Preface

In this publishing project we are editing a book titled Aligning Enterprise, System, and Software Architecture (AESSA). The main goal of this book is to outline some of the current thinking on the processes and practices for aligning enterprise, system, and software architectures. These three architecture areas have many commonalities and they often overlap in practice. There are many gray areas in which the expertise of more than one of these architectures are required in the planning and design of a system.

This book brings together representative views of recent research and practice in the area of aligning enterprise, system, and software architectures. Practicing software engineers, software architects, all researchers advancing our understanding and support for the aligning enterprise, system, and software architectures, and all students wishing to gain a deeper appreciation of underpinning theories, issues and practices within this domain will benefit from this book.

INTRODUCTION

It is imperative that the architecting process be intertwined and interleaved with requirements and architectures: much of the requirements communicate stakeholders’ and the enterprise technical, business, environmental and strategic concerns. These concerns need to be realized and accommodated by the architecture of the software system. Architecting software systems is process that crosses business, technical and strategic concerns. In today’s world of rapidly changing information technology, organizations, and market places, requirements tend also to change, and in ways that affect the stability of the architecture. The landscape for managing the co-evolution of requirements and architectures is becoming more complex; as a result, it is becoming a trend to describe various architectural views related to the enterprise, software and systems perspectives. Such as an approach is believed to cater for “modularity,” “separation of concerns,” and manage complexity through “decomposition”; it follows the trend towards heterogeneity in reasoning and assist in development and evolution of long-lived complex software systems. However, reconciling these views and managing their evolution is a problem. For example, it is challenging to prioritize these views, weight their importance and trace their concerns to relevant stakeholders and software artefacts. It is also challenging to manage the conflicts arising from inconsistencies in reconciling these views and co-evolving the views with the associated artefacts (and as the requirements change). Current architectural practices, however, do not provide a support for traceability from the requirements specification to the architectural description related to these views (e.g., which and (how) requirement(s) in the requirements specification an individual architectural element relate to and satisfy and vice versa). Maintaining traceability “links” between these views is necessary.
for managing the change, the co-evolution of both the requirements and the architecture, confining the change, understanding the change impact on both the structure and the other requirements, providing a support for automated reasoning about a change at a high level of abstraction. Further, such traceability “links” make it easier to preserve the enterprise strategy, the acquired knowledge of the team, the architectural knowledge through guided documentation. For example, this may then minimize the impact of personnel losses and may allow the enterprise to make changes in the software system without damaging the architectural integrity (and making the software system un-evolvable).

**WHAT ARE ENTERPRISE, SYSTEM, AND SOFTWARE ARCHITECTURES?**

Enterprise, System, and Software architectures play a critical role in developing software-intensive complex systems. However, the scope and focuses of each type of architecture design is different but overlapping. Enterprise architecture is a complete expression of the enterprise; it describes the alignment of business processes with IT, the composition of software systems and subsystems, and their relationships with the external environment, and the guiding principles for the design and evolution of an enterprise. Enterprise architecture acts as a collaboration platform between aspects of business planning such as goals, visions, strategies and governance principles; aspects of business operations such as business terms; organization structures, processes and data; aspects of automation such as information systems and databases; and the enabling technological infrastructure of the business such as computers, operating systems and networks.

System architecture is defined as a formal description of a system, or a detailed plan of the system at component level to guide its implementation. It is also the structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time. System architecture comprises the design of both software and hardware systems, design issues arising from such a system architecture design often concern both systems.

As the field of Software Architecture enters its third decade of formal study it finds itself moving from its traditional focus on software structures to the more general notion of software architecture as the set of design decisions made to ensure the software requirements will be met. Consistent with this view is the trend toward focusing software architecture documentation on supporting stakeholders in understanding how the software solution satisfies their concerns. These stakeholder concerns can be viewed roughly along two axes: concerns related to the software itself and concerns about how the software relates to other systems with which it interacts. Over the last decade it has been widely recognized and accepted that architectural decisions must take into consideration not only the functions the software is expected to perform but also the quality attributes associated with the functions. In recent years the community has come to consensus that quality concerns related to security, performance, reliability, maintainability, etc. cannot be ensured unless they are considered and their realization mechanisms properly documented from the outset, and they cannot be coded in during development. In the next decade we will see increasing focus on the second of our two axes: how systems software relates to the software of systems with which it interacts. Thus there will be increasing collaboration between those concerned with software architecture and those concerned with system and enterprise architectures.

