or reengineering strategies (OpenGroup 2012)

Stage	Decision Type	Decision	Decision Alternatives (Exemplary)
	Service Provider Decisions (SSD)	SSD: service provider selection	heuristics or patterns for prioritizing potential service providers, i.e. based on estimated development effort or based on the Total Cost of Ownership (TCO) for the future service (including maintenance and support costs)
Stage 5: SOA Roadmap Planning	Planning Decisions (PD)	PD: program organization	i.e. program-wide coordination (multi-project / program-wide coordination of projects) vs. decentral project coordination (separate project organizations for each project); heuristics (or patterns) concerning the identification of relevant work streams and the initiation of a program
		PD: project identification	heuristics (or patterns) concerning identifying and defining relevant projects, i.e. based on the the service provider selection
		PD: (multi-) project planning paradigm	heuristics (or patterns) concerning the planning and visualization of multiple projects on a roadmap

3.2 Stage 1 – Setting Directions

The first stage, Setting Directions, refers to major executive decisions about the objectives and priorities of the SOA modernization program, which create the cornerstones for the design and execution of the program. Considering outdated legacy systems as a starting point, the primary directive for the modernization program may point to either reducing IT maintenance costs or to providing new functionality as soon as possible. Additionally, more detailed business priorities may be defined by executive stakeholders, for example new or consolidated functionality (in form of services) may be required as first priority in order to start a marketing campaign with new products. Furthermore general directions may be given concerning the consideration (and selection) of modernization strategies or industry reference architectures. For example, a financial service company may decide applying the service landscape of the Banking Industry Architecture Network (BIAN) as a reference for the design of its SOA architecture and services (BIAN, 2014). Instead, a telecom provider may select the business process framework eTOM of the tmforum (eTOM, 2014), a global association of telecom providers and suppliers, as reference for designing its SOA architecture.

3.3 Stage 2 – SOA Architecture Definition

Considering the directives from the first stage, the SOA target architecture is defined on a high-level in the second stage. Conceptual and technology decisions are required, like selecting a SOA reference architecture, standard technologies and defining architectural principles of the targeted SOA. In this stage the decisions refer to selecting appropriate architectural patterns, as well as reference architectures. For example, if the conceptual architecture refers to a *process-driven SOA* (Hentrich & Zdun, 2012), which means that services are invoked to carry out business processes, appropriate architectural patterns, like DOMAIN-/TECHNICAL VIEW and the MACRO-/MICROFLOW (Hentrich & Zdun, 2012) should be considered which facilitate closing the gap between the business perspective of the business process flow and the technical perspective of underlying services. Furthermore company-specific technology stacks are defined in this stage, considering (selecting) industry- or vendor-specific reference technology stacks.

3.4 Stage 3 – Service Identification

The outline of the conceptual and technical target architecture facilitates identifying the required business services in the third stage. Business services provide functionality to the end-users. There are different approaches and patterns for service identification. For example, Frey et al. propose applying the CAPABILITY-BASED SERVICE pattern (Frey et al., 2014) if the services must be identified within a short timeframe and if there a comprehensive business process model is missing. The CAPABILITY-BASED SERVICE pattern is a top-down approach which is based on business capabilities. A business capability defines *what*

an enterprise does, and leaves out details about *how* it is done (Rosen, 2010). Instead, a business process describes in more details (usually in form of a flow of process steps) how a function is executed. In case the business processes of a company have already been analyzed and documented, the BUSINESS-DRIVEN SERVICE pattern (Hentrich and Zdun, 2006), which uses business processes as a starting point, may fit better for identifying business services. Fareghzadeh, too, proposes to break-down business process for identifying services top-down (Fareghzadeh, 2008).

3.5 Stage 4 – Service Design

Stage 4, Service Design, refers to high-level technology decisions for the design of the services. First, it must be decided, *who* should act as provider for the determined business services. A service can be developed from the scratch as a new service component. Alternatively, a legacy system may be re-used and service-enabled, for example by applying the WRAPPER FACADE (Schmidt et al., 2000) or WRAPPER (Ciurana, 2008). The OpenGroup, a global consortium for open, vendor-neutral IT standards and, certifications, provides a catalog of service enabling and service reengineering strategies, which are similar to pattern descriptions (OpenGroup 2012). Re-using an existing application as service provider can save costs and effort, depending on the application as well as the requirements of the service. If there are several potential service providers, for example legacy systems from different regions, a favorite system must be determined. At the end, it should be decided for each of the identified services which legacy system or new service acts as service provider and which service-enabling pattern will be applied (for legacy systems).

Second, high-level technical services are determined during the service design phase. A technical service refers to a cross-cutting technical concern, like an enterprise service bus, and is re-used by business-services. One example of a technical service is a transformation service, which provides standard functionality for converting different data models to the organization's standardized, canonical data model. Determining technical services is closely linked to the SOA reference architecture and its technology stack determined in the previous stage. Decisions about technical services can also be documented by choosing appropriate SOA patterns, like AGGREGATOR, SERVICE BUS or PROCESS AGGREGRATION (for a description of these patterns, please refer to Ciurana, 2008).

Technical services may also refer to an intermediate integration platform which may be required for providing an abstraction layer between service consumers and platforms which provide services. For example, consider a transactional banking platform providing a low-level proprietary service interface with complex data structures. It may not be a good idea to directly use the low-level services for the development of web-based online banking applications, because it takes a lot of effort for using the complex, proprietary interface. Furthermore the online banking application would be (too) strictly coupled with the transactional banking platform with respect to changes of the interfaces and data structures. This example shows that for some cases it may be helpful introducing an intermediate layer in form of a service-based integration platform which integrates services of underlying systems and provides a loosely-coupled service interface for service consumer, like end-user applications or other systems. Lytra et al. describe a pattern language for service-based platform integration and adaptation (Lytra et al., 2012), which considers four categories for high-level architectural decisions about service platform integration: Integration and Adaptation, Interface Design, Communication Style, and Communication Flow. For each category a relevant ADs and respective patterns are described. This pattern language can be applied for determining ADs for a service-based integration platform.

