Aligning Enterprise, System, and Software Architectures

Ivan Mistrík
Independent Consultant, Germany

Antony Tang
Swinburne University of Technology, Australia

Rami Bahsoon
University of Birmingham, UK

Judith A. Stafford
Tufts University, USA

BUSINESS SCIENCE Reference

Managing Director:	Lindsay Johnston
Editorial Director:	Joel Gamon
Book Production Manager:	Jennifer Romanchak
Publishing Systems Analyst:	Adrienne Freeland
Development Editor:	Myla Merkel
Assistant Acquisitions Editor:	Kayla Wolfe
Typesetter:	Alyson Zerbe
Cover Design:	Nick Newcomer

Published in the United States of America by
Business Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Aligning enterprise, system, and software architectures / Ivan Mistrik ... [et al.], editors.
p. cm.
Includes bibliographical references and index.
Summary: “This book covers both theoretical approaches and practical solutions in the processes for aligning enterprise, systems, and software architectures”--Provided by publisher.
ISBN 978-1-4666-2199-2 (hardcover) -- ISBN 978-1-4666-2200-5 (ebook) -- ISBN 978-1-4666-2201-2 (print & perpetual access) 1. Management information systems. 2. Business enterprises--Computer networks. 3. Information technology--Management. 4. Software architecture. I. Mistrík, Ivan.
HD30.213.A45 2013
658.4’038011--dc23

2012026463

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 10
Decisions Required vs. Decisions Made:
Connecting Enterprise Architects and Solution Architects via Guidance Models

Olaf Zimmermann
IBM Research GmbH, Switzerland & ABB Corporate Research, Switzerland

Christoph Miksovic
IBM Research GmbH, Switzerland

ABSTRACT

Contemporary enterprise architecture frameworks excel at inventorying as-is and at specifying to-be architecture landscapes; they also help enterprise architects to establish governance processes and architectural principles. Solution architects, however, expect mature frameworks not only to express such fundamental design constraints, but also to provide concrete and tangible guidance how to comply with framework building blocks, processes, and principles – a route planner is needed in addition to maps of destinations. In this chapter, the authors show how to extend an existing enterprise architecture framework with decision guidance models that capture architectural decisions recurring in a particular domain. Such guidance models codify architectural knowledge by recommending proven patterns, technologies, and products; architectural principles are represented as decision drivers. Owned by enterprise architects but populated and consumed by solution architects, guidance models are living artifacts (reusable assets) that realize a lightweight knowledge exchange between the two communities – and provide the desired route planners for architectural analysis, synthesis, and evaluation.

DOI: 10.4018/978-1-4666-2199-2.ch010

INTRODUCTION

A key objective of enterprise architects is to align the existing and the future IT systems with the business model and the strategic direction of an enterprise. Architecture frameworks support enterprise architects when they inventory the existing (as-is) and when they specify the future (to-be) architecture landscapes; they also help them to establish governance processes and architectural principles. However, solution architects that work on specific implementation projects expect mature frameworks not only to express such fundamental design constraints, but also to provide concrete and tangible guidance how to comply with framework building blocks, processes, and principles. In other words, a route planner is needed in addition to maps of destinations.

In practice, we have come across the following collaboration issues between enterprise architects and solution architects that call for such a route planner:

1. **Availability Issues:** Experienced, knowledgeable enterprise architects have been appointed, who managed to define to-be architectures and to release enterprise-wide architectural principles. However, they did not find the time yet to author additional documentation *how* to adhere to these principles and they are slow to respond to requests for reviews and/or project participation. Consequently, the enterprise architecture artifacts are ignored by projects teams or, even worse, "pseudo-compliance" is declared at an early stage, but never really strived for and, consequently, never actually reached.
2. **Consumability Issues:** To-be architectures and/or architectural principles are documented, but difficult to understand and to relate to design concerns on projects. Such issues are often caused by inadequate levels of abstraction and detail: if specified on rather high levels, enterprise architecture artifacts run the risk of being perceived to be full of obvious truisms and/or trivial; on the contrary, rather detailed specifications take a long time to create, comprehend, and maintain; they might also be impossible to implement under economic constraints.
3. **Enforcement and Acceptance Issues:** Workable enterprise architecture guidelines are in place, as well as a governance process. However, the guidelines established by the enterprise architects (for the benefit of the whole enterprise) are not followed properly because project teams do not fully appreciate their value; due to their narrower design scope, they view these guidelines as unwelcome additional design constraints. In practice, we also observed that solution architectures often pass formal quality reviews with certain obligations; e.g., architectural smells are reported and refactorings are suggested to reduce technical debt (Brown, Nord, & Ozkaya, 2011). However, such obligations are not always followed up upon. In such settings, solution architects can expect to "get away" with violations of architectural principles, which typically are justified by short term business priorities and stakeholder pressure.

These issues are further complicated when third parties such as external consulting firms and outsourcing providers with different goals and concerns get involved; this is often the case in practice today.

Examples of architecture design issues that often require the attention of enterprise architects are:

- The architectural principle that all sensitive data has to be secured: security is not a single requirement, but a set of responses to certain threats requiring a risk analysis and risk management (mitigation) strategy, as well as a security requirements engineering effort. For instance, a data sensitivity classification scheme might be missing entirely. In other cases, such schemes only exist in rudimentary forms; specifically, examples or concrete advice how to classify application data such as customer profiles, orders, and invoices are rarely given. This example illustrates issues 1 and 2 from above (no availability and poor consumability).

- A data classification scheme exists and distinguishes uncritical from critical Personal Information (PI) and from highly critical Sensitive Personal Information (SPI); an infrastructure-level security zone model including network transport-level firewalls has been defined as well. However, the enterprise-wide security architecture does not specify whether HTTPS connections using server-side certificates (128 bit) are good enough to protect zone-to-zone traffic that goes through firewalls if this traffic carries SPI. Such assessment requires links from business-level compliance rules to logical (functional) application architectures and then to physical infrastructure architectures; such links often are not modeled explicitly so that traceability cannot be provided. This scenario also exemplifies issues 1 and 2 (lack of availability and limited consumability).
- The security architecture on the enterprise level clearly states that if SPI is transferred across firewalls, HTTPS has to be used; however, this decision is overruled and an unsecured HTTP channel is established by opening a particular port for a tactical application supposed to go live soon due to urgent business needs (e.g., to respond to a product announcement recently made by a competitor); this tactical solution remains operational for several years although violating an architectural principle from the enterprise level. This example illustrates/instantiates issue 3 from above (lack of enforcement and acceptance).

Other examples of architectural decisions that call for guidance from the enterprise level are: technology platform choices, vendor preferences, how to deal with regulatory and legal constraints, including audit requirements such as completeness, accuracy, validity and restricted access (Julisch, Sutter, Woitalla, & Zimmermann, 2011), usage of open source software (often a separate principle or decision is required per open source license scheme such as Apache or GNU Public License) (Haischt & Georg, 2010), and so forth. Additional examples, also pertaining to the design of logical (functional) application architectures and to application integration, are presented in our previous publications (Zimmermann, 2009).

These examples illustrate the gap between enterprise architecture and solution architecture that exists in the industry. The gap has multiple facets – objectives, terminology, methods, techniques, and tools of the two communities differ substantially. In our opinion, enterprise architecture only has a chance to have a sustainable impact on the business and IT projects if this gap is overcome; to do so, extensions to existing methods are required.

Objective: To close the gap between enterprise architects and solution architects, this chapter introduces a novel way for these communities to improve their communication and knowledge exchange:

How can enterprise architects support project-level solution architects with concrete managerial and technical advice regarding the architectural decision identification, making, and enforcement activities required to ensure that the system under construction meets the expectations of its primary *stakeholders (i.e., users, sponsors), but also respects the constraints imposed by implementation governance authorities (e.g., enterprise architects, auditors)?*

In response to this question, we propose to let enterprise and solution architects share architectural decision models that collect key issues and proven solutions to them. In this effort, we leverage, combine, and extend concepts from The Open Group Architecture Framework (TOGAF) (The Open Group, 2009) and from SOA Decision Modeling (SOAD) (Zimmermann, 2011). Specifically, we show how to extend TOGAF with SOAD guidance models that capture architectural decisions recurring in a particular domain such as Service-Oriented Architecture (SOA) design (Krafzig, Banke, & Slama, 2005). Such guidance models codify architectural knowledge by recommending proven patterns, technologies, and products; architectural principles are represented as decision drivers. Owned by an enterprise architect but populated and consumed by solution architects, guidance models are living artifacts and reusable assets that realize a lightweight knowledge exchange between the two communities.

Being integrated into TOGAF, guidance models can provide the desired route planners for architectural synthesis on implementation projects. Enterprise-level decision making is out of our scope for the time being; the presented concepts are designed to also work on that level. One can view TOGAF as an "über-guidance model" for enterprise architecture construction: TOGAF does not formally model decisions in an explicit guidance model, but still comprises a knowledge repository. Note that a solution constructed in an implementation project may be a marketed product such as an Enterprise Resource Planning (ERP) package or a client-specific solution (resulting from one-of-a-kind application development or integration).

INTRODUCTION TO THE CASE STUDY: PREMIER QUOTES GROUP (PQG)

To be able to exemplify our approach, we now introduce a basic scenario from the insurance industry. The scenario concerns a fictitious company and is simplified due to space constraints; however, business scenario, existing enterprise application architectures, and technical design considerations in the case originate directly from our rich project experience in several industry sectors (e.g., financial services, telecommunications, and automotive).

Let us assume that PremierQuotes Inc., an insurance company, acquired DirtCheap Insurance, another insurance company, and formed the PremierQuotes Group (PQG) to fulfill the growth expectations of its stakeholders (Zimmermann, Tomlinson, and Peuser, 2003). The two merged companies have just consolidated their customer care, contract, and risk management applications. Let us assume that the unified contract application communicates with a customer database and a policy backend to serve end users via three customer self service, agent, and back office channels. Risk management application and policy backend are COBOL applications running on the IBM System z platform. The contract application is a Java Enterprise Edition (JEE) application. Customer care is a software package; its Web application part consists of PHP scripts. An external data source, currently provided by a government information server on the Internet, is integrated, providing crime statistics (fraud history) by geographical area in a proprietary file format.

The three physical tiers are the client tier, the mid tier hosting presentation, domain, and resource (data) access logic, and the backend tier. World-Wide Web (WWW) infrastructure connects the client tier with the mid tier (over the Internet for the customer self service channel and the agent channel, over an intranet for the back office channel). Traditional Enterprise Application Integration (EAI) middleware (e.g., message-oriented middleware) is used to connect the mid tier with the backend tier.

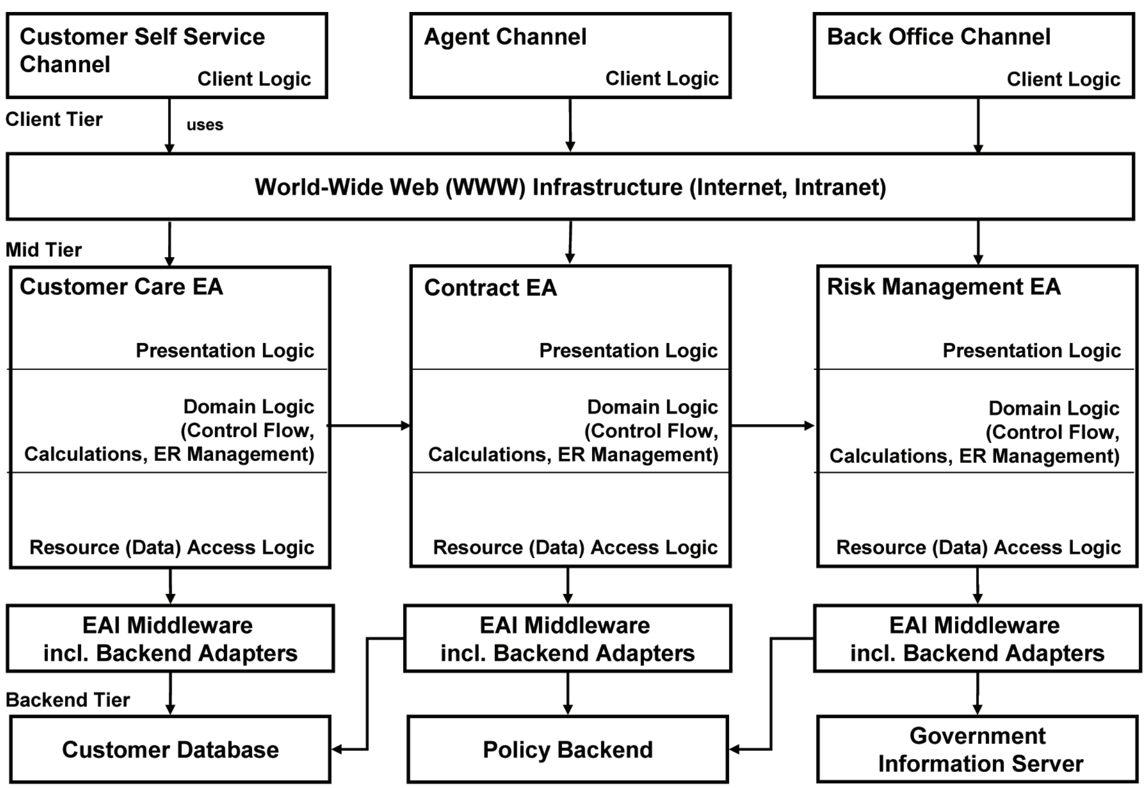
Figure 1 illustrates a representative subset of this Enterprise Application (EA) landscape at PQG.