As business processes are supported and realized by software systems, the design of software and systems in turn can pose constraints and generate requirements for the former. The interrelationships between the three areas of architecture designs are intricate and interdependent. The current practice of
treat the three architectures separately cannot work as Business-IT alignment remains a major issue. This issue is partially a result of not having alignments between the three architectural practices. For instance, the contradictions in the scoping, feasibility, budget and schedule between enterprise architecture and software architecture are often experienced in practice when the two architecture planning are misaligned.

**BOOK OVERVIEW**

We have divided this book into four parts, with a general editorial chapter providing a more detailed review of the domain of aligning enterprise, system, and software architectures.

**Section 1: Architecture Alignment - Theories**

The goal of Section 1 is to clarify the relationships among different kinds of architectures, explore and identify areas of commonalities and differences, and discuss the role of various approaches in helping to capture these architectures.

The four chapters in this section deal with: relationships between enterprise, applications, and infrastructure architects; using semantic Wiki for tracing process and requirements knowledge in enterprise: evolutionary architecting of embedded systems; relating software licenses, open source components, and open architectures.

In recent years, a rapid emergence of the role of the technology architect has been seen. However the lack of widely accepted definitions for the responsibilities of different types of architect can cause quite a lot of confusion between them, often reducing their effectiveness. Chapter 1, by Rozanski and Woods, helps to remedy this situation by identifying the defining characteristics that all architects share and defining a simple taxonomy of architects that can be used to classify the many varied jobs that technology architects perform (enterprise architects, application architects and infrastructure architects). The authors examine the responsibilities of each group and the interactions that should be expected between them. This analysis helps to define each type of architect more clearly so helps to define the relationship between enterprise, system and software architects.

Small and medium enterprises (SME) lack knowledge and resources for managing traceability links between requirements, processes, system, and software architecture. Wiki systems are widely used for documentation, but currently too limited in functionality to be applicable for these purposes. Based on SME case studies Seedorf, Nordheimer, and Thum describe in chapter 2 a solution for the lightweight modeling, documentation, and traceability of business processes, requirements, and software components. The wiki-based solution is realized as an extension to the Semantic Media Wiki software. Practical cases demonstrate how SMEs can benefit from the suggested approach.

Many industries rely heavily on embedded software and systems to maximize business value in their systems. These products are very complex and the architecture is important to control this complexity and make development efficient. It is rare that these companies start from scratch when developing new products, and instead they evolve their products over time, which means that the architecting needs to be evolutionary, and that this evolutionary architecting becomes a central business process. In chapter 3, Axelsson describes what such an evolutionary architecting process can look like based on our observations from industry. He also discusses how the process can be continuously improved using a maturity
model and through process optimizations using lean principles. A special concern is to find a balance between short-term and long-term considerations, and in this, strategic refactoring plays an essential role which is further elaborated, together with a description of analysis methods that can be used to ensure business value over time.

The central problem Alspaugh, Asuncion, and Scacchi examine and explain in chapter 4 is to identify principles of software architecture and software licenses that facilitate or inhibit success of an enterprise open architecture (OA) strategy when an enterprise develops large systems by composing open source software together with other non-open software components through open APIs. This is the knowledge we seek to develop and deliver. Without such knowledge, it is unlikely that an OA that is clean, robust, transparent, and extensible can be readily produced that aligns strategy, system architecture, and software together. Such an outcome results in an enterprise not knowing what its license obligations are, thus requiring expensive legal consultation for uncertain technical guidance, or the enterprise becoming subject to legal liabilities for license infringement. On a broader scale, this chapter seeks to explore and answer the following kinds of research questions:

- What license applies to an OA system composed with components with different licenses?
- How do alternative OSS licenses facilitate or inhibit the development of OA systems?
- How should software license constraints be specified so it is possible to automatically determine the overall set of rights and obligations associated with a configured software system architecture?