3.6 Stage 5 – SOA Roadmap Planning

While the Stages 1-4 concern the direction and the definition of the targeted SOA architecture and required services and service providers, the final stage refers to identifying and planning required projects for a high-level roadmap of the SOA modernization program. Projects refer to the development of new service components and service-enabling activities of legacy systems. Furthermore, projects for migration activities are required, taking into account that the (full or partly) replacement of large legacy applications needs to be considered and aligned thoughtfully with the development, test and rollout of the services. Directives and priorities from Stage 1 need to be considered, for example for determining which services are required first and/or which of the outdated legacy systems should be replaced first. Apart from the projects for

service development and migration additional projects need to be considered for introducing cross-functional technical platforms, like an enterprise service bus, and for other cross-cutting concerns, like the consolidation of information models. Hence, results from Stage 2 about architectural principles, the SOA reference and standard technologies also need to be considered. Furthermore it should be considered that a SOA usually requires organizational change, regarding roles, responsibilities, and governance processes for the development and operation of technical platforms as well as services. Of course, organizational capabilities and requirements of the particular company also need to be considered, referring, for example, to restrictions of people, skills, and the number of projects, which can be planned and executed in parallel.

The complexity of determining and planning multiple projects for a SOA modernization program requires a systematic planning approach. But the actual planning procedures for SOA roadmaps have not been in the center of research in the recent years. Khadka et al., for example, analyzed 121 publications regarding the evolutionary step from legacy applications to SOA (Khadka et al., 2012). They considered the (evolution) planning stage as one of two major parts of their analysis framework. This planning stage consists of the following aspects: legacy system understanding, target system understanding and evolution feasibility determination. But the study does not consider any criteria about actual project planning activities. Even this comprehensive literature review does not point to any research results about planning projects for a SOA roadmap. So, patterns regarding the actual roadmap planning activities of Stage 5 have not been identified, indicating the need for further research.

But there are various kinds of planning guidelines. For example, TOGAF[5], the architecture framework of the Open Group, considers a migration planning phase, which mentions activities for estimating required resources, project scheduling, and a cost-/value-based prioritization of work packages (TOGAF 2011). TOGAF mentions some techniques as well as diagrams and catalogs for visualizing migration planning activities. The recommendations are general (high-level) to be suitable for all kind of architecture projects. Defining the roadmap of a large-scale SOA modernization program (which is the context of this paper), comes along with specific challenges, which should be addressed by more specific guidelines or heuristics in order to provide guidance for real-world projects.

To sum it up, we have not discovered any specific patterns or concrete guidelines for prioritizing, estimating and planning multiple SOA projects on a roadmap. Following, we summarize specific challenges considering the context of a SOA modernization program. Furthermore we propose a planning heuristic which addresses these challenges.

Challenges

A SOA modernization program transforms the existing application landscape to a set of service components and services, which provide the functionality to end-users or to other (high-level) services. For example, atomic services provide basic functionality consumed by composite services, which support activities of business processes. Consequently, a SOA roadmap contains development projects for different types of services (depending on the SOA architecture and identified projects from the previous Stages).

Developing new service components and service-enabling existing applications need to be estimated and be planned on a timeline. If outdated legacy applications need to be shut-down and replaced, migrating the operations from those needs to be planned and aligned with the service development projects, too. Major challenges of planning the development and migration projects (on a high-level) refer to the following questions:

- How can business priorities be considered sufficiently in the planning process? For example, how can be ensured that those legacy applications with the highest maintenance costs and/or worst business performance are replaced first by services (results from Stages 1 and 3)?
- How can the duration of the projects be estimated, considering different types of services and varying complexity?
- How can the complexity of planning multiple related projects be handled?

[5] TOGAF refers to The Open Group Architecture Framework. Its current version TOGAF 9.1 was published in 2011.

- How can the projects be planned on a roadmap in a systematic way, considering organizational limitations, like a limited number of projects executed in parallel?

Planning Heuristic

As a response to those challenges and based on the practical experience from SOA modernization projects we developed a heuristic-based planning procedure, which consists of ten steps. They are described in Figure 3.