The client tier contains all application components directly serving the users. Examples are Web browsers and rich client applications running on Personal Computers (PCs) used by customers, agents, and back office staff. The mid tier comprises the three applications. These applications are logically layered into presentation, domain, and resource (data) access logic layers. Typical responsibilities of the mid tier are input validation, processing control, session state management, calculations, and manipulations of enterprise resources. The backend tier stores enterprise resources persistently and coordinates concurrent access to the enterprise resources (i.e., customer profiles, offers, and policies). This tier hosts database servers, but also other systems which in themselves may be physically tiered, but are located external to the company or in another organizational domain. The policy backend and the government information server are examples. Various communication channels exist within and between tiers (Figure 1).

BACKGROUND: APPLYING THE OPEN GROUP ARCHITECTURE FRAMEWORK (TOGAF)

There is a wide range of methods, artifact classification taxonomies, and entire frameworks that can be adapted by enterprise architects to manage enterprise architecture landscapes such as the one at PQG. Prominent examples include The

Figure 1. PQG enterprise applications: System contexts, architecture overview (as-is situation)



Open Group Architecture Framework (TOGAF) (Open Group, 2009), the Department of Defense Architecture Framework (DoDAF) (Department of Defense, 2010), the Enterprise Unified Process (EUP) (Ambler, Nalbone, & Vizdos, 2005), which is an extension of the Rational Unified Process (RUP) (Kruchten, 2003), and the Zachman Framework (Sowa & Zachman, 1992). The Zachman Framework was one of the first of its kind to provide a comprehensive approach for the classification of enterprise architectural artifacts. Numerous proprietary architecture frameworks from commercial (professional) service providers such as consulting firms exist as well.

Among these frameworks, TOGAF can be viewed as representative – or even as a de facto standard – due to its general availability, maturity, comprehensiveness, and wide acceptance in many countries around the world. We applied TOGAF ourselves in a number of enterprise architecture engagements for clients in several industries. Hence, we will focus on the structure and content of the TOGAF 9 framework in this chapter. Our integration concepts are designed to also work with other enterprise architecture frameworks.

TOGAF Overview

TOGAF originally has its roots in the Technical Architecture Framework for Information Management (TAFIM) (Department of Defense, 1996), which focused more on the infrastructure and technical aspects of enterprise architecture. TOGAF 8.1 eventually widened the scope of TOGAF to feature business, data, and application architectures more prominently. At the time of writing, TOGAF 9 was the latest (current) edition (The Open Group, 2009). It includes major enhancements over previous versions to better address state-of-the-art concepts such as iterative architecture development, SOA, specific security architecture considerations, and capability-based planning.

The main components and key concepts in TOGAF 9 are (The Open Group, 2009):

- The Architecture Development Method (ADM) is the core of TOGAF. It describes a step-by-step approach to developing and maintaining an enterprise architecture.
- The ADM guidelines and techniques component contains a collection of blueprints, recommendations, and best practices for applying the ADM. Examples include architectural principles, architectural patterns, governance of SOA initiatives, and migration planning techniques.
- The TOGAF architecture content framework provides a meta-model for architectural content that allows the major architectural artifacts to be consistently defined, structured, and presented. An example of such an artifact is a reusable architecture building block (component) for audit tracking that must be utilized for the development of new business applications.
- Enterprise continuum and tools discusses appropriate taxonomies and tools to categorize and store the outputs of architecture activities within an enterprise and elaborates the concept of an architecture repository. The enterprise continuum supports a very broad architectural scope, covering generic IT system services such as transaction processing, information integration architecture topics, industry-specific architectures like insurance or telecommunications architectures, and enterprise-specific architectures. Thus, the enterprise continuum can be seen as an "umbrella framework" for the TOGAF architecture content framework (which typically focuses on enterprise-specific architectures).
- The TOGAF reference models component introduces two fundamental architectural reference models, namely the TOGAF

Technical Reference Model (TRM), which provides a high-level taxonomy for the description of application software, application platforms, and communications infrastructures, and the Integrated Information Infrastructure Reference Model (III-RM). The III-RM is a reference model for application components to support information integration across domains (e.g. between the sales and invoicing business units of an enterprise).

- An architecture capability framework discusses the organization, processes, skills, roles, and responsibilities required to establish and operate an architecture practice within an enterprise.

As stated in Chapter 2.10 of TOGAF 9, many TOGAF concepts, artifacts and deliverables are generic in order to address a wide variety of enterprises in different industries. TOGAF 9 therefore defines a specific process step for tailoring (Chapter 6.4.5, Select and Tailor Architecture Framework) (The Open Group, 2009). According to our practical experience, TOGAF can be adapted for many functional and technical domains, providing a great deal of flexibility. However, these adaptation and tailoring capabilities require considerable effort (to enhance and refine the TOGAF framework for a specific enterprise).

Architecture Development Method

In contrast to many other enterprise architecture frameworks, TOGAF provides the Architecture Development Method (ADM) as its core component. ADM provides a sound repeatable process for developing architectures. According to Chapter 2.4 of TOGAF 9, this process includes activities such as establishing an architecture framework, developing architecture content, identifying and prioritizing implementation projects, and governing the realization of architectures (The Open Group, 2009). An iterative and incremental development style is explicitly promoted and elaborated in Chapter 19 of TOGAF 9 (The Open Group, 2009).

Figure 2 presents the eight ADM phases (as well as a subset of the artifacts produced in these phases):

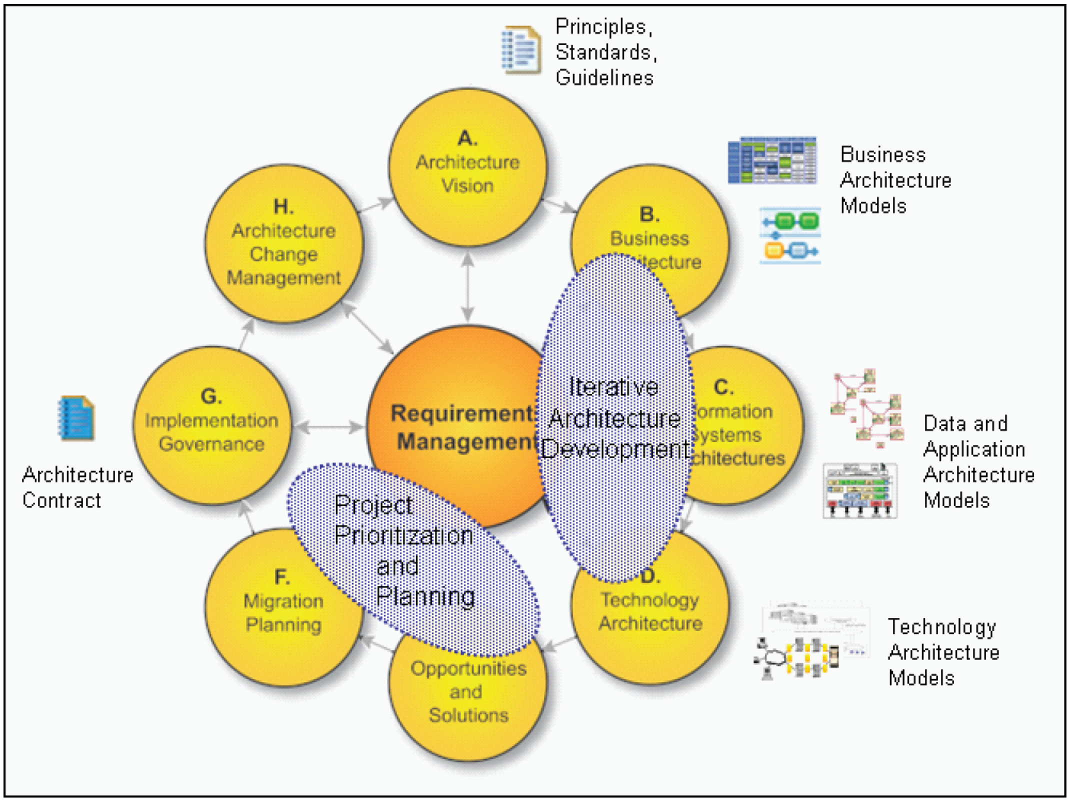
A. Architecture Vision
B. Business Architecture
C. Information Systems Architecture
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
J. Requirements Management

There is an additional preliminary phase which describes the initial preparation and startup activities required to set up a new enterprise architecture process; this phase is not shown in Figure 2. The phases form a cycle (with requirements management as a recurring phase across the entire cycle), which typically is traversed in a highly iterative manner. This cyclic organization of the phases allows enterprise architects to frequently validate (intermediate) results against the original expectations and to introduce new requirements, both on the level of the whole ADM cycle and on the level of a particular phase.

Phase A focuses on scoping decisions regarding the planned ADM cycle, on stakeholder identification, on validation of enterprise principles, standards and guidelines, as well as on the development of an architecture vision. In other words, the business case for the ADM development cycle is defined during this phase.

The phases B, C and D support the development of the business, data, application and technology architecture models. Ideally, traceability between these architecture models and the general enterprise principles and strategies is pursued.

Figure 2. TOGAF ADM with key artifacts and phases (adapted from The Open Group, 2009)



The phases E and F deal with the identification and the planning of projects that support the strategic requirements of the enterprise.

Once important projects from the perspective of the enterprise requirements are defined, the phase G specifies the governance activities for these implementation projects. In this context, the architecture contract is a key deliverable that defines the responsibilities and obligations of both a particular implementation project and the enterprise architecture group. It is discussed in more detail the subsection of this chapter that discusses architectural decisions in TOGAF (see below).

Phase H defines the architecture change management processes required to keep the architecture models current. For example, changes to the business or technology environment or lessons learned from implementation projects are consolidated and requirements for architecture model updates are defined.

TOGAF in Practice

Based on our personal experience with the adoption of TOGAF in various enterprises, we identified a number of recurring challenges. Fundamentally, the ADM is a very useful instrument for establishing the basis for an enterprise architecture management process. It provides a process-oriented view on the typical enterprise architecture activities and defines essential artifacts (work products). This process-oriented view makes the ADM as well as the overall TOGAF framework tangible and therefore eases their deployment. Moreover, the process-oriented view increases the chances of TOGAF for getting accepted by stakeholders such as business domain experts, data architects, and application portfolio managers. However, the rather generic documentation of the ADM and the other key TOGAF components requires experienced subject matter experts both for TOGAF as well as for the specific application domain in order to tailor and refine the framework for concrete usage. The ADM has to be adapted and detailed (just like the other key concepts, artifacts, and deliverables that are intended to be used in a specific enterprise). In order to address these challenges, we recommend conducting a series of TOGAF Adoption Workshops (TAWs*)* involving all key stakeholders. Key objectives of these workshops are to agree on available and to be developed architectural artifacts, on a common understanding of the terminology, on the eligible methods, and on a basic work breakdown structure for the upcoming architecture development activities.

As already outlined in the introduction as an integration/adoption issue, a general risk for the acceptance of enterprise architecture guidelines is that implementers may be reluctant to establish a trust relationship with the enterprise architecture staff. This is a fundamental conflict: enterprise architects set guidelines and constrain projects for the benefit of the whole enterprise; however, projects often do not see the need for such guidelines and constraints because by definition they have a much narrower scope. The classic function of enterprise architecture as a strategic planning tool works quite well in practice (a.k.a. upstream enterprise architecture, with relatively coarse models and focus on migration and transition planning). These functions correspond to TOGAF ADM phases E (opportunities and solutions) and F (migration planning). However, the connection to implementation programs (a.k.a. downstream enterprise architecture) often does not work well according to our experience: solution architects and enterprise architects have difficulties to collaborate effectively, e.g., during ADM phase G. As a consequence, enterprise architecture does not produce the type of information (specifications) that can be readily used by project teams (e.g., architecture and solution building blocks, easy-to-use reference models, etc.). We outlined and exemplified three of the most pressing collaboration issues (i.e., availability, consumability, and enforcement/acceptance issues) in the introduction to this chapter.