This chapter may help establish a foundation for how to analyze and evaluate dependencies that might arise when seeking to develop software systems that embody an OA when different types of software components or software licenses are being considered for integration into an overall system configuration. Simple examples involving software systems commonly found in different enterprise subject to different licenses are included.

### Section 2: Crossing Enterprise, System, and Software Architecture Boundaries

It is imperative that the architecting process is intertwined and interleaved between requirements and architectures: much of the requirements communicate stakeholders’ and the enterprise technical, business, environmental and strategic concerns. These concerns need to be realized and accommodated by the architecture of the software system.

The four chapters in this section provide insights into following topics: using RESTful architecture to tackle software diversity in mobile web systems; enterprise applications; using genetic algorithms to search for key stakeholders; seamless coherence in security issues across enterprise, system, and software architectures.

Mobile devices create new opportunities for companies. Innovative applications can cause challenges for software and system architecture and have an impact on business. In Chapter 5, Wehrmaker and Schneider describe an anti pattern of short-sighted propagation. A new and risky application is explored using a single technology to start with. When the system spreads throughout the enterprise, diversity of mobile platforms increases. The initially simple architecture gets messed up by “simple” extensions to accommodate other platforms. This leads to decreasing performance and maintainability. Their proposed solution approach is based on the REST paradigm. As a road to a better system, software, and
enterprise architecture, they propose a migration process and explain its key steps to an extensible and forward-looking application.

In Chapter 6, Choppy, Hatebur, Heisel, and Reggio provide a method, which allows one to systematically develop enterprise application architectures from problem descriptions. For enterprise applications, they have developed a specialized enterprise application problem frame (pattern). The authors describe the requirements through problem diagrams that are instances of the enterprise application frame and of other problem frames. Based on the problem diagrams, they systematically develop enterprise system architecture. For this, they consider technical requirements, for example that some functionality should be implemented on another computer (e.g., remote databases). They introduce coordinator components, considering the formal descriptions of the business model, and facade components. Their method not only provides detailed methodological guidance for developers, but also validation conditions to find semantic errors early in the development process.

Large software projects have many stakeholders. In order for the resulting software system to be aligned with the enterprise and stakeholder needs, key stakeholders must be adequately consulted and involved in the project. Chapter 7 by Soo Ling Lim, Angelo Susi, and Mark Harmon proposes the use of genetic algorithms to identify key stakeholders and their actual influence in requirements elicitation, given the stakeholders’ requirements and the actual set of requirements implemented in the project. The proposed method is applied to a large real-world software project. Results show that search is able to identify key stakeholders accurately. Results also indicate that many different good solutions exist. This implies that a stakeholder has the potential to play a key role in requirements elicitation, depending on which other stakeholders are already involved. This chapter demonstrates the true complexity of requirements elicitation – all stakeholders should be consulted, but not all of them should be treated as key stakeholders, even if they appear to be significant based on their role in the domain.

Section 3: Architecture Processes, Tools, and Techniques

A good architecture is realized through the application of solid processes and carried out using appropriate tools and techniques. Development of these supporting technologies has been an increasingly active area of research over the past two decades and current thinking suggests that the use of models as a part of documentation is fundamental to communication of the architecture to others for such purposes as education of various stakeholders, architecture assessment to ensure the systems built to it will meet the project’s quality as well as functional goals, and as the embodiment of architectural decisions. In Section 3 of the book we present 4 chapters describing new techniques presented with supporting examples and case studies describing application of the processes, tools, and techniques in practice.