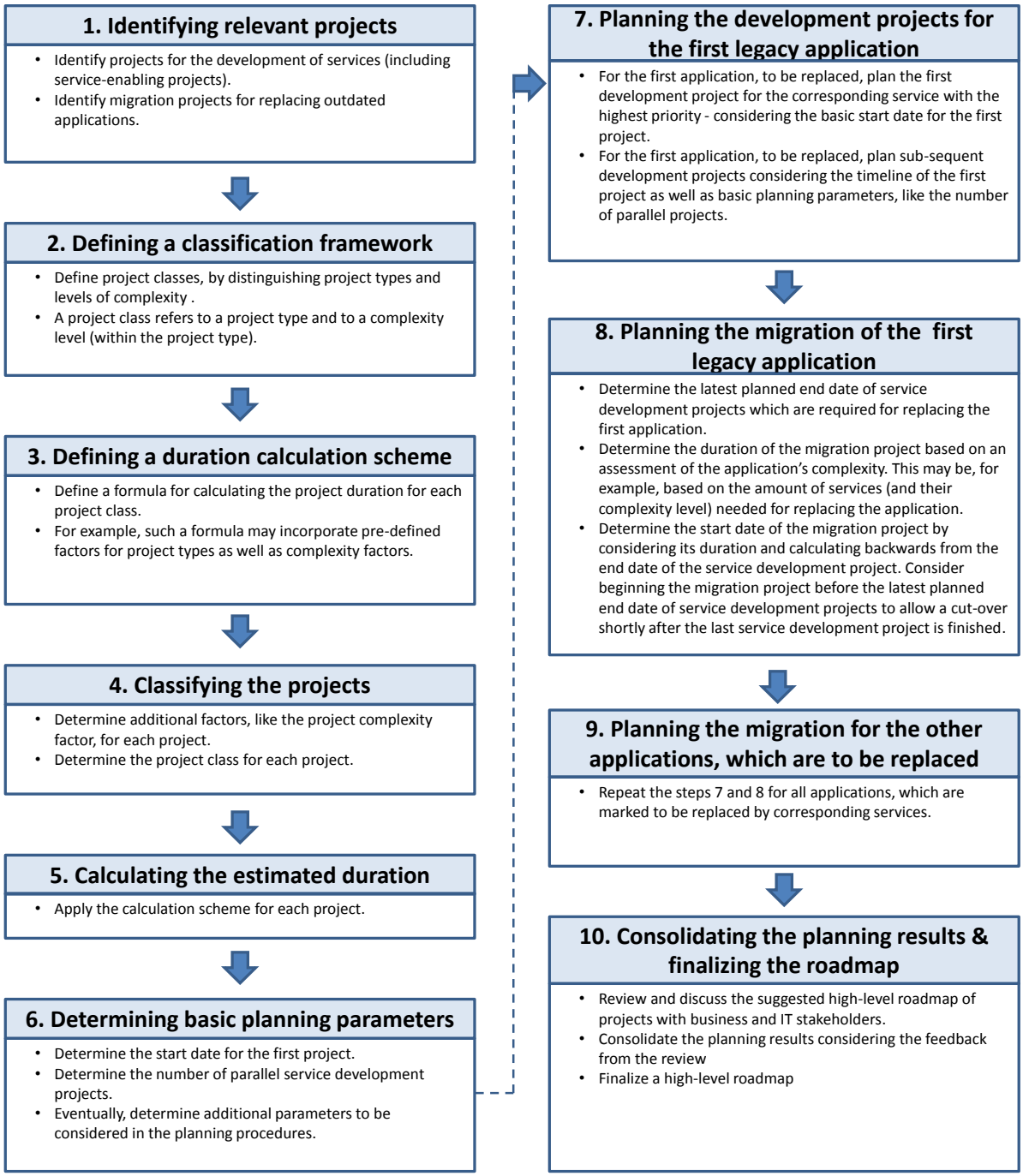

Figure 3: Roadmap Planning Heuristic

This heuristics is supposed to provide guidance on how to create a high-level project roadmap based on the results from the previous Stages. Of course, it needs to be adapted and parameterized according to the drivers and needs of a concrete organization and a concrete modernization program or project for which it should be applied. For example, the projects classification framework and a scheme or formula for estimating the duration of the projects must be defined. Furthermore a function point method (Balzert, 2000) could be considered for estimating purposes in order to distinguish projects of different sizes and estimate their duration.

4 INDUSTRY CASE STUDY

4.1 Context

The design process for the definition of a SOA roadmap, described in Section 3 was applied in a real-world scenario for defining a SOA modernization program. We picked this case study because it is a large-scale comprehensive case covering all Stages of the roadmap design process. This case was also presented in our paper introducing the CAPABILITY-BASED SERVICE pattern (Frey et al., 2014). We identified similar situations at other companies, but not all cases were so comprehensive or only limited information was available.

The case study concerns the shipment business of a global logistics provider, operating in more than 200 countries. The SOA modernization program was supposed to be defined on a high-level basis concerning the SOA target architecture, roadmap, governance model, and recommendations about development tools and processes. Stakeholders from business and IT were convinced that the existing application architecture, with several 20+ years old monolithic applications needs to be modernized and transformed into a modern SOA, which is more flexible regarding business requirements, and re-use of functionality for different regions and leads to less functional redundancies and maintenance costs. A few services have already been developed before, but previous attempts for initiating a large-scale modernization program have failed because the target state and roadmap was unclear and difficult to align. Hence, the initiative for defining the SOA modernization program was another (maybe the last) attempt and required defining the program within a short timeframe (less than 3 months) in order to keep the momentum and get the commitment from business and IT stakeholder.

4.2 Stage 1 – Setting Directions

The first stage refers to primary directives for the whole modernization program. Based on previous analysis the primary objective of this program was to replace a few major shipment applications, which were outdated (20+ years-old) and costly to maintain. They had a lot of functional overlaps because they provide more or less the same functionality, but were built for different regions, for example one application for America, another for Europe, and a third one for Asia. Key drivers and expected benefits are illustrated in Figure 4.

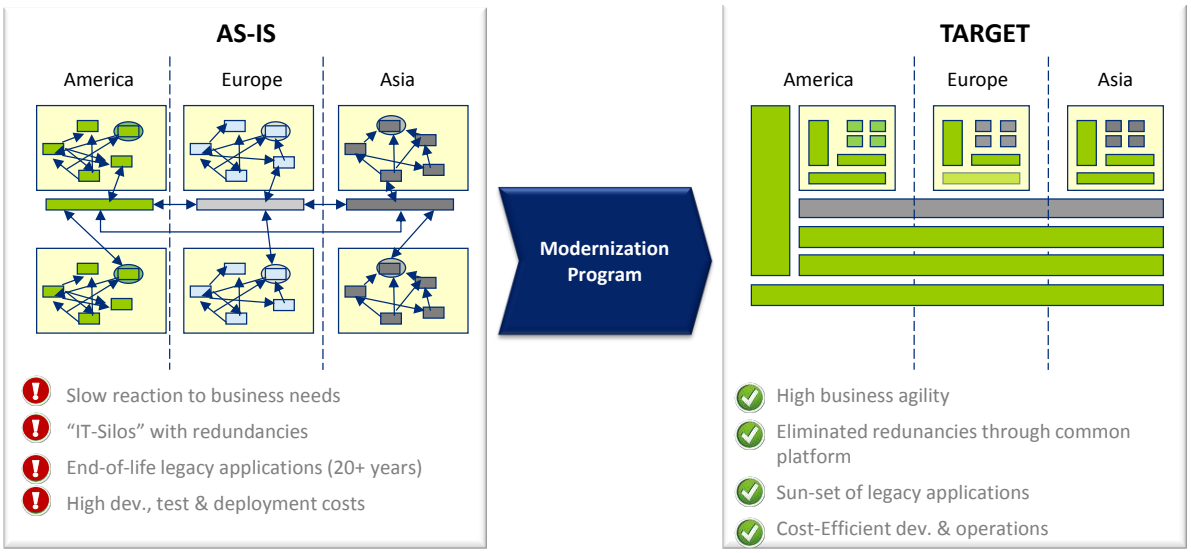

Figure 4: Drivers for IT Modernization - Example from Logistics Provider (Frey et al., 2014)

Business executives as well as IT executives prioritized which legacy systems must be replaced and shutdown as a result of the modernization program. The executive clearly stated that reducing costs is the primary goal. Especially, reducing maintenance cost by shutting down the old shipment applications was said to be more important than rolling out new functionality.