Architectural Decisions in TOGAF

We experienced another challenge in our industry projects: guidance is needed how to enforce important architectural decisions on the implementation project level (to ensure consistency with the enterprise architecture principles). In chapter 41.5, TOGAF 9 promotes the management of architectural decisions and proposes to use a governance log to store architectural decisions:

[architectural] decisions made during projects (such as standards deviations or the rationale for a particular architectural approach) are important to retain and access on an ongoing basis. […] Having sight of the key architectural decisions that shaped the initial implementation is highly valuable, as it will highlight constraints that may otherwise be obscured (The Open Group, 2009).

The governance log calls for capturing decisions with rationale, but without any details when and how to do this. This is where solution delivery processes like RUP are supposed to step in on the implementation project level. However, we are not aware of any detailed advice in existing enterprise architecture frameworks and software engineering methods on the types of decisions that typically must be made at a certain step in implementation governance, let alone any guidance for the actual implementation-level decision making.

Let us now have a look at the interfaces between the two roles from a TOGAF ADM perspective. The main interaction activities between enterprise and solution architects occur during TOGAF ADM Phase G (implementation governance). Chapter 15.4 of TOGAF defines phase G as the phase that

guides the implementation from an enterprise architecture perspective. The enterprise architects have to provide architectural oversight for the implementation and ensure that the implementation projects conform to the enterprise architecture. Specifically, the implementation governance phase defines activities and steps to guide development of solutions deployment and perform enterprise architecture compliance reviews (The Open Group, 2009).

We believe that the enterprise architect should not only follow this advice, but also coach and support the solution architects throughout the implementation project – if enterprise architects only conduct reactive compliance reviews rather late in the project, they might struggle to get their review findings accepted as changes already have become costly to implement. An early and continuous involvement helps to ensure that the architectural decisions in an implementation project adhere to the enterprise-level directions throughout the project; it may also serve as a reliable architectural frame that allows agile practices to be applied during construction.

Architectural decisions on the implementation project level that potentially impact the overall enterprise architecture must be made in awareness of enterprise-wide managerial and technical directions. If such decisions are not carefully made, business applications may become difficult to integrate with each other, or even worse, regulatory standards may be violated (as illustrated by the three exemplary design issues from the introduction). As mentioned previously, TOGAF therefore defines the architecture contract deliverable to be developed at the beginning of the implementation governance phase. The architecture contract comprises an agreement between the implementation project partners and the enterprise architects that states the suitability of the developed solution architectures and the project implementation deliverables. According to chapter 49.1 in TOGAF 9, this includes the adherence to the enterprise principles, standards, and requirements of the existing or developing solution architectures, risk management procedures, and a set of processes and practices that ensure proper usage and development of all architectural artifacts (The Open Group, 2009). For instance, the guiding principles and constraints regarding the design options chosen by the solution architect to solve the three exemplary security design issues given in the introduction to this chapter (dealing with data classification and transport channel protection) could be summarized in such architecture contract, providing rationale for option selections.

Enterprise Architecture in the PQG Case Study

Let us assume that an enterprise architect was appointed shortly after the takeover of Dirt Cheap insurance. Having evaluated candidate assets (mostly by screening by studying online resources such as white papers from vendors, consulting firms, and analysts, but also by issuing a Request For Information (RFI) to several architecture consulting firms and evaluating their responses), he/she selects TOGAF as the architecture framework to steer the future evolution of the application landscape at PQG. As a first step towards TOGAF adoption, the ADM is tailored in a TOGAF Adoption Workshop (TAW); this TAW is organized with the help of services provided by of one of the consulting firms that responded to the RFI. Next, in ADM phase A (architecture vision) the goals of the current ADM cycle are defined; in the PQG case, this includes the development of suitable architecture models to identify synergies with the acquired IT systems from DirtCheap Insurance and to assess information integration opportunities. Infrastructure rationalization and consolidation targets (in terms of required cost saving figures per year) are also stated in the architecture vision deliverables.

ADM phases B (business architecture), C (information systems architecture) and D (technology architecture) are conducted next. In these phases, the as-is and to-be architectures are elaborated to the levels of detail that were agreed upon during phase A. Figure 1 from the previous section (system context and architecture overview diagram for as-is enterprise application landscape) is an output of phase C.

During the ADM phase E (opportunities and solutions), gap analyses between the as-is and to-be architectures are conducted and evaluated, resulting in the identification of major shortcomings in the customer enquiry processing of PQG. As a consequence, a recommendation to the PQG Chief Executive Officer (CEO) is made to launch a strategic initiative that improves the customer enquiry processing. The objectives of the initiative are to improve customer service, measured by the conversion rate (i.e., ratio between accepted offers and enquiries processed), and to increase profit by not making an offer if there is a high risk of fraudulent claims. The CEO decides to launch this initiative as it directly supports his business strategy for PQG.

In the following ADM phase F (migration planning) the enterprise architect proposes to the Chief Information Officer (CIO) to launch an application development and integration project, with the goal to develop a new process-centric Customer Enquiry System (CES) which reuses logic from the existing systems to support the strategic business initiative launched in phase E. A lead solution architect for the CES project is appointed, as well as business analysts, a project manager, and a development team.

Let us assume that architectural principles also have been established in the form that is recommended by TOGAF: name-statement-rationale-implications (The Open Group, 2009). For instance, client data such as addresses and accounting information is stored in the customer database and processed by the customer care and the contract management applications (see Figure 1); due to an architectural principle from phase A that all sensitive data has to be secured, this data is classified as Sensitive Personal Information (SPI) that is valued as a strategic corporate asset to be protected against tampering and loss during transport.

RECURRING ARCHITECTURAL DECISIONS AS DESIGN GUIDES

In the previous chapter we motivated the importance of architectural decisions from a TOGAF perspective. Now we approach this topic from an implementation project-centric point of view.

Background and Motivation

Architects make many decisions when creating designs. Both classical and recent books on software architecture (Bass, Clements, & Kazman, 2003; Rozanski & Woods, 2005; Eeles & Cripps, 2010) emphasize how important it is to get the key decisions right. However, it is rather difficult to generalize what the key decisions are, let alone when and how to make them. Therefore, these decisions are often made ad hoc. Architectural knowledge management has become an important research and development topic since 2004 (Kruchten, Lago, & van Vliet, 2006). For instance, decision capturing templates have been published (Tyree & Ackerman, 2005) and modeling tools been prototyped (Ali Babar, Dingsøyr, Lago, & van Vliet, 2009).

Architectural decisions have been characterized as a subset of design decisions that is architecturally significant (Eeles & Cripps, 2010), hard to make (Fowler, 2003a], and costly to change (Booch, 2009). The following definition adopts these themes and adds several qualification heuristics (Zimmermann, 2011):

Architectural decisions capture key design issues and the rationale behind chosen solutions. They are conscious design decisions concerning a software-intensive system as a whole, or one or more of the core components and connectors of such system (in any given view). The outcome of architectural decisions influences the non-functional characteristics of the system such as its software quality attributes.

According to this definition, architectural decisions are made when selecting a programming language, an architectural pattern, an application container technology, or a middleware asset. For instance, integration patterns such as BROKER discuss the many forces distributed systems are confronted with, e.g., location independence and networking issues (Buschmann, Henney, & Schmidt, 2007); these forces qualify as decision drivers. Hence, adding an EAI middleware that implements the BROKER pattern to an architecture is an architectural decision that should be justified and documented in the governance log for the project (see previous section).

Capturing decisions after-the-fact (i.e., retrospectively) has been recognized to be important both by the enterprise architecture community (The Open Group, 2009) and by the software architecture community (Kruchten, 2003); however, many inhibitors such as lack of immediate benefits also have been identified (Ali Babar, Dingsøyr, Lago, & van Vliet, 2009). Relaxing one assumption – documentation rigor – and making a new one – multiple projects in an application genre follow the same architectural style (i.e., they share principles and patterns) – allows graduating architectural decisions from documentation artifacts to design guides:

As an architect specializing on a particular application genre and employing a certain architectural style, I would like to know about the design issues that I have to resolve and the solution options that have been successfully applied by my peers when they were confronted with these design issues – what do they know that I don't know?

After this repositioning from documentation to design, recurring architectural decisions become reusable assets just like methods and patterns. Novel usage scenarios arise. For instance, recurring issues may help to prioritize design and development work items and may serve as checklists during reviews. In this chapter, we investigate how recurring architectural decisions can improve communication between enterprise architects and solution architects. As a first step, let us now identify the architectural decisions required in the case study.

Architectural Decisions in CES Project at PQG (Case Study)

In the beginning of their architecture design work, solution architects should select an appropriate architectural style. Service-Oriented Architecture (SOA) (Zimmermann, 2009) is a state-of-the-art option; a more conservative alternative is to develop three separate three-tier applications (Fowler, 2003) (assuming that both styles have been approved by the PQG enterprise architect).

Many follow-up design issues arise before any of the two top-level design options can be implemented:

Strategic Design Issues

Assuming that SOA is the preferred option, a particular SOA reference model should be selected, which includes agreeing on terminology and identifying relevant pattern languages, and setting technology and product procurement direction. The business strategy (e.g., planned mergers and acquisitions, or divestitures and outsourcing) and strategic IT principles (e.g., to prefer or ban open source assets and to prefer certain software vendors and server infrastructures) must be considered. The architectural principles around security and data privacy from the introduction to this chapter belong to this category of requirements and design constraints. Hence, design guidance from the enterprise architecture level is particularly appreciated in this context.

Conceptual Design Issues

Next, conceptual patterns must be selected and adopted, decomposing the ones that define SOA as an architectural style. All pattern components have to be refined, e.g., the router capability that is a core element of the Enterprise Service Bus (ESB) pattern (Zimmermann, 2009). Functional and non-functional requirements, business rules, and legacy constraints influence the conceptual design work.

In the CES case, the business analyst identified customer care, contract, and risk management services. It is now required to design service providers for these services. The granularity of the service contracts in terms of number of service operations and structure of request and response messages must be decided. Once such service contracts are in place, it becomes possible to design service consumers.

The detailed design and configuration of the ESB triggers another set of concerns: According to the ESB definition in (Zimmermann, 2009), message exchange patterns and formats, as well as mediation, routing, and adapter patterns have to be selected (or banned). In this pattern selection and adoption process, format transformations, security settings, service management (e.g., monitoring), and communications transactionality must be defined precisely.

The service composition design also must be refined if this SOA pattern is selected. The choice of a central process manager implementing workflow concepts as opposed to distributed state management in individual applications is an important architectural concern (Zimmermann, 2009). Other key architecture design issues regarding service composition are where to draw the line between composed and atomic services, how to interface with the presentation layer (in terms of request correlation and coordination), and how to integrate legacy workflows, e.g., those residing in software packages. System transaction boundaries and higher level error handling strategies such as compensation handlers have to be defined as well.

Platform-Related Design Issues

Implementation technologies for the conceptual patterns must be selected and profiled, for instance WS-* technologies (Weerawarana, Curbera, Leymann, Storey, & Ferguson, 2005) or other integration technologies. Once technologies have been chosen, implementation platforms must be selected and configured. Many of the SOA patterns are implemented in commercial or open source middleware assets. It must be decided whether middleware assets should be procured and how the chosen ones should be installed and configured. Performance, scalability, interoperability, and portability are important types of quality attributes when selecting and configuring implementation platforms; enterprise-level guidance regarding strategic vendor preferences and software licensing policies helps to ensure that the decisions made on different projects are consistent with each other, that opportunities for synergies are not missed (e.g., discounts), and that unnecessary costs are avoided (e.g., hidden integration effort introduced by incompatible middleware products).

In summary, PQG has several architecture alternatives to realize CES, including a) SOA or b) three-tiered client-server applications integrated via traditional EAI middleware. Making this decision is only the start of the architecture design; detailed design work follows. Numerous design issues are encountered, which qualify as architectural decisions. The design issues differ substantially depending on the architectural style and patterns chosen. Numerous forces influence the decision making: Quality attributes in categories such as reliability, usability, efficiency (performance, scalability), maintainability, and portability drive the selection of architectural style, the adoption of conceptual patterns, and the design of their platform-specific refinements. Many dependencies exist between the design issues encountered on the CES project, but also from and to those on other projects. Guidance from the enterprise architect is desired.

SOA Decision (SOAD) Modeling

To give recurring architectural decisions a guiding role during architecture design and implementation governance, related project experience has to be captured and generalized in an effective and efficient manner. This is a knowledge engineering activity. SOA Decision Modeling (SOAD) (Zimmermann, Koehler, Leymann, Polley, & Schuster, 2009) is a knowledge management framework that supports such an approach: SOAD provides a technique to systematically identify the decisions that recur when applying a certain architectural style (such as SOA) in a particular application genre (such as enterprise applications). SOAD enhances existing metamodels and templates (Tyree & Ackerman, 2005; IBM Unified Method Framework, 1998) to distinguish *decisions required* from *decisions made*. Platform-independent decisions are separated from platform-specific ones; the alternatives in a conceptual model level reference architectural patterns such as those presented in (Buschmann, Henney, & Schmidt, 2007; Fowler, 2003, Hohpe & Woolf, 2004, Zdun, Hentrich, & Dustdar, 2007). Decision dependency management allows architects to check model consistency and prune irrelevant decisions. A managed issue list guides through the decision making process. To update design artifacts according to decisions made, decision outcome information can be injected into design model transformations (Zimmermann, 2011).