Betz, Burger, Eckart, Oberweis, Reussner, and Trunko discuss the importance of understanding dependencies among business process and business software lifecycles in Chapter 8. They present a framework based on business simulations and predictions that improves overall management of the organization.

In Chapter 9, Muller describes a technique for capturing various abstractions of architecture as models for which the relationships among the models are mapped in a way that allows the models to be used together to gain insight into the project plan as a part of the larger enterprise.

In Chapter 10, Zimmermann and Miksovic present a lightweight technique for capturing architectural knowledge in a way that will preserve decisions that recur in a domain along with their rationale. Their technique extends enterprise architecture frameworks with guidance models to support the exchange of knowledge about known problems and solutions between enterprise and solution architects.
Taken together these chapters give insights as to the importance of understanding and modelling interdependencies among various artifacts and processes and present solutions to help organizations deal with human and technology based complexities inherent to any project involving software. In Chapter 11, Kamath discusses the importance for understanding the relationships among business, information technology, and application architectures as these three facets of an organization, though usually created and managed as if they are isolated, are highly related to each other and indeed generally suffer from crucial interdependencies. Kamath presents an approach using Category Theory to link these architectures in such a way as to highlight interdependencies in a way that increases an organizations ability to evolve based on sound decision-making that promotes agility in all facets of operation.

Section 4: Industrial Case Studies and Practices

Section 4 contains three chapters looking at a variety of industrial cases and practices. The chapters in this section present practical approaches and cases. Chapter 12 analyzes architecture practice in an agile environment; chapter 13 focuses on the software architecture of the problem; chapter 14 describes using obstacles and goals to guide adoption of cloud computing.

Chapter 12 is a review of the results of a survey carried out in twelve IT companies in Finland. In this survey, Eloranta and Koskimies mapped out existing practices regarding software architecting work in agile enterprises, and analyze their benefits and problems. They found correlations with the agility degree of the enterprise and those practices. They studied the Scrum process as the reference work management framework to explore the relationship of agility (in the form of Scrum) and the process of architecting. They found that some architecture practices are problematic and are in conflict with using Scrum. The study indicates that possible management risks arising from conflicts between the new Scrum patterns and the adopted way of architecture work need to be managed.

Enterprise, Systems, and Software Architecture share many commonalities and many of their concerns overlap, but they also have marked differences in focus and approach. Each solves problems for different stakeholders, uses different technologies, and employs different practices. The specialization on their respective solutions has made it difficult to transfer methods and knowledge across a broad range of topics. One way to align these topics is to shift the focus from solution to problem domains. In Chapter 13, Alfred suggests using context and challenge constructs for modeling a problem domain. A context represents a set of perceptions, desired outcomes and priorities of like-minded decision makers. Challenges are significant risks or obstacles within a context which must be overcome to deliver value. From this perspective, much of what separates Enterprise, Systems, and Software architecture are their respective focus on solution concerns.

- Enterprise Architects focus on issues such as security, fault tolerance, availability, and system integration.
- Systems Architects focus on issues such as electro-mechanical control, real-time scheduling, product line opportunities, safety and fault handling.
- Software Architects focus on GUI’s, business logic, workflow, databases, and non functional requirements in software.
The different stakeholders that enterprise architects, systems architects and software architects interact with each have their environments, their expectations, and their constraints. The modeling of a problem domain provides a means of aligning these stakeholders’ concerns.

Chapter 14 motivates the need for a systematic approach for cloud adoption, aligning the enterprise goals and its architecture with cloud systems architectures. This chapter suggests use goals and obstacles for identifying, managing, and mitigating risks in the cloud adoption process. Zardari, Faniyi, and Bahsoon propose an architecture-centric risk management framework (CloudMit) for systematically identifying risks and possible mitigating strategies, using architecture-level security risks as an example.

Ivan Mistrík  
Independent Consultant, Germany

Antony Tang  
Swinburne University of Technology, Australia

Rami Bahsoon  
University of Birmingham, UK

Judith A. Stafford  
Tufts University, USA