If any business pattern had been applied to reach these executive decisions is unknown, because they were already given as input from the previous attempts to modernize the IT architecture. Furthermore the following directives were also given: The targeted architecture should follow the architecture style of SOA and a company-specific SOA architecture needs to be defined because no industry reference architecture is thought to capture the company specific requirements.

4.3 Stage 2: SOA Architecture Definition

As prerequisite for the definition of the SOA architecture major architectural requirements as well as the architecture vision and architecture principles were defined. Architecture principles, for example, refer to the following decisions:

- The SOA must have a layered solution stack, which means that a service can be composed of services from the same or lower levels of the solution stack.
- The services are designed stateless and the functionality and non-functional characteristics of a service are described in form of a service contract.
- The services should be "business-driven services", which, in this case, means that the requirements and the design of the services are supposed to be driven by business capabilities and business processes (this is relevant for the service identification in stage 3)
- Furthermore an event-driven-architecture was selected which is based on a PUBLISHER/SUBSCRIBE ARCHITECTURE and an enterprise service bus as described by Hentrich and Zdun (Hentrich and Zdun, 2012).

After that, a high-level company-specific SOA architecture was defined following the directives of Stage 1. Table 2 shows the summarized results of key decisions of Stage 2.

Table 2: Decisions of Stage 2

Decision Type	Decision	Decision Results
Conceptual Decisions (CD)	CD1: SOA reference architecture	A company-specific SOA architecture was defined (i.e. including terminology, definition of SOA layers, architectural principles, service types, architectural building blocks)
	CD2: process / service integration paradigm	Process-driven SOA patterns, i.e. • DOMAIN-/TECHNICAL-VIEW (Hentrich and Zdun, 2012) Event-driven architecture patterns, i.e. • EVENT-BASED-ACTIVITY (Köllmann and Hentrich, 2007)
	CD3: routing paradigm	Routing patterns, i.e. • DYNAMIC ROUTER Ciurana, 2008) • AGGREGATOR (Ciurana, 2008) • other patterns concerning routing and proxy services (without explicit sources)
	CD4: interaction paradigm	Interaction patterns, i.e.: • FIRE EVENT ACTIVITY (Hentrich and Zdun, 2008) • MULTIPLE ASYNCHRONOUS RESULTS SERVICE (Hentrich and Zdun, 2008)
	CD5: Data transformation paradigm	Business object transformation patterns, i.e. • INTEGRATED BUSINESS OBJECT MODEL (Hentrich and Zdun, 2012) • other data transformation patterns concerning cache services and file gateways (without explicit sources) A blueprint of a business object model was created including a mapping to services.
Technology Directive Decisions (TDD)	TDD1: high-level technology stack	Basic technologies and a high-level technology stack Blueprint of the data and security architecture

Furthermore a blueprint was defined concerning conceptual building blocks of the target architecture as well as the transition architecture describing an intermediate state of the architecture. Infrastructural aspects were considered in the definition of basic technologies and the blueprint of the data and security architecture.

4.4 Stage 3 – Service Identification

At this stage the services for business users were determined, considering the architecture principle to provide business-driven services, based on business capabilities and business processes. A comprehensive business process model was missing for the shipment domain. But a business capability model had been defined and used by the business experts for other purposes. As this model was considered to be a sufficient starting point and allowed to get commitment from business experts, it was decided to determine services based on business capabilities, following the CAPABILITY-BASED SERVICE pattern (Frey et al., 2014). The existing business capability model was updated, based on input from the business experts, considering not only current capabilities, but also capabilities which will be required several years ahead (at the end of the modernization program).

4.5 Stage 4: Service Design

After determining which business services are required, it must be determined which services need to be developed from scratch and which can be provided by service-enabled legacy systems. A comprehensive

survey based application assessment was conducted with respect to business fit, technical fit and options for service-enabling. Furthermore it was analyzed which applications are considered to be shut down and replaced (in addition to the applications defined in Stage 1). The following options for service-enabling where distinguished:

- ***Option 1 - Wrap**:* Reuse the existing application and build a service wrapper on top of it. The application is considered technically fit for purpose, being able to represent the mapped services, but is not yet (fully) service enabled (referring to the WRAPPER pattern (Ciurana, 2008)
- ***Option 2 - Reengineer**:* The application is based on fit for purpose technology but needs to be reengineered to implement mapped services (referring to reengineering strategies, which are similar to pattern descriptions (OpenGroup 2012)
- ***Option 3 - Reuse DB**:* The application itself cannot be reused, because it is, for example, based on the wrong technology. But the database of the application can be reused. Eventually, data cleansing and data quality improvements may be required.

Five of the large shipping applications were marked to be replaced with top priority. For the other applications potential service-enabling options or replacement options were determined. The results, documented in form of service-enabling (or shutdown) options, are partly comparable to the selection of patterns.

Knowing which legacy applications are suitable to be service-enabled is not sufficient for defining a roadmap. A functional mapping is required for determining which services can be provided by which legacy application. In this case, the business capability model was used to map functionality of the legacy systems, broken down into functional modules, over business capabilities to the targeted business services.