In support of reuse, the SOAD metamodel defines two forms of models:

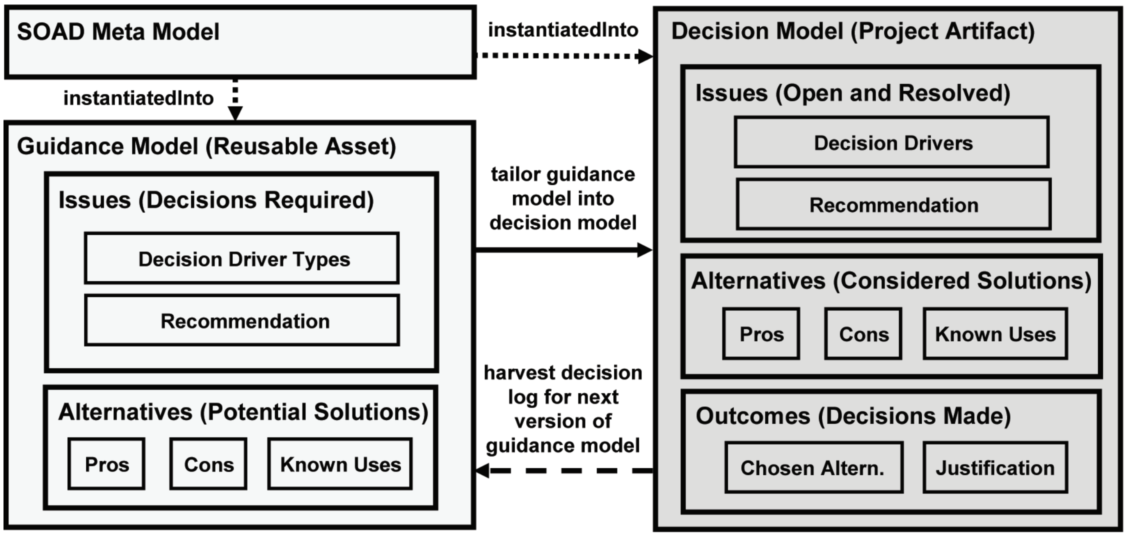
- Guidance models identifying *decisions required* (formerly known as Reusable Architectural Decision Models (RADMs) (Zimmermann, 2009) and
- Decision models logging *decisions made* (formerly known as architectural decision models).

Figure 3 shows the relations and the internal structure of these two types of models (Zimmermann, 2009).

A guidance model is a reusable asset containing knowledge about architectural decisions required when applying an architectural style in a particular application genre. An *issue* informs the architect that a particular design problem exists and that an architectural decision is required. It presents types of decision drivers (e.g., quality attributes and architectural principles) and references potential design alternatives which solve the issue along with their pros (advantages), cons (disadvantages) and known uses (previous applications). It may also make a recommendation about the alternative to be selected in a certain requirements context. Issues and alternatives, authored by a knowledge engineer, use the future tense and a tone that a technical mentor would choose in a personal conversation.

A guidance model captures architectural knowledge from already completed projects that employed the architectural style for which the guidance model is created. Project-specific decision models are created from such guidance models in a tailoring step. Such a tailoring step is conceptually similar to and inspired by method tailoring and adoption activities (e.g., TOGAF adoption as outlined in the previous section); it might involve deleting irrelevant issues, enhancing relevant ones, and adding issues not included

Figure 3. Guidance model and decision model elements



in a guidance model. The guidance model does not have a direct counterpart in TOGAF; it can be seen as an additional artifact in the enterprise continuum.

As shown in Figure 3, a decision model is an architecture documentation artifact that contains knowledge about architectural decisions required, but also captures information about architectural decisions made. A decision outcome is the record (log) of a decision actually made on a project and its justification. Outcomes can be viewed as a form of design workshop minutes and are therefore documented in present or past tense. In a TOGAF context, such decision log forms an important part of the governance log.

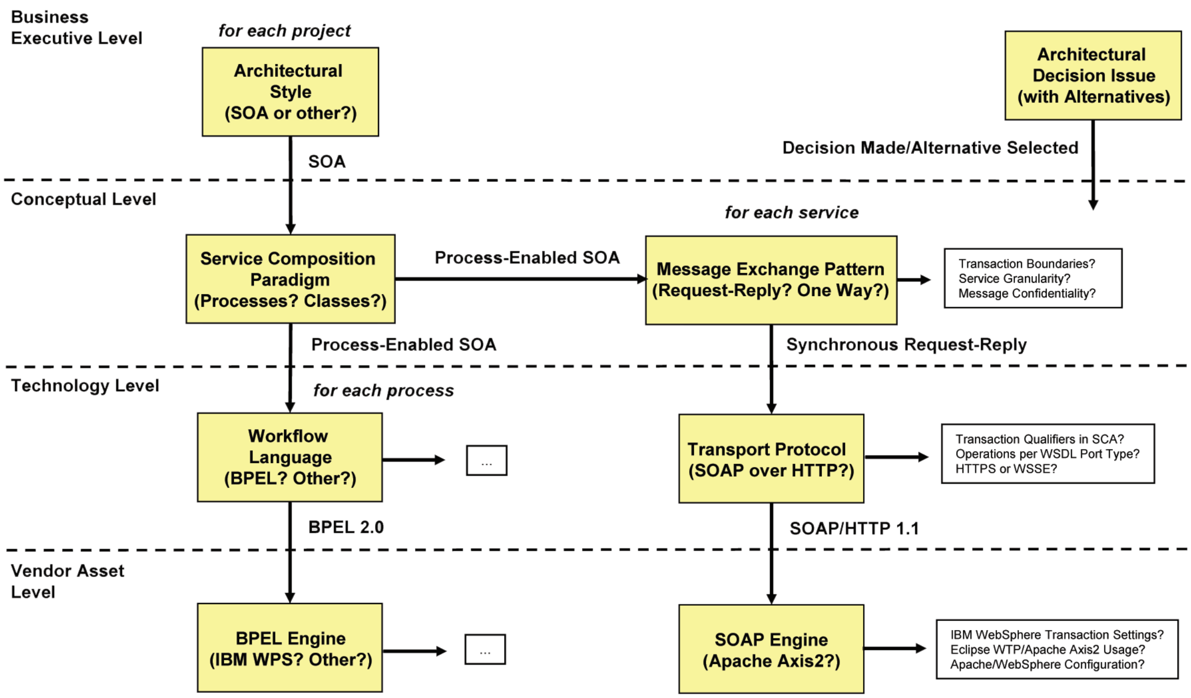
A decision model may reuse one or more guidance models. Information about decisions made can be fed back to the guidance model after project closure via informal or formal lessons learned reviews and/or asset harvesting activities. The required updates to the guidance models can be defined during ADM Phase H, architecture change management. Such guidance model updates have the objective to further improve the breadth, depth, and quality of the architectural knowledge in the enterprise continuum.

Decision Identification and Knowledge Harvesting Activities in SOAD

Guidance model creation activities are described in detail in our previous work, e.g., in Chapter 5 and Appendix A of (Zimmermann, 2009) and Chapter 12 of (Ali Babar, Dingsøyr, Lago, & van Vliet, 2009). We will get back to these activities later in the chapter in the context of the TOGAF ADM and our proposed integration of SOAD into TOGAF.

A particularly comprehensive result of our own guidance modeling activities is a SOA guidance model which comprises about 500 issues with more than 2500 alternatives. The exemplary SOA issues from the previous section (i.e., the strategic design issues, conceptual design issues, and platform-related design issues at PQG) appear in this guidance model in the form of issues and alternatives. Figure 4 outlines the level and layer organization of the guidance model for SOA and positions a subset of the examples from the CES case study as issues (boxes represent issues; alternatives are highlighted by question marks).

Figure 4. SOA guidance model (a.k.a. reusable architectural decision model)



The issues in the SOAD guidance model originate from the author's project experience, input from practitioner communities, as well as the literature; they are meant to be illustrative, not normative here.

Guidance Model Tailoring and Decision Making Processes in SOAD

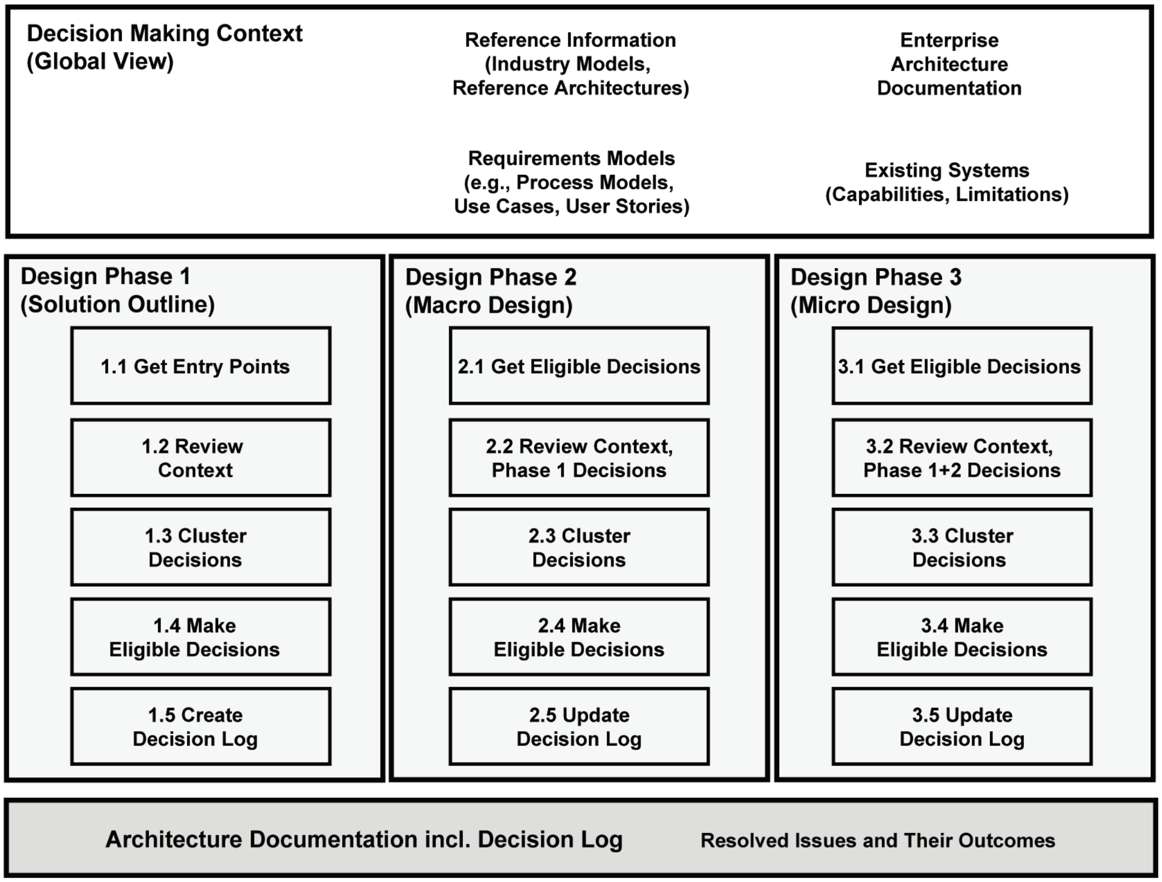
The tailoring of one or more reusable guidance model into the decision model for a project as well as a macro and a micro process for decision making based on guidance models are described in Chapter 7 of (Zimmermann, 2009). We summarize the essence of these two decision making processes in the following subsections.

Macro Process (Project Level)

The SOAD macro process works with a managed issue list (Zimmermann, 2009). We use the phases from the IBM Unified Method Framework (UMF) in this macro process. UMF (IBM Unified Method Framework, 1998) comprises three design phases, solution outline, macro design and micro design; these phases correspond to the RUP phases RUP inception, elaboration and construction. Figure 5 shows the activities to be conducted in these three phases (Zimmermann, 2009).

The decision making context (Hofmeister, Kruchten, Nord, Obbink, Ran., & America, 2007) includes reference information, requirements models, and documentation of the enterprise architecture as well as existing systems, e.g., legacy systems. The enterprise continuum in TOGAF is an example of a repository of such context information.

Figure 5. SOAD macro process for decision making on projects



The output of the macro process is the decision log; it becomes part of the architecture documentation. As explained previously, this decision log becomes part of the TOGAF governance log.