In order to explain how the functional mapping works, Figure 5 shows a simplified example for the replacement of two legacy systems. Both have a functional overlap because they both support the Functional Module “Export”, which refers to the Business Capability “Checkpoint Data Capture” and respective Service “S4: Capture shipment checkpoint”. Thus, if both systems are suitable to be service-enabled it needs to be clarified which is supposed to be the future provider for service “S4”. In case a service could be provided by several legacy systems, a heuristic was used for selecting one legacy system per service. The heuristic considers, for example, that Option 1 requires less development effort than Option 2 because the existing system does not need any changes of inner structures, which are required for Option 2. Consequently, Option 1 is preferred over Option 2 or 3.

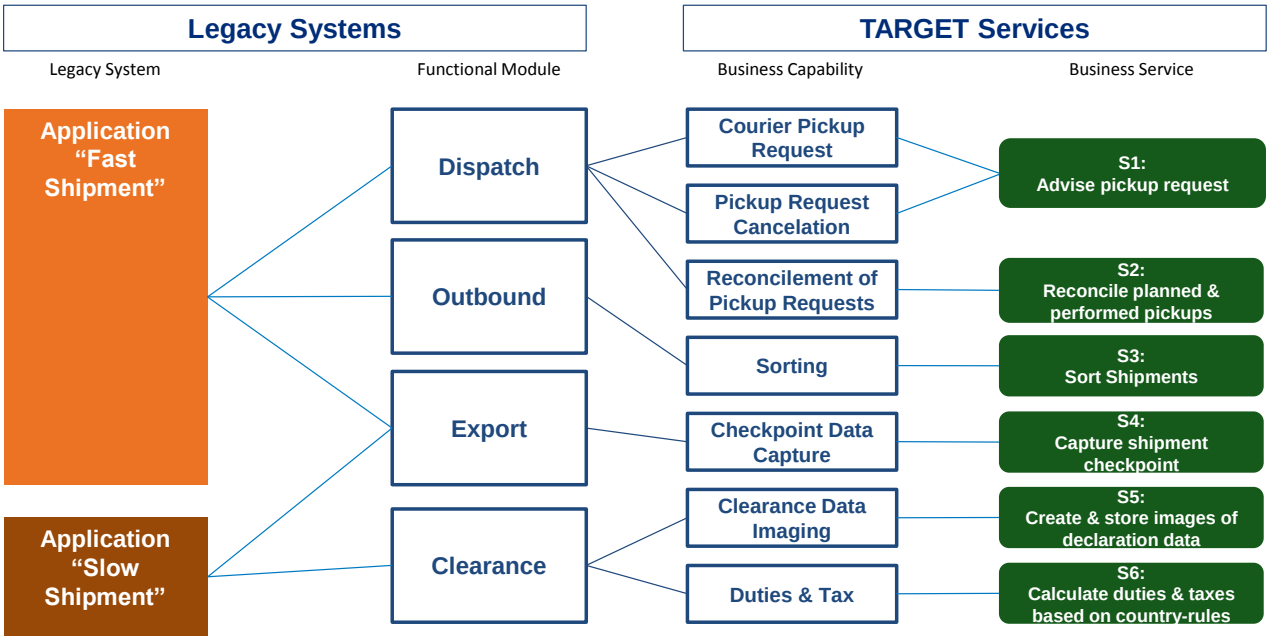

Figure 5: Functional Mapping of Legacy Systems and Target Services – Illustrative Example

Finally, for each business service a future service provider is determined – either based on a service-enabling option or in form of a new service component. Apart from business services, technical services were determined considering the architectural decisions from Stage 2. As a result, a technical service model was defined containing services, which, for example, refer to interaction, routing, and data transformation patterns indicated above (see Section 4.3).

4.6 Stage 5: SOA Roadmap Planning

Overview

After going through the previous Stages – defining the primary objectives of the SOA modernization program (Stage 1), defining how the service-oriented target architecture should look like on a high-level (Stage 2), identifying business services (Stage 3) and determining who will act as service provider in the future (Stage 4) – the last Stage concerns determining required work packages and projects and defining a high-level roadmap.

Transforming company's IT to a SOA required new, cross-functional technical platforms, like an enterprise service bus (based on the results from Stages 2 and 4). Activities for introducing these technical platforms were planned manually, considering a default process for vendor selection, piloting and rollout of the platform. Furthermore a phase for the program initiation (including the definition of the program organization) and other aspects, like cross-functional support from enterprise architects, continuous program management and transformation management (covering organizational change management), were planned manually based on proven frameworks and previous experience from large IT programs.

Figure 6 gives an overview of the work streams of the modernization program: Program Initiation, Program Management, Technical Platform, Enterprise Architecture Support, Transformation Management, and Projects. The figure indicates a major work stream for Projects, referring to the projects for service development and for the migration (replacement) of the outdated legacy applications.

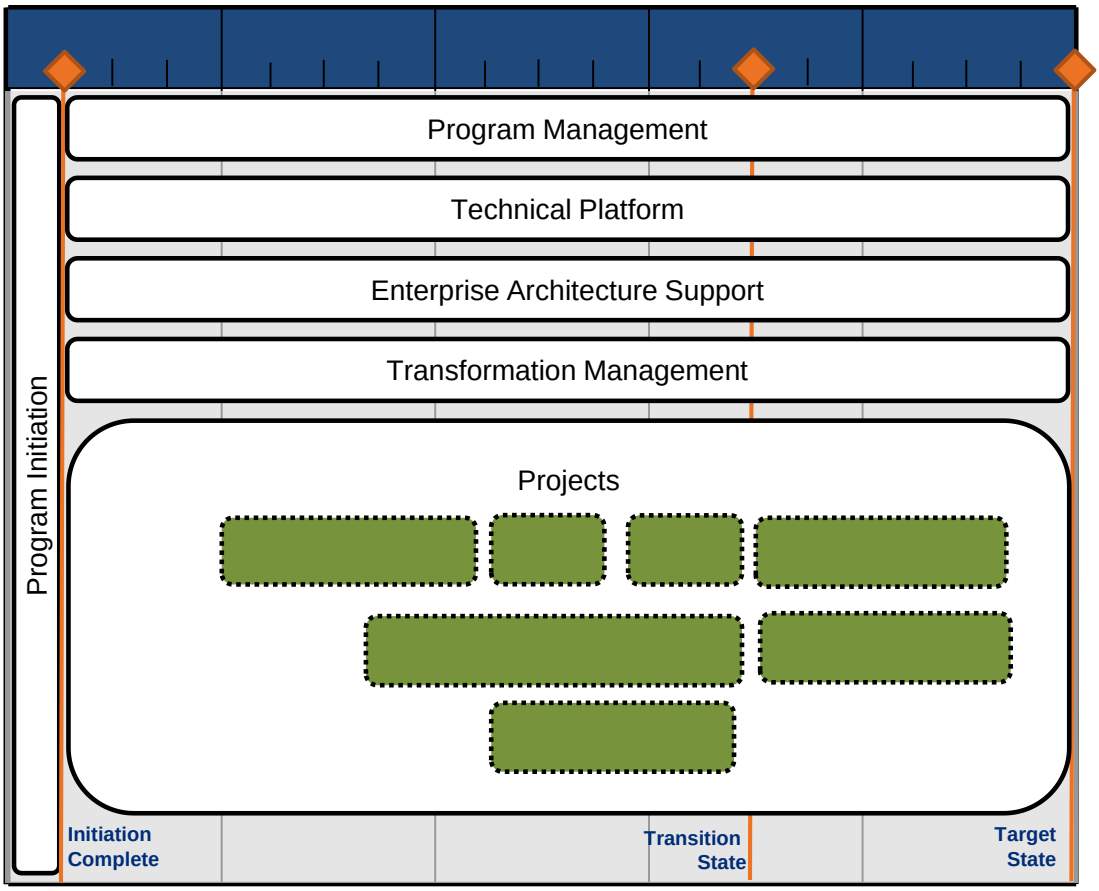

Figure 6: Work streams of the SOA Modernization Program

Challenges
Considering the Projects workstream we were facing similar challenges concerning the roadmap planning like the ones mentioned in Section 3.6. In this case, shutting down the large outdated shipping was the driving force for the whole program and a particular challenge for the planning process.

Planning Heuristic Example
For creating a roadmap of development projects we adapted and detailed the general planning heuristics described in Section 3.6. Following, we want to give an illustrative (simplified) example how we applied the heuristics.

Let us assume that six services are required for replacing both legacy systems “Fast Shipment” and “Slow Shipment” (referring to the example from Section 4.5). Some services must be developed as new service components, others will be provided by the service-enabled applications “New Shipment” and “Modern Shipment”. Table 3 shows an overview of the services, applications, and future service providers.

Table 3: Service Development Options (Example)

Service of the Target Architecture	**Applications to be replaced**	**Service-development Option**	**Future Service Provider**
s1: Advise pickup request	Fast Shipment	new service component	to be defined
s2: Reconcile planned & performed	Fast Shipment	new service	to be defined

Service of the Target Architecture	Applications to be replaced	Service-development Option	Future Service Provider
pickups		component	
s3: Sort shipments	Fast Shipment	new service component	to be defined
s4: Capture shipment checkpoint	Fast Shipment, Slow Shipment	wrap	New Shipment
s5: Create & store images of decleration data	Slow Shipment	re-engineer	Modern Shipment
s6: Calculate duties & taxes based on country-rules	Slow Shipment	re-engineer	Modern Shipment

The planning heuristic described in Section 3.6 was applied in this project. Following, it is described how the heuristic was adapted considering the data from Table 3:

Step 1: Identifying relevant projects:

- service development projects: s1, s2, s3, s4, s4, s6
- migration project for replacing “Fast Shipment”
- migration project for replacing “Slow Shipment”

Step 2: Defining a classification framework (for projects):

- We assume that the projects are classified by type (service development new/wrap/re-engineer and migration) and by complexity of each type (low/high).
- For service development projects a low complexity refers to an atomic service and high complexity to a composite service (which uses more than one atomic service).
- A migration project is considered as highly complex, if it refers to more than 5 services with low complexity or more than 2 services with high complexity. Otherwise the migration project is regarded as low complex.

Step 3: Defining a duration calculation schema: see Table 4

Table 4: Calculation Schema per Project Type (Example)

Project Type	Basic duration in months	Complexity factor “low”	Complexity factor high”	Duration in month
service dev. “wrap”	1	1	3	Basic duration x complexity factor
service dev. “re-engineer”	2	1	4	Basic duration x complexity factor
service dev. “new”	3	1	3	Basic duration x complexity factor
migration	4	1	2	Basic duration x complexity factor

Steps 4 & 5: Classifying the projects & calculating the estimated duration:

- Applying the definition of the project classes and the calculation schema Table 5 shows the results of steps 4 and 5 regarding classification of the projects and estimation of the project duration.

Table 5: Classification and Estimation of Projects (Example)

Project Type	Related Service	Type of Service	Service Provider	Applications to be replaced	Com-plexity	Estimated Duration in months
service dev. "wrap"	s4	Atomic	New Shipment	Fast Shipment, Slow Shipment	Low	1
service dev. "new"	s1	Composite	<new>	Fast Shipment	High	9
service dev. "new"	s2	Composite	<new>	Fast Shipment	High	9
service dev. "new"	s3	Composite	<new>	Fast Shipment	High	9
service dev. "re-engineer"	s5	Composite	Modern Shipment	Slow Shipment	High	8
service dev. "re-engineer"	s6	atomic	Modern Shipment	Slow Shipment	Low	2
Migration	n/a	n/a	n/a	Fast Shipment	High	8
Migration	n/a	n/a	n/a	Slow Shipment	Low	4

Step 6: Determining basic planning parameters:

- We assume the following basic planning parameters (step 5):
  - Start date: 1 July 2014
  - Number of parallel service development projects: 3
  - Migration projects: are supposed to end 2 months after the latest end of the service development projects, which are required for replacing the application.