Activity 1.1, 2.1, 3.1

Activities 1.1, 2.1, and 3.1 in our macro design process deal with the retrieval of entry points into the decision making. These activities can be approached in multiple ways. Tacit knowledge or external stakeholder input often guide the architect in the temporal ordering and prioritization of decisions; when decision dependencies between recurring issues are modeled, as suggested and made possible by the SOAD metamodel, tools can assist with this important scoping effort, which defines the focus for the following architecture design work (Zimmermann, 2009).

Activities 1.2, 2.2, 3.2

The second activity in each phase of our macro process is a review activity conducted by the architect. It includes a review of requirements and architectural documentation already available in the decision making context. In solution outline, the review includes *legacy decisions* (i.e., decisions made in another project or pertaining to a different enterprise application). The other project might have been a presales activity, the development of a legacy system a long time ago, or an enterprise architecture project. The decisions made in previous phases of the macro process are also reviewed.

Activities 1.3, 2.3, 3.3

These activities deal with decision clustering. Decisions are rarely made in isolation due to their amount and due to the many dependencies between them. However, it is not obvious how to group and order the decisions that are eligible in a particular macro process phase. Grouping decisions into clusters is typically part of the tacit knowledge of an architect; mature software engineering and architecture design methods provide related advice (Hofmeister, Kruchten, Nord, Obbink, Ran., & America, 2007; Ran & Kuusela, 1996). The actual grouping also depends on the project setup (e.g., methods adopted, human resources available) and on the architects' experience and personal preferences (bias). For instance, one of the authors' rules of thumb is "worst first" (with worst being determined by negative consequences regarding risk, cost, and flexibility).

A decision filtering concept as introduced in Section 7.1 in (Zimmermann, 2009) can be leveraged in addition to tacit knowledge about decision clustering. Due to the formalization of the SOAD meta-model, tools can give clustering advice. However, the architect drives the activity. In SOA design, a tool might suggest to assign all issues about an "ESB router" to be made in the "macro design" phase to an "integration architect". The architect may decide to follow, refine, or overrule this clustering (e.g., by splitting service consumer and provider issues and assigning them to two different integration architects).

Activities 1.4, 2.4, 3.4

These activities instruct the architect to make the decisions that were classified to be eligible in the respective phase. The micro process is launched from this activity once per issue.

Activities 1.5, 2.5, 3.5

As the last activity on the macro level, the decision log is created or updated with the outcome instances created during the execution of the micro process. It becomes part of the project deliverables and, consequently, the TOGAF governance log.

In essence, the managed issue list from SOAD implements the architecture contract required by TOGAF: open decisions form the part of the architecture contract that has not been delivered/satisfied yet (thus listing pending project obligations), whereas made decisions capture completed parts of the contract.

Micro Process (Issue Level)

Figure 6 illustrates the SOAD micro process (Zimmermann, 2009).

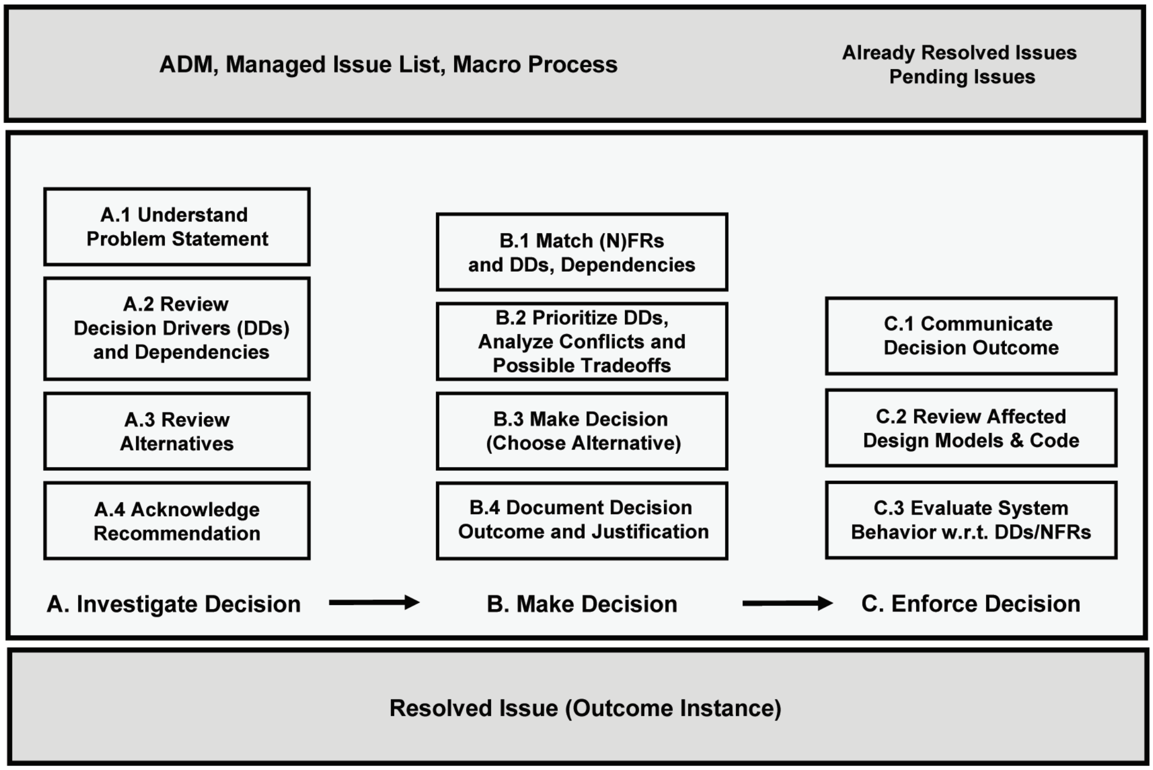
When performing the micro process activities, architects make use of the architectural knowledge in the guidance model and the decision model, which both are structured according to the SOAD metamodel, e.g., listing decision drivers and decision dependencies (see Figure 3).

Step A: Investigate Decision

As a first step, the information about an issue in the ADM must be analyzed; the architects can add missing information. In this step, the problem statement, defined in the SOAD metamodel (Figure 3), must be understood first; if the motivation for the issue remains unclear, the referenced background reading can be consulted (activity A.1).

Next, the decision driver attribute is studied (activity A.2). Like the problem statement, it is an issue attribute; it is reusable, but not project-specific (unless information about actual requirements has been added during tailoring). Hence, it can only list *types* of decision drivers. Still in activity A.2, decision dependencies, particularly those to and from already resolved issues (but also open ones) are investigated.

Figure 6. SOAD micro process for making single decision



The available alternatives have to be considered next (activity A.3). The pros and cons information is particularly relevant; when studying it, the decision drivers and project requirements already considered in A.1 and A.2 are revisited.

The final investigation activity A.4 is to review and acknowledge the recommendation. This does not mean that the recommendation should always be followed. The global decision making context (Figure 5) determines whether this is possible.

Step B: Make Decision

The second step of the micro process is the actual decision making. In activity B.1, the architect matches the actual requirements on the project against the decision drivers (including architectural principles) and decision dependencies investigated in activity A.2. Both functional requirements and Non-Functional Requirements (NFRs) are taken into consideration. Activity B.2 advises the architect to prioritize the decision drivers according to their importance and to analyze potential conflicts and tradeoffs. Before an alternative can be selected, both short term and long term consequences (implications) must be assessed. In many cases, an alternative which may appear to be suited on the micro process level cannot be selected due to certain constraints which are only visible at the macro process or enterprise architecture level (e.g., limitations of legacy systems, existing operations and maintenance procedures). Activity B.3 is to actually make the decision, based on the insight gained during the already completed step A and step B activities.

In Activity B.4, the chosen alternative and the justification for the decision are documented in outcomes. Decision drivers, pros and cons of alternatives, and the recommendation should be referenced in the justification. The justification should not only quote reusable background information such as the types of decision drivers coming from the guidance model, but also refer to actual project requirements (Zimmermann, Schuster, & Eeles, 2008).

Step C: Enforce Decision

The third step of the micro process deals with enforcing the decision. The three activities in this step are to communicate the decision outcome (Activity C.1), to review affected design model elements and code (Activity C.2), and to compare the behavior of the emerging implementations of the system under construction with the decision drivers and actual NFRs including project-specific quality attributes (Activity C.3). It is necessary to re-evaluate on the macro level, as decisions often unveil their full consequences in combination (e.g., decisions that have an impact on end-to-end scalability of the system under construction and on the system performance under heavy load from concurrent users).

Termination of Macro and Micro Process

Macro process and, in turn, micro process continue as long as architectural decision making is still required. More than three phases can be required. It may take a long time to complete the decision making; the managed issue list can even be continued to be used during system operations and maintenance (Sommerville, 1995).

SOAD-Based Decision Making in CES Project at PQG

We identified and informally presented selected design issues in the CES project earlier in this section; as discussed, they arise from the adoption of SOA patterns such as service consumer-provider contract, ESB, and service composition (Zimmermann, 2009).

Figure 7 assigns a subset of these issues to logical components in a SOA reference model; the resulting SOA can be seen as an output of TOGAF phase C, information system architecture. The issues are shown as questions. Several of them appear multiple times, e.g., those about the ESB and those dealing with the three atomic services (customer care service, contract service, and risk management service). This is the case because the respective patterns are applied multiple times in the architecture.

The CES solution architect selects alternatives resolving the open issues based on project-specific requirements. During the SOA design and architectural decision modeling activities, (s)he captures the justifications for their decisions in outcomes, which refer to issues.

Let us assume the CES project to be in the macro design (elaboration) phase; several key decisions have already been made and documented during solution outline (inception). This becomes apparent in Figure 7, e.g., a service composition layer and two ESBs have already been introduced in the architecture.

Table 1 gives more examples for decisions already made, captured as outcomes; the table content is the result of the macro and micro decision making processes introduced in Figure 5 and Figure 6 earlier in this section. The issues and alternatives come from the guidance model for SOA. The sample justifications are specific to the case, referring or paraphrasing PQG/CES requirements.

Refining the previously made decisions, the ones in the following Table 2 proceed from conceptual to platform-specific design.

The table records the output of the partial execution of one phase of the SOAD macro process; each decision, captured in a single table row, is the result of one execution of the SOAD micro process.

Figure 7. Decision identification in PQG case study (Zimmermann, 2009)

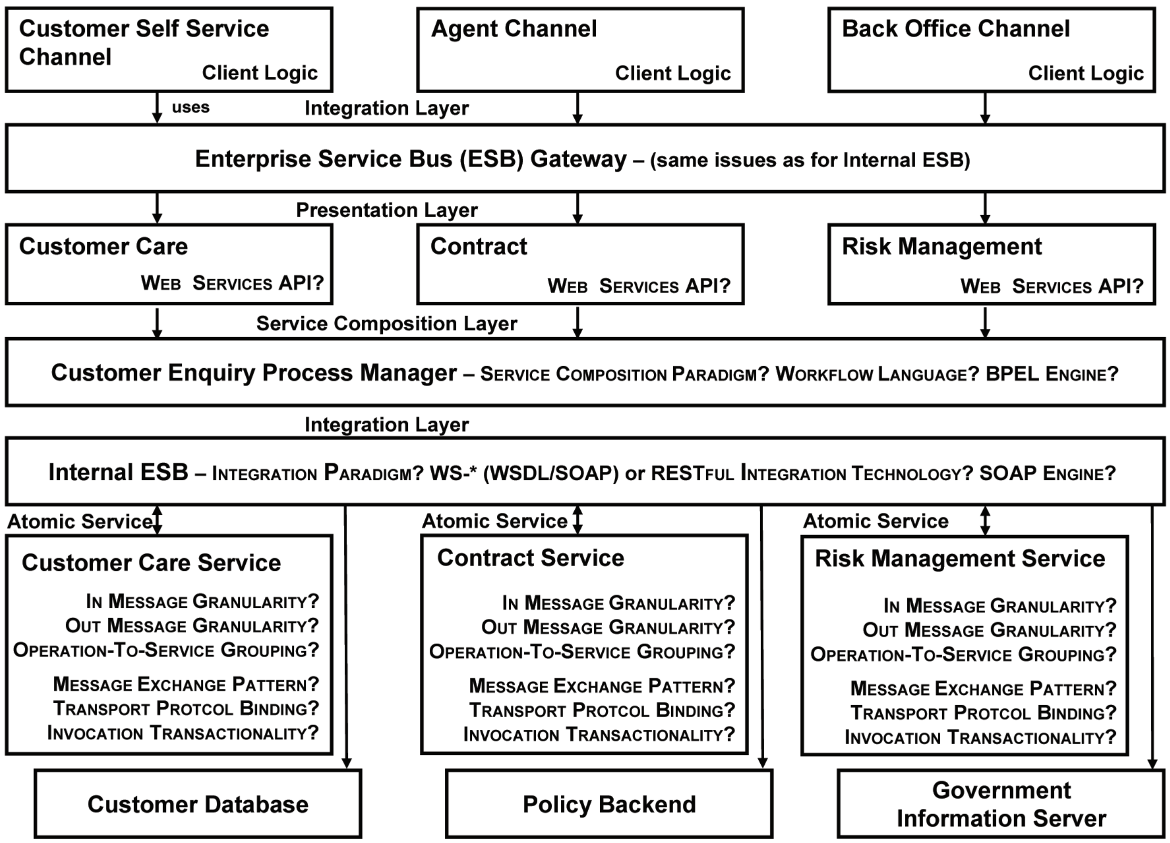


Table 1. PQG case study: Architectural decisions made already

Resolved Issue	Alternative Chosen as Outcome (and Rejected Ones)	Examples of Justifications for Decisions Made for CES (Rationale)
Architectural Style (not shown in Figure 7	SOA Messaging File Transfer, Shared Database, RPC (Hohpe & Woolf, 2004)	Strategic initiative, cross platform integration required and desired, reliability needs
Layering (sketched only in Figure 7)	Layers in SOA Reference Architecture POEAA Layering (Fowler, 2003))	Defined by enterprise architecture team; no industry standard
Integration Paradigm	ESB (Zimmermann, 2009) Traditional EAI, Custom Code	Integration needs (legacy constraints identified in TOGAF as-is models), service monitoring required
Service Composition Paradigm	Workflow (Leymann & Roller, 2000) Human User, Object-Oriented Programming	Long running process, central process manager can preserve integrity across channels (a related business rule has been stated as an architectural principle in the business architecture)
Service Registry	None (UDDI, Vendor Products) (Zimmermann, Tomlinson, & Peuser, 2003)	Only a few services, no business case for a registry yet (according to output of TOGAF ADM phases A to F)

Table 2. PQG case study: Architectural decisions made now

Resolved Issue	Alternative Chosen as Outcome (and Rejected Ones)	Examples of Justifications for Decisions Made for CES (Rationale)
Integration Technology	WS-* Web Services (Zimmermann, Tomlinson, & Peuser, 2003) RESTful Integration (Pautasso, Zimmermann, & Leymann, 2008))	Interoperability and standardization requirements (NFRs), tool support
Workflow Language	Business Process Execution Language (BPEL) (Proprietary Languages)	Standardized (to be preferred according to an architectural principle), used by BPEL Engine selected (see below)
SOAP Engine	IBM WebSphere (Apache Axis2)	Comes with BPEL Engine
BPEL Engine	WebSphere Process Server (Oracle BPEL Process Manager, Active BPEL)	Operational procedures and enterprise license agreement in place (executive decision before project start)

So far, we merely captured decisions already made and their rationale. Table 3 lists additional issues, this time issues still open at the current project stage.

More comprehensive guidance modeling and decision making examples are given in (Zimmermann, 2011), (Zimmermann, Koehler, Leymann, Polley, & Schuster, 2009), and (Zimmermann, 2009), as well as tutorials and other presentation material from the SOAD project (SOAD).

Enforcement of Decisions

In this step, the CES architects create reports about decisions made: The outcome content of Table 1 and Table 2 is exported to a decision log which becomes a part of the TOGAF governance log for the CES project. This artifact is then shared within the technical project team (e.g., other architects, developers, and system administrators) and other stakeholders (e.g., reviewers such as the PQG enterprise architect). The made decisions are executed, e.g., through procurement, installation, and configuration of the selected BPEL Engine and through BPEL and Java development activities.

Table 3. PQG case study: Architectural decisions still required

Open Issue	Alternatives	Decision Drivers (Guidance Model for SOA)
In Message Granularity	Dot Pattern Dotted Line Pattern Bar Pattern Comb Pattern	Structure and amount of enterprise resources to be exchanged, message verbosity, programming convenience and expressivity, change friendliness
Operation-To-Service Grouping	Single Operation Multiple Operations	Cohesion and coupling in terms of security context and versioning
Message Exchange Pattern	One Way Request-Reply	Consumer semantics and availability needs, provider up times
Transport Protocol Binding	SOAP/HTTP SOAP/JMS Plain Old XML(POX)/ HTTP	Provider availability, data currency needs from consumer's perspective, systems management considerations
Invocation Transactionality	Transaction Islands Transaction Bridge Stratified Stilts	Resource protection needs, legacy system interface capabilities, process lifetime, enterprise-level guidelines regarding system operations (e.g., regarding error handling and auditing/archiving policies)

ARCHITECTURAL DECISION MODELING AND MAKING IN TOGAF

In this section, we show how to overcome the controversy and gap between enterprise architects and solution architects, with an enterprise architect leading guidance model creation and solution architects contributing to them and using them (continuous improvement cycle).

Issues, Controversies, Problems in CES Project at PQG

The current status in the PQG case study is that the enterprise architect, following the TOGAF ADM, has established architectural principles and defined the scope for the current ADM cycle in phase A. He/she created as-is architecture models providing an inventory of existing systems and outlining to-be target architectures during the architecture development phases B to D. The output of the ADM phases E and F have triggered the CES development project.

The CES solution architecture has to meet specific project requirements to satisfy the CES project sponsor and end users, but also to adhere to the architectural principles established and enforced by the PQG enterprise architect. Several key decisions have already been made. A SOAD guidance model was not available, all issues and alternatives had to be documented by the solution architect as part of the CES project. Many of the decision outcomes referenced architectural principles in their rationale (justification attribute). This is a budget challenge for the individual project; the connection to enterprise architecture artifacts is not obvious and not tangible. Moreover, each project makes its decisions without knowing about those on other projects (past or present). Let us assume that a high-level SOA reference model has been created, which defines a layering scheme, but is not detailed enough for project development work.

To overcome the outlined governance problems, the PQG enterprise architect initiates a guidance model creation effort (project or working group), with the objective to make experience with – and knowledge about – the consolidation and modernization of enterprise applications (specifically when using the SOA style) explicit so that this knowledge can take an active guiding role on projects like CES.

Extending TOGAF with SOAD Guidance Modeling Concepts

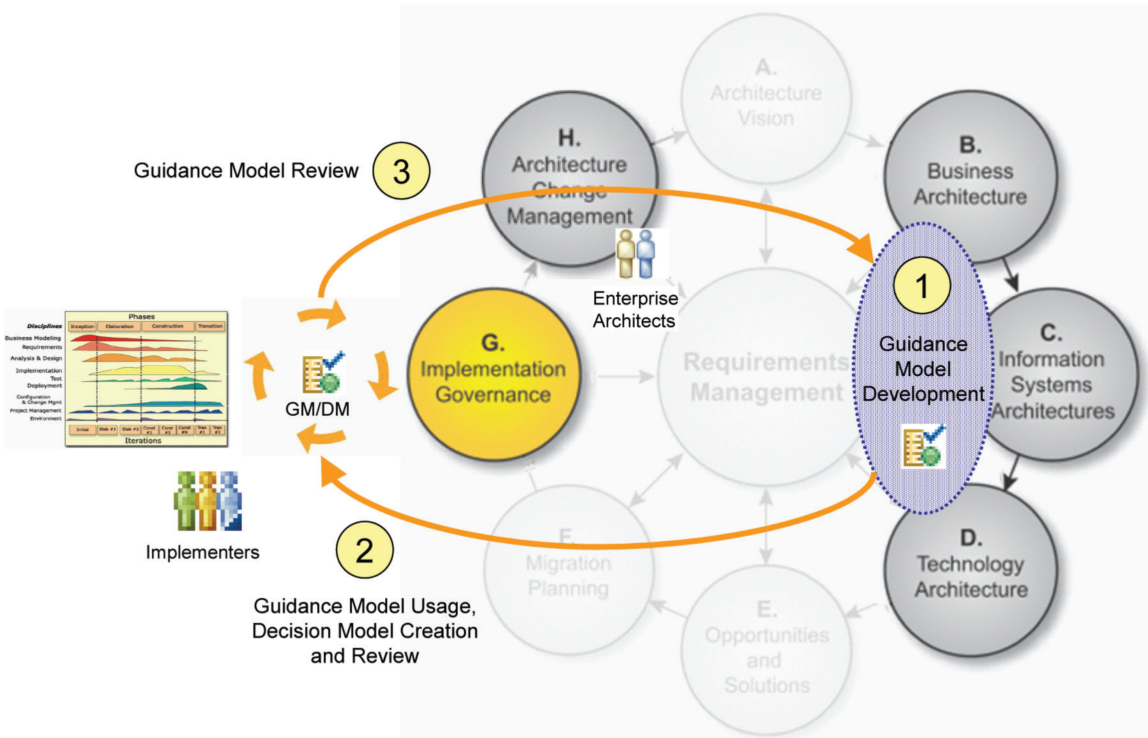
This section presents the core contribution and novelty of the chapter, an integration of the SOAD concepts into TOGAF. Figure 8 illustrates our overall integration approach by mapping TOGAF ADM phases to SOAD guidance modeling and decision making activities.

The primary integration point with several intense interactions is TOGAF phase G, implementation governance. These interactions will be covered in more detail later in the section.

Faithful to the iterative and incremental nature of ADM, we introduce a continuous improvement cycle:

1. **Guidance Model Development (Creation and Update):** The enterprise architect provides a guidance model, with the objective to support and promote the usage of enterprise architecture models and guidelines (e.g., architectural principles) on the project level.
2. **Guidance Model Usage, Decision Model Creation and Review:** The solution architect uses the guidance model to steer the architecture design work on a project (e.g., when creating decision models/logs) and provides feedback on the relevance and consumability of the architectural knowledge found in the guidance model. The enterprise architect uses the decision model to review the evolving implementation architecture for

Figure 8. Guidance modeling and decision making in the ADM process in TOGAF



general fitness (adequateness) and compliance with the enterprise-level guidelines and constraints.

3. **Guidance Model Review:** The enterprise architect updates models and guidelines accordingly (with additional input from the solution architect community).

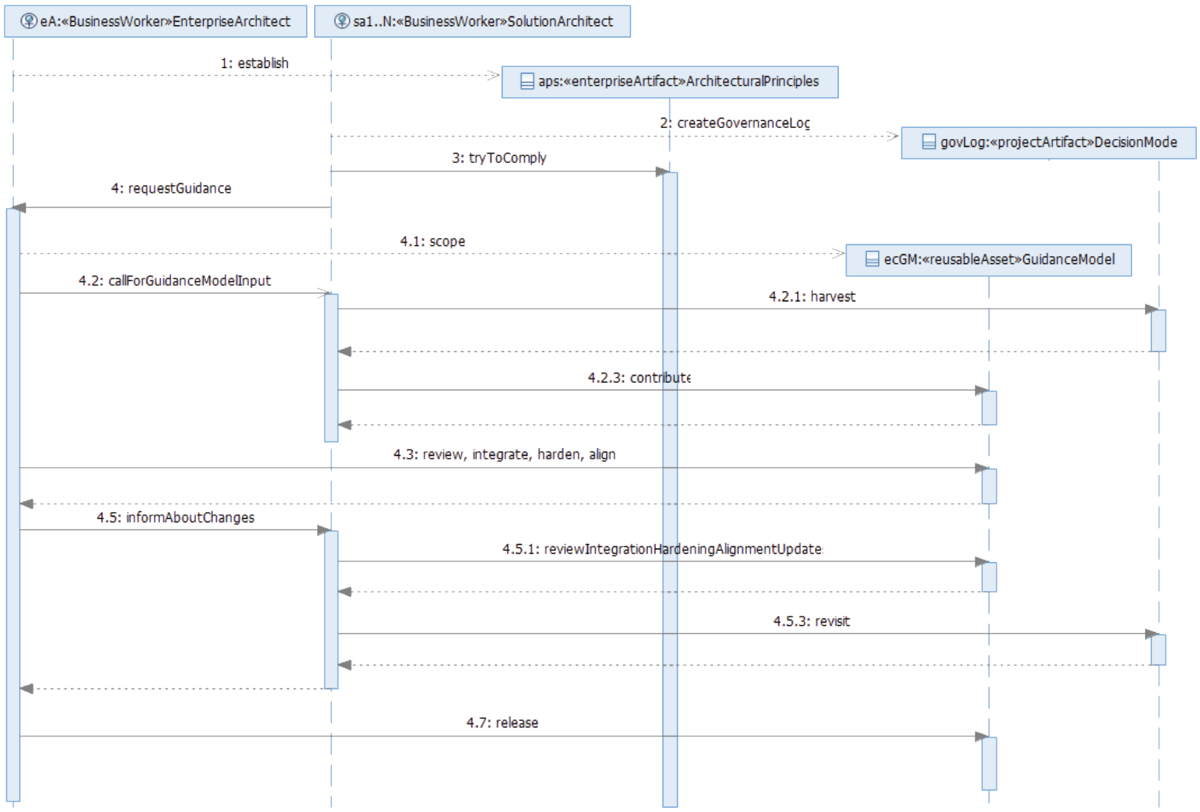
1. Guidance Model Development

The first integration point is guidance model creation (a.k.a. knowledge engineering or harvesting), taking place (i.e., positioned and attached to) in TOGAF phases B, C, and D. To realize this integration point (step), we propose a collaborative approach (i.e., a series of fine grained interactions between enterprise architects and solution architects). Figure 9 details this collaborative approach in the form of a UML interaction diagram.