Steps 7-10: Planning the development projects for the first legacy application, **Planning the migration of the first legacy application, Planning the migration for the other applications, which are to be replaced & Consolidating the planning results & finalizing the roadmap:**

- The planning results for service development and migration projects (steps 7-10) are presented in the following table.

Table 6: Results of the Project Planning (Example)

Project Name	Type of Service	Service Provider	Systems to be replaced	Com-plexity	Duration / months	Start Date	End Date
service dev. "wrap" - s4	Atomic	New Shipment	Fast Shipment, Slow Shipment	Low	1	01.07.2014	30.07.2014
service dev. "new" - s1	Composite	<new>	Fast Shipment	High	9	01.07.2014	31.03.2015
service dev. "new" - s2	Composite	<new>	Fast Shipment	High	9	01.07.2014	31.03.2015
service dev. "new" - s3	Composite	<new>	Fast Shipment	High	9	01.08.2014	30.04.2015
migration - Fast Shipment	n/a	n/a	Fast Shipment	High	8	01.11.2014	30.06.2015

Project Name	Type of Service	Service Provider	Systems to be replaced	Com-plexity	Duration / months	Start Date	End Date
service dev. “re-engineer” - s5	Composite	Modern Shipment	Slow Shipment	High	8	01.04.2015	30.11.2015
service dev. “re-engineer” - s6	atomic	Modern Shipment	Slow Shipment	Low	2	01.04.2015	31.05.2015
migration - Slow Shipment	n/a	n/a	Slow Shipment	Low	4	01.10.2015	31.01.2016

Figure 7 illustrates the planning results for the illustrative in a (roadmap) diagram, which provides an aggregated view of the timeline of the service development and migration projects. Different planning views were created in order to present, discuss and consolidate the results.

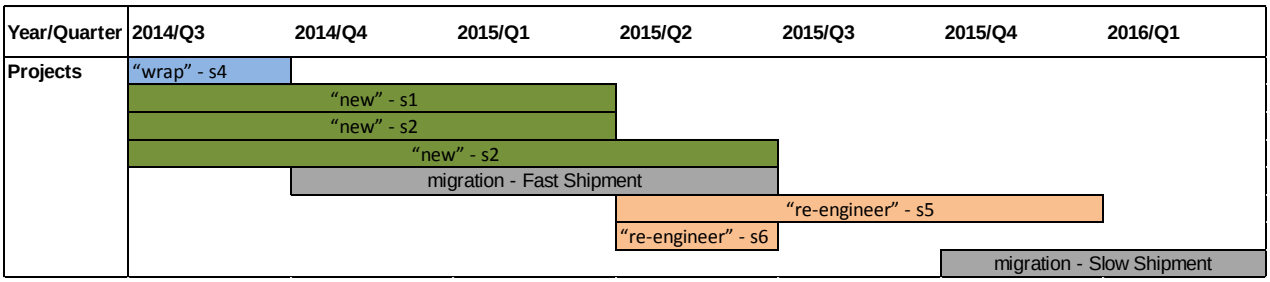

Figure 7: Roadmap diagram (Example)

Consolidating the projects roadmap was more complex than in the simplified example described above because the large number of projects (>100) and dependencies between them had to be considered. Consequently, a macro-based planning tool was used to support the planning procedures. This also helped creating different planning views which allowed discussing and aligning the results with different stakeholder groups. For example, an aggregated, high-level view was used to discuss the planning with the senior management.

5 CONCLUSION

In this paper we described a design process for defining a roadmap in the context of a preliminary phase for the definition of a legacy to SOA modernization program. The design process is inspired by the ArchPad design method (Zimmermann et al., 2008). Like ArchPad, our design process also considers a sequence of design stages and ADs for each stage. The decisions are supposed to be captured by selecting a pattern or reference architecture reflecting the decision. Selecting and referencing a pattern or reference architecture is expected to reduce the documentation effort required for each AD, because details (like context, problem and solution) are explained in the description of the referenced pattern. Furthermore the design process is supposed to give some guidance for industry experts who are involved in the definition of a SOA roadmap and/or definition of a high-level SOA target architecture.

The proposed design process, including the suggested roadmap planning heuristic, has been applied and adapted successfully in an industry case. The case study showed that the systematic approach with a distinction between the stages and ADs on different levels from different stakeholders is a helpful framework for organizing and documenting the definition of a SOA modernization program and its

roadmap. But it also made clear that selecting patterns and reference architectures was not possible for documenting all ADs in all Stages. While several patterns were documented for ADs concerning the Stages 2-4, specific patterns regarding the actual planning process in Stage 5 were not considered. Instead, the heuristics, suggested by the authors, was applied. Consequently, further research, for example a systematic literature review, may show if there is a research gap concerning roadmap planning patterns and their usage (providing a comprehensive answer to the following questions: Which roadmap planning patterns exist? How can they be categorized and how are they applied successfully in real-world projects?).
Furthermore the case study showed that in many cases the selected patterns were not referenced but rather copied into the documentation of the SOA modernization program. Copying the pattern descriptions was done because a well-accepted, standardized pattern repository (enterprise-wide or industry-wide), which could be used as source for references, was missing. Another reason was that the client explicitly wished to include the descriptions of the most important patterns into the documentation, so it is not necessary for the audience to refer to additional documents. Still for some patterns only a short description and/or reference was added to the documentation. Consequently, the documentation, containing full text, short descriptions as well as references, shrank a little bit compared with a full-blown description of all ADs. It also needs to be noted, that the standardized, systematic format of pattern descriptions was well accepted by the client.

Finally, it needs to be considered that the proposed roadmap design process for SOA roadmaps has only been applied in one real-world project. Further research and practical application in other industry cases is required for determining the general applicability of the proposed design process. Furthermore a comprehensive architectural decision model may be developed, based on further research of patterns and corresponding ADs, which provides additional guidance to lead industry experts through all stages of the process.

ACKNOWLEDGEMENTS

We would like to thank our shepherd Stefan Sobernig for his strong support and valuable feedback for improving this paper.

REFERENCES

Avgeriou, P., Zdun, U. (2005): Architectural patterns revisited – a pattern language. In Proceedings of 10th European Conference on Pattern Languages of Programs (EuroPlop 05). Irsee, Germany, pp. 1–39.
Balzert, H. (2000): Lehrbuch der Software-Technik, zweite Auflage. Spektrum Akademischer Verlag Heidelberg, Berlin.
BIAN (2014). Service Landscape of the Banking Industry Architecture Network. http://bian.org/assets/bian-standards/bian-service-landscape-2-5/ (accessed in February 2014).
Buckl, S., Ernst, A. M., Lankes, J., and Matthes, F. (2008). Enterprise Architecture Management Pattern Catalog. In Technical Report TB 0801. Software Engineering for Business Information Systems (sebis), TU München.
https://wwwmatthes.in.tum.de/pages/qs55fpnsaq5/EAM%20Pattern%20Catalog/EAM%20Pattern%20Catalog%20v1.0 (accessed in January 2014).
Ciurana, E. (2008). SOA Patterns. RefCardzs. http://refcardz.dzone.com/refcardz/soa-patterns (accessed in January 2014).
Fareghzadeh, N. (2008). Service Identification Approach to SOA Development. World Academy of Science, Engineering and Technology, Vol. 2 No. 9, pp. 243-252.
eTOM (2014). Business process framework. tmforum. http://www.tmforum.org/businessprocessframework/1647/home.html (accessed in February 2014).
Evans, E. (2003). Domain-driven Design. Tackling Complexity in the Heart of Software. Addison-Wesley.
Frey, F., Hentrich, C., Zdun, U. (2014). Capability-based Service Identification in Service-Oriented Legacy Modernization. In Proceedings of 18th European Conference on Pattern Languages of Programs (EuroPlop 2013). Irsee, Germany (to be published).
Harrison, N. and Avgeriou, P. and Zdun, U. (2007). Using Patterns to Capture Architectural Decisions. IEEE Software, 24 (4). pp. 38-45.
Hentrich, C., Zdun, U. (2006). Patterns for process-oriented integration in service-oriented architectures. In Proceedings of 11th European Conference on Pattern Languages of Programs (EuroPlop 06). Irsee, Germany, pp. 141–189.
Hentrich, C. and Zdun, U. (2008). Service integration patterns for invoking services from business processes. In Proceedings of the 12th European Conference on Pattern Languages of Programs (EuroPLoP) 2007. Universitätsverlag Konstanz.
Hentrich, C., Zdun, U. (2012). Process-Driven SOA: Patterns for Aligning Business and IT. CRC Press.
Homann, U. (2006). A Business-Oriented Foundation for Service Orientation. Microsoft MSDN. http://msdn.microsoft.com/en-us/library/aa479368.aspx.
Ionita, M. Ltoiu, G. Lewis. Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments. IGI Global, pp. 40-70.

Jansen, A., Bosch, J. (2005). Software architecture as a set of architectural design decisions. In Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture, IEEE Computer Society, Pittsburgh, USA, pp. 109–120.
Khadka, R., Saeidi, A., Idu, A., Hage, J., and Jansen, S.. (2012). Legacy to SOA Evolution: A Systematic Literature Review. In A.
Kruchten, P. (2000). The Rational Unified Process: An Introduction, 2nd Edition. Addison-Wesley.
Köllmann, T. and Hentrich, C. (2007). Synchronization patterns for process-driven and service-oriented architectures. In Proceedings of the 11th European Conference on Pattern Languages of Programs (EuroPLoP) 2006, pp. 199 – 227. Universitätsverlag Konstanz.
Lytra, I., Sobernig, S., Tran, H., and Zdun, U. (2012). A Pattern Language for Service-Based Platform Integration and Adaptation. In Proceedings of the 17th European Conference on Pattern Languages of Programs (EuroPlop 12). Irsee, Germany.
Maréchaux, J.L. (2008). Modernize legacy systems using an SOA approach. IBM Developer Works.
http://public.dhe.ibm.com/software/dw/webservices/ws-soa-legacymod/ws-soa-legacymod-pdf.pdf (accessed in January 2014).
Open Group (2011). SOA Reference Architecture Technical Standard.http://www.opengroup.org/soa/source-book/soa_refarch/ (accessed in April 2014).
Open Group (2012). Legacy Evolution to SOA. Guide 122. www.opengroup.org/soa/source-book/l2soa/best-practices.htm#X_Modernization_Strategies (accessed in January 2014).
Rasch, M., Billeb M. (2013). IT-Finanzarchitektur – Zufallsprodukt oder gezielte Weiterentwicklung?.
http://www.pwc.de/de/finanzdienstleistungen/banken/pwc-studie_auf-dem-weg-zu-einer-integrierten-it-finanzarchitektur.jhtml (accessed in January 2014).
Schmidt et al. (2000).Pattern-Oriented Software Architecture, Vol 2: Patterns for Concurrent and Networked Objects, Wiley and Sons.
That, M. T. T, Sadou, S., Oquendo, F. (2012). Using Architectural Patterns to Define Architectural Decisions. In Proceedings of the 10th Working IEEE/IFIP Conference on Software Architecture, IEEE Computer Society, Helsinki, Finland, pp. 196-200.
TOGAF (2011). The Open Group Architecture Framework. Version 9.1. http://pubs.opengroup.org/architecture/togaf9-doc/arch/index.html (accessed in April 2014).
Zimmermann, O., Zdun, U., Gschwind, T., and Leymann, F. (2008), Combining Pattern Languages and Reusable Architectural Decision Models into a Comprehensive and Comprehensible Design Method. In Proceedings of the 7th Working IEEE/IFIP Conference on Software Architecture, IEEE Computer Society, Vancouver, Canada pp. 157-166.