Key initial inputs to the guidance model creation work (i.e., the early scoping activities) are architectural principles, enterprise architecture models and artifacts, enterprise blueprints and standards, reusable assets, and existing documentation of architectural decisions from implemented solutions. These inputs are ideally classified using a comprehensive taxonomy (e.g., based on the TOGAF content framework) and obtained from the organization's enterprise continuum.

Regarding depth and breadth of a guidance model, its creators have a choice between making comprehensive knowledge packs available and lightweight approaches; a basic form of an initial guidance model is a checklist with questions and possible answers for solution architects, or a simple decision tree such as the 10-node cloud buyer guide from the Open Group (Open Group, 2010).

Figure 9. Guidance model creation (knowledge harvesting in TOGAF phases B, C, and D)



During the guidance model creation activities, enterprise architects can start with an existing guidance model such as the SOA one outlined in the previous section. Possibly, it will be required to add new levels and topic groups to adjust the structure of the guidance model for a particular enterprise. Certain fundamental issues in the guidance model may be marked with a tag like "enterprise architect involvement/review particularly important and required" (e.g., the issues for which guidance was requested in the first place, those with severe long term consequences, e.g., regarding operations and maintenance, or those with implications for multiple lines of business).

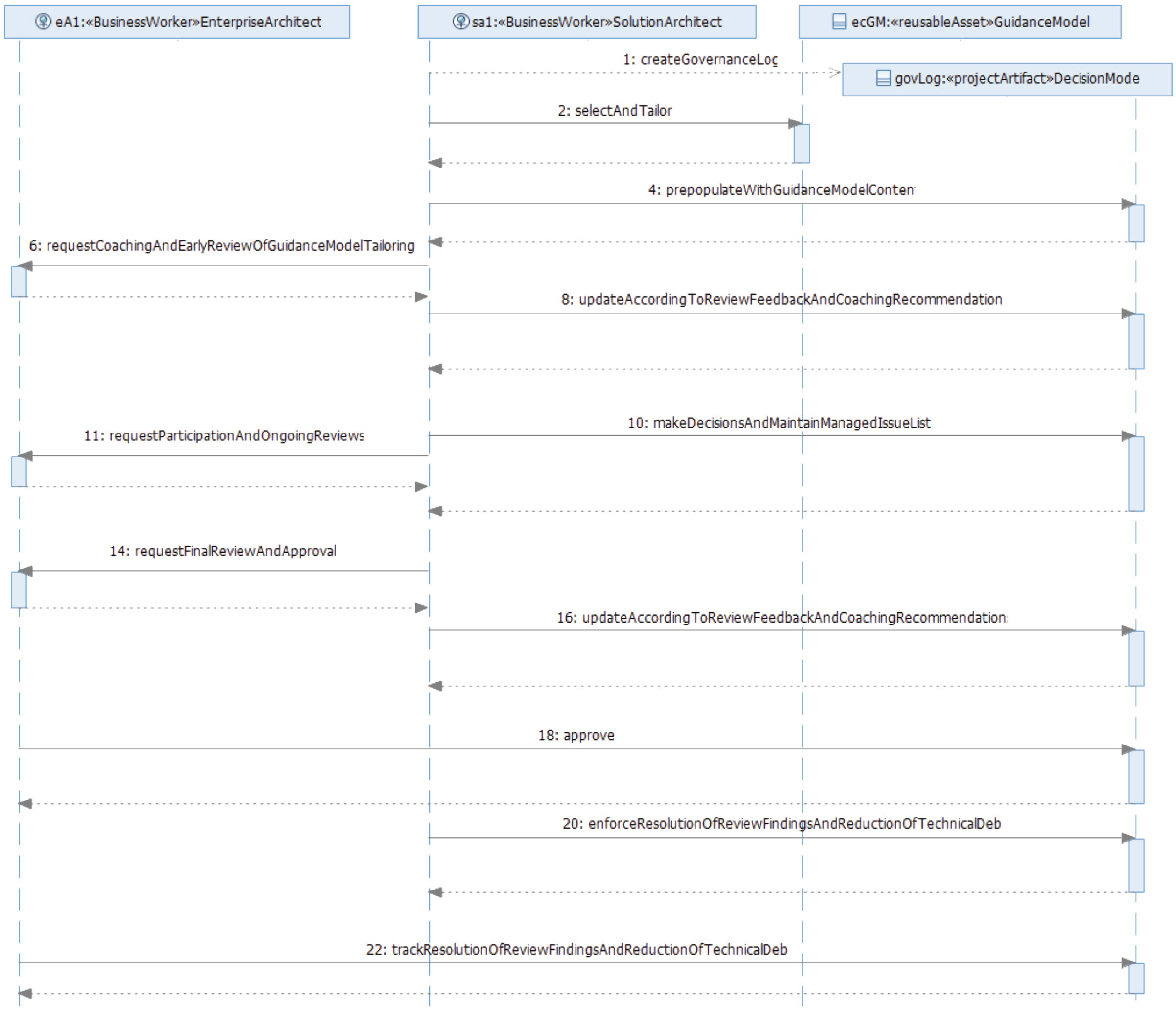
In a practical application of our TOGAF-SOAD integration concepts, each of the interactions can be supported by reusable assets such as mail templates (e.g., *callForGuidanceModelInput*, *release*), wiki pages (e.g., *contribute*), and predefined/-populated questionnaires (e.g., *requestGuidance*). Furthermore, activity owners and activity initiation triggers should be defined to ensure timely and diligent execution.

See existing work (Zimmermann, 2009) and previous section for additional information, e.g., about our experience with the review-integrate-harden-align steps, shown as a single activity in Figure 9.

2. Guidance Model Usage

Having covered the decision harvesting activities (i.e., guidance model creation and review), the following Figure 10 focuses on decision making in TOGAF phase G (again in the form of a UML interaction diagram).

Figure 10. Decision making in TOGAF phase G (implementation governance)



The interactions in the figure detail the involvement of the enterprise architect during the SOAD macro and micro processes that we introduced in the previous section.

If solution architecture requirements have to be satisfied that require enterprise architecture artifacts (models) that do not exist yet, guidance modeling activities may be triggered (step 1). More agile approaches may also be applied, e.g., a temporary involvement of the enterprise architect on the solution development project in the form of architectural decision making workshops. The minutes (protocols) of these workshops then may serve as initial versions of future guidance models.

Our existing work and the previous section provide additional information, e.g., about tailoring (Zimmermann, 2009).

3. Guidance Model Review

The review activities have to be defined in detail when implementing the interlock between the enterprise architect and the solution architect. For instance, it has to be specified whether regular proactive/periodic reviews and content update cycles are planned.

We foresee a continuum of modes of operation and review rigor: two ends of the spectrum are a conservative process with funded design authorities and formal approvals on the one end and an opportunistic approach solely relying on volunteers (e.g., Web 2.0 crowd sourcing) on the other end. The maturity of the owning organization, amount of executive-level support and budget, and company culture are among the decision drivers for this design issue pertaining to the guidance modeling process.

The interactions during an update step are identical to those performed during guidance model creation, resulting in a new version of the guidance model (see Figure 9). During TOGAF phase H, the requirements for guidance model updates are consolidated. In the following phase A of the next TOGAF cycle, such updates (or a subset that addresses high-priority topics) may be planned to be included this cycle. They may then be implemented during the subsequent architecture development phases B, C, and D.

Initial Guidance Model Content

We propose the following candidate issues for a guidance model supporting decision making on implementation projects:

- Design, adoption, and rollout of governance and design processes as well as supporting notations and tools (e.g., UML modeling versus architecture description language versus other domain-specific language, possibly different for each stakeholder viewpoint (Küster, Völzer, & Zimmermann (2011)). Refer to TOGAF Chapter 48.3, architecture compliance reviews (The Open Group, 2009).
- Top-level functional slicings of responsibilities both in the organization and in the IT systems, e.g., business domain concept in SOA (Krafzig, Banke, & Slama, 2005) and strategic domain-driven design (Landre, Wesenberg, & Rønneberg (2006).
- One particularly important topic group is the question when to prefer build over buy (e.g., to achieve a competitive advantage or to avoid hidden, uncontrollable integration efforts (Wesenberg, Landre, & Rønneberg (2006)), even if a general architectural principle exists to prefer software procurement and customization over custom development.
- Evaluation criteria for and selection of software packages (typically per business component or functional area) and SOA middleware whose purchase implies significant licensing cost and/or training and operations effort (e.g., workflow engine, enterprise service bus).
- Decisions about selection of open source software as well as other reusable assets (both private and public ones) and about development of company-internal solution building blocks.
- Information management decisions with an impact on the degree to which a solution adheres to relevant data privacy laws, audit compliance rules and company-internal security standards (see introduction to this chapter for examples).
- Decisions on required refactorings of existing systems to match enterprise architecture guidelines.
- For strategic system maintenance or enhancement projects, how to identify the technical debt to be reduced (and how to do so).

Note that the candidate issues are "TOGAF ADM phase G-specific" decisions; decisions about the enterprise architecture itself are not included yet. The above collection does not aim

Table 4. PQG case study; exemplary mappings from TOGAF ADM phases to SOAD activities (by role)

TOGAF Phase	Role	Activity (with Supporting Tools and Notations)
A	Enterprise architect (PQG)	Establish first version of architectural principles
	Solution architects (CES, other)	Review architectural principles
B, C, D	Enterprise architect (PQG)	Scope guidance model (SOAD tool) Call for guidance model input (via email to community, via wiki or other social networking/collaboration tool)
	Solution architects of existing applications (customer care, contract management, risk management)	Harvest decision logs from previous projects (review tool) Contribute to guidance model (via template, via copy-paste)
	Enterprise architect (PQG)	Review, Integrate, Harden, Align (RIHA) (SOAD method and tool) Inform solution architect about changes
	Solution architect (existing applications)	Review RIHA updates Revisit decision logs from projects Provide feedback to enterprise architect
	Enterprise architect (PQG)	Incorporate review feedback Release first/subsequent versions of SOA guidance model
E, F	Enterprise architect (PQG)	Identify potential projects that directly support the enterprise strategy Create a high-level roadmap and project plan for those projects Prioritize and select suitable projects Plan and prepare adoption of SOAD method and tool for selected projects
	Solution architects	n/a
G	Solution architect (CES)	Create governance log (SOAD decision model) Try to comply with architectural principles Request guidance for strategic design issues, SOA pattern selection and adoption, technology and product platform preferences Select and tailor SOA guidance model (SOAD tool) Pre-populate decision log with guidance model content Request enterprise architect participation and ongoing reviews Make and document decisions and maintain managed issue list (supported by SOAD macro and micro process and supporting tool) Update decisions according to review feedback (SOAD tool) Enforce correct implementation of decisions made
	Enterprise architect (PQG)	Review governance log/decision model (ongoing)
	Solution architect (CES)	Request final review and approval (e.g., at the inception and elaboration phase milestones of the implementation project (Kruchten, 2003)
	Enterprise architect (PQG)	Approve decisions
	Solution architect (CES)	Enforce and track resolution of review findings and reduce technical debt
H	Solution architect (CES)	Provide feedback regarding use of guidance model and additional architectural knowledge gained on CES project
	Enterprise architect (PQG)	Plan guidance model updates

at being complete; as a rule of thumb, all strategic solution decisions that require involvement of enterprise architects can/should be included eventually. In fact, all executive decisions in the taxonomy established by Kruchten, Lago and van Vliet (Kruchten, Lago, & van Vliet, 2006) benefit from enterprise-level decision making guidance.

Application of TOGAF-SOAD Integration Concepts at Premier Quotes Group

Table 4 lists the guidance modeling and decision making activities at PQG per TOGAF phase. The table also comments on notations and tools that are suited for certain activities.

FUTURE RESEARCH DIRECTIONS

Architectural Decisions and Agile Practices

An important area of future work is to investigate architectural decision making in the context of agile practices.

The literature on agile practices typically focuses on process aspects (e.g., ceremonies in Scrum (Sutherland) rather than design advice, although the original article introducing Scrum (Schwaber, 1995) mentions architecture work to be a key part of the project start phase (happening before any sprint). The notion of a sprint/iteration 0 has also been proposed (Ambler, 2009). However, it remains unclear when and how to (pre-)populate the decision backlog both for iteration 0 and for following iterations. Lean software development promotes the principle of deferring decisions until the last responsible moment (Poppendieck & Poppendieck, 2003); however, it remains unclear when this moment has come.

In our previous work, we have developed the notion of a managed issue list serving as a decision backlog (Zimmermann, 2009); this decision backlog can also be seen as a particular subset of the Scrum product backlog (featuring open design issues as a new type of backlog entry). We envision the processing of a decision backlog to steer the design work on implementation projects. Such decision backlog can highlight and prioritize the issues that have a particular relevance from an enterprise architect perspective.

Recent work by Fairbanks is particularly relevant the context of an agile governance log; he suggests an architectural haiku, a short architecture description specifically designed and compacted/comprised for agile project teams (Fairbanks, 2011). The haiku provides a short and concise syntax for capturing decision rationale:

<Driver-x> is a priority, so we chose design <Alt-y>, accepting downside <Cons-z>

The variables x and z represent instances of quality attributes or other decision drivers here, including architectural principles; y combines an issue with an alternative. We envision similar Haikus, written in the future tense, to be suited for the development of agile guidance models.

TOGAF Updates

One could also consider extending TOGAF to give architectural decisions an even more prominent place, similar to the overarching “über-phase” requirements management in the center of the ADM. Such effort would require significant changes to the existing TOGAF practices and their documentation and is therefore out our scope for the time being.

TOGAF could also provide pre-populated guidance models for particular domains such as SOA or cloud computing. Such guidance models could complement and accompany TOGAF reference models or reference architectures for these architectural styles and technical domains; they would fit into the TOGAF architecture content framework as well as the enterprise continuum.

CONCLUSION

Architectural decisions make or break a project – whether made consciously, subconsciously, or by 3rd parties like technology thought leaders (or software vendors). It is essential to identify, make, and communicate the key ones adequately; their rationale should be preserved. Capturing these decisions after the fact is a labor-intensive undertaking with many long-term, but few short-term benefits. In practice, this important documentation task is often neglected for this reason.

TOGAF is a state-of-the-art architecture framework; an architecture development method, a comprehensive collection of guidelines and techniques and the concept of the enterprise continuum are three of its key components. Tailoring TOGAF to provide tangible advice for solution architects and other technical decision makers is a challenge in practice. For instance, the structure of the content of the governance log for phase G (implementation governance) is not defined in detail and no processes or tools for creating and maintaining the log exist. Pre-defined governance log content for certain architectural styles or technology domains such as SOA or cloud computing does not exist either.

Many important decisions are encountered and solved repeatedly on multiple projects; it is therefore desirable to share related architectural knowledge between these projects. Hence, our previous SOAD work introduced guidance models as reusable assets compiling the design issues and options that will occur whenever a certain architectural style is applied in an application genre. SOAD was originally created to support enterprise application and Service-Oriented Architecture (SOA) design, but is also applicable to other application genres and architectural styles. In this chapter, we investigated how to use SOAD as governance instrument to improve the communication and knowledge exchange between enterprise and solution architects. To support this usage scenario, SOAD promotes the reuse of architectural knowledge by compiling recurring issues and options in guidance models.

The integration of SOAD into TOGAF that we presented in this chapter can be summarized as:

- Both enterprise and solution architects make decisions; solution architects expect guidance regarding a subset of their decisions from the enterprise architects. We observed this situation repeatedly on projects in various industry sectors, including financial services, telecommunications, and automotive.
- Architectural principles are established through a decision making process; once they exist, they become decision drivers and justifications for subsequent decisions.
- Guidance models may be created during TOGAF phases B to D. As reusable assets, the guidance models become part of the enterprise continuum. They are tailored into decision models and then used in TOGAF phase G (implementation governance). Requirements for guidance model updates are consolidated in phase H (architecture change management).
- The SOAD decision log becomes an integral part of the TOGAF governance log; decisions required and decisions made (maintained in the managed issue list in SOAD) form an important part of the architecture contract between enterprise architects and solution architects. The fulfillment of this contract can be monitored by observing the managed issue list and the decision log.
- The macro and micro process for decision making in SOAD is integrated into ADM phase G (implementation governance); UML interaction diagrams specify the collaborations between solution architects and enterprise architects during these processes and guidance model creation.

The result of a guidance modeling effort and a SOAD-TOGAF integration effort is a guidance model that serves as a "virtual enterprise architect" preserving formerly tacit knowledge. This relieves the real enterprise architects from routine work so that they can focus on hard design problems – or consider transitioning to implementation projects or other assignments for a certain amount of time.

We developed and evaluated the presented approach on real-world architecture consulting engagements. On these engagements, certain limitations of the presented approach became apparent. These limitations can be categorized under 1) usage prerequisites, 2) motivation issues/potential inhibitors and 3) guidance model maintenance. As for 1), we assume familiarity with and use of two rather rich and comprehensive assets, TOGAF and SOAD. At least the motivation and budget for training have to exist. Regarding 2), is has been reported that many knowledge sharing systems fail to work in practice because people feel threatened to share their knowledge; hence, architects have to be encouraged to share their knowledge within their organizations. Personal incentives are one way of doing so, e.g. tokens of appreciation (such as informal of formal knowledge management awards); ownership of or contributions to guidance models may also become a criterion that has to be met before an architect is promoted to a higher level of seniority. 3) Defining and funding a sustainable approach to guidance model management over time remains a challenge; for a more detailed discussion and solution approaches that address this governance challenge, we refer the reader to our previous publications (Miksovic & Zimmermann, 2011; Zimmermann, Koehler, Leymann, Polley, & Schuster, 2009).

Our TOGAF-SOAD integration solution allows enterprise architects and solution architects to improve their communication and the knowledge exchange between the two communities; the availability, consumability, and enforcement/acceptance issues that we observed in practice can be resolved (or at least relieved and mitigated) this way. The integrated approach allows enterprise and solution architects to share best practices recommendations in a problem-solution context:

We learn best from mistakes – but who said all these mistakes have to be our own ones?

REFERENCES

Ali Babar, M., Dingsøyr, T., Lago, P., & van Vliet, H. (2009). *Software architecture knowledge management: Theory and practice*. Springer-Verlag. doi:10.1007/978-3-642-02374-3

Ambler, S. (2009). *Agile model driven development (AMDD): The key to scaling agile software development*. Essay available online.

Ambler, S., Nalbone, J., & Vizdos, M. (2009). *The enterprise unified process: Extending the rational unified process*. Prentice Hall.

Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice* (2nd ed.). Addison Wesley.

Booch, G. (2009). *AoT presentation.* IBM internal.

Brown, N., Nord, R., & Ozkaya, I. (2011). *Strategic management of technical debt*. WICSA 2011 Tutorial.

Buschmann, F., Henney, K., & Schmidt, D. (2007). *Pattern-oriented software, Vol. 4 – A language for distributed computing*. Wiley.

Department of Defense. (1996). *Technical architecture framework for information management* (*Vol. 1*).

Department of Defense. (2010). *The DoDAF architecture framework,* Version 2.02, 2010. Retrieved from http://cio-nii.defense.gov/sites/dodaf20/index.html

Eeles, P., & Cripps, P. (2010). *The process of software architecting*. Addison-Wesley.

Fairbanks, G. (2011). *Architecture Haiku.* WICSA 2011 tutorial. Retrieved from http://rhinoresearch.com/files/Haiku-tutorial-2011-06-24-final.pdf

Fowler, M. (2003a). *Patterns of enterprise application architecture*. Addison Wesley.

Fowler, M. (2003b). Who needs an architect? *IEEE Software*, *20*(5). doi:10.1109/MS.2003.1231144

Haischt, D., & Georg, F. (2010). *Get me approved, please! Lizenzkompatibilitaet von Open-Source Komponenten. Objektspektrum, Sonderbeilage Agilitaet, Winter 2010*. SIGS Datacom.

Hofmeister, C., & Kruchten, P., Nord, Obbink, J. H., Ran, A., & America, P. (2007). A general model of software architecture design derived from five industrial approaches. *Journal of Systems and Software*, *80*(1), 106–126. doi:10.1016/j.jss.2006.05.024

Hohpe, G., & Woolf, B. (2004). *Enterprise integration patterns*. Addison Wesley.

IBM. (2009). *Unified method framework: Work product description ARC 0513 (Architectural Decisions)*. IBM Corporation.

Julisch, K., Suter, C., Woitalla, T., & Zimmermann, O. (2011). Compliance by design – Bridging the chasm between auditors and IT architects. *Computers & Security*, *30*(6-7). doi:10.1016/j.cose.2011.03.005

Krafzig, D., Banke, K., & Slama, D. (2005). *Enterprise SOA*. Prentice Hall.

Kruchten, P. (2003). *The rational unified process: An introduction*. Addison-Wesley.

Kruchten, P., Lago, P., & van Vliet, H. (2006). Building up and reasoning about architectural knowledge. *Proceedings of QoSA 2006, LNCS 4214*, (pp. 43-58). Springer.

Küster, J. M., Völzer, H., & Zimmermann, O. (2011). Managing artifacts with a viewpoint-realization level matrix. In Avgeriou, P., Grundy, J., Hall, J. G., Lago, P., & Mistrik, I. (Eds.), *Relating requirements and architectures*. Springer-Verlag. doi:10.1007/978-3-642-21001-3_15

Landre, E., Wesenberg, H., & Rønneberg, H. (2006). *Architectural improvement by use of strategic level domain-driven design*. OOPSLA Companion. doi:10.1145/1176617.1176728

Leymann, F., & Roller, D. (2000). *Production workflow – Concepts and techniques*. Prentice Hall.

Miksovic, C., & Zimmermann, O. (2011) Architecturally significant requirements, reference architecture and metamodel for architectural knowledge management in information technology services. *Journal of Systems and Software*, *85* (9), 2014–2033, doi: http://dx.doi.org/10.1016/j.jss.2012.05.003

Pautasso, C., Zimmermann, O., & Leymann, F. (2008). RESTful web services vs. Big web services: Making the right architectural decision. *Proceedings of WWW*, *2008*, 805–814. ACM. doi:10.1145/1367497.1367606

Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: An agile toolkit*. Addison Wesley.

Ran, A., & Kuusela, J. (1996). Design decision trees. *Proceedings of 8th International Workshop on Software Specification and Design, International Workshop on Software Specifications & Design,* (pp. 172-175). IEEE Computer Society.

Rozanski, N., & Woods, E. (2005). *Software systems architecture: Working with stakeholders using viewpoints and perspectives*. Addison-Wesley.

Schwaber, K. (1995). Scrum development process. *Proceedings of OOPSLA'95 Workshop on Business Object Design and Implementation*.

Sommerville, I. (1995). *Software engineering* (5th ed.). Addison Wesley.

Sowa, J. F., & Zachman, J. A. (1992). Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, *31*(3), 590–616. doi:10.1147/sj.313.0590

Sutherland, J. (n.d.). *Scrum blog.* Retrieved from http://scrum.jeffsutherland.com/

The Open Group. (2009). *The Open Group architecture framework,* Version 9. Retrieved from http://www.opengroup.org/togaf

The Open Group. (2010). *Cloud buyers' decision tree.*

Tyree, J., & Ackerman, A. (2002). Architecture decisions: Demystifying architecture. *IEEE Software*, *22*(2), 19–27. doi:10.1109/MS.2005.27

Weerawarana, S., Curbera, F., Leymann, F., Storey, T., & Ferguson, D. F. (2005). *Web services platform architecture*. Prentice Hall.

Wesenberg, H., Landre, E., & Rønneberg, H. (2006). *Using domain-driven design to evaluate commercial off-the-shelf software*. OOPSLA Companion. Retrieved from http://dblp.uni-trier.de/db/conf/oopsla/oopsla2006c.html - Wesenberg

Zdun U., Hentrich C., & Dustdar, S. (2007). Modeling process-driven and service-oriented architectures using patterns and pattern primitives. *ACM Transactions on the Web, 1*(3).

Zimmermann, O. (2009). *An architectural decision modeling framework for service-oriented architecture design*. PhD Thesis, University of Stuttgart.

Zimmermann, O. (2011). Architectural decisions as reusable design assets. *IEEE Software*, *28*(1), 64–69. doi:10.1109/MS.2011.3

Zimmermann, O., Koehler, J., Leymann, F., Polley, R., & Schuster, N. (2009). Managing architectural decision models with dependency relations, integrity constraints, and production rules. *The Journal of Systems and Software and Services*, *82*(8).

Zimmermann, O., Schuster, N., & Eeles, P. (2008). *Modeling and sharing architectural decisions, Part 1: Concepts*. IBM developerWorks.

Zimmermann, O., Tomlinson, M., & Peuser, S. (2003). *Perspectives on web services: Applying SOAP, WSDL, and UDDI to real-world projects*. Springer Professional Computing.