
Preface

This book is about building and sustaining a competitive advantage for your company, thus contributing to its value and, not least, to your own personal success. We can think of business processes as the central nervous system of a company, but they are often sorely neglected in company strategy. Of course, a company must offer excellent products and/or services to have any chance of long-term success, but there is also enormous potential for differentiation and innovation in a company's business processes, and concentrating efforts in this area can make the difference that sets your company apart from the competition. Process innovations also have the unique advantage that they are not as visible to your competitors as products and services are, and they are much more difficult to copy. Just think how innovative Apple's iPhone was when it was first released, but how quickly competitors brought out their own imitations. Focusing on process-oriented innovation helps you protect and sustain growth and success.

How can we tap into this precious resource? How can we optimize business processes and exploit their full potential? These are the questions I want to answer in this book, and with the BPMN (Business Process Model and Notation) standard we now have new possibilities for planning, implementing, and monitoring processes. I particularly want to address the transition from business process planning to implementation. You have probably experienced cases yourself where business BPMN models have been altered beyond all recognition in their implementation as executable BPMN models, but this *can* be avoided. The BPMN standard enables you to create a *single process model* for both business and IT departments. I want you to seize this opportunity and use BPMN to extract the maximum benefit from the processes in your company. The approach I present here shows you how to use BPMN effectively, without having to sacrifice a common business process model, and ultimately makes the BPMN objective of executable business processes a reality.

We must also look at the transition from business process planning to implementation from another angle: The decisions we make at this early stage have far-reaching implications on resources (and of course costs), so we have to think very carefully about how to ensure that strategic processes can be implemented successfully. I favor a process-driven approach combining a process-driven methodology, a process-driven architecture, and a process-driven application. Ultimately, we have to rise to the challenge of developing applications in

heterogeneous IT landscapes, as most end-to-end processes involve a number of different systems and partner connections. This is what originally inspired my interest in this topic.

My own journey into the world of application development for complex distributed system landscapes began more than 10 years ago. I was working at Siemens Business Services (SBS) in a department that developed Java-based applications. At that time, our team was developing new software that would implement new business processes for managing software licenses. The software would keep the licenses required by a company in a central license pool. When new computers were purchased, the IT department would install the software relevant to the role of the employee that the computer was intended for, taking it from the license pool. Conversely, whenever a computer was retired or an employee left the company, the license would return to the pool, where it would then be available for further installations. The processes that we were developing also had to interact with a number of existing systems. For example, new licenses were ordered using an SAP system, employee information was stored in an LDAP (Lightweight Directory Access Protocol) system, and workflow notifications were sent using an e-mail server. To top it all off, the application had its own JDBC-based persistence for saving information about who the computer belonged to and what software was installed on it. To sum up, it was a classic, distributed, Web-based application for implementing new processes spanning numerous different systems.

Our team developed a solution based on the prevalent JavaEE methodology, using the whole palette at our disposal: session beans, entity beans, message-driven beans, servlets, Java Server Pages, Apache Struts as a Web framework, and Web services. Our architect also insisted on a modern software architecture. I can remember him coming over to my desk one day and proudly telling me how many Gang-of-Four patterns (Gamma et al. 1995) he had managed to put to good use in the software (of course, more patterns does not necessarily mean higher quality software!). The software ran on a BEA Weblogic server and eventually found its way onto an SAP Web Application Server (Stiehl 2004). Customer feedback to our new solution was certainly positive, but before long it became clear that potential customers were storing their data in systems other than those we had envisaged when creating the software. How could we tackle this problem and keep our software attractive to as many customers as possible? Should we create a new variant for each customer, perhaps? Considering the sheer number of possible combinations of back-end systems, this proposal was not very encouraging. It was obvious that our solution was very vulnerable to changes in system landscapes. To put it another way, we had relied too heavily on a fixed and stable IT landscape. How did this happen? Why did we not abstract the solution from the back-end systems from the very start? But how do you actually implement this abstraction? And to what extent? How do you distribute the tasks across the abstraction levels? And above all: shouldn't a service-oriented architecture solve exactly this problem? All these issues culminated in one central question: How can I create robust applications for distributed IT landscapes that are becoming ever more fragmented as the number of on-demand and cloud solutions increases?

I wanted to get to grips with this problem: I analyzed projects, read technical articles, and attended conference lectures discussing these challenges. The more I learnt, the clearer it became that the architectures that were being presented all had similar shortcomings. Talking to project managers, I often hear other reasons for these inadequacies: They face intense pressure in their projects, and tight deadlines and budgets mean that they cannot invest the resources required to be able to produce a sufficiently robust software architecture. This, in spite of the fact that they are well aware of the consequences: short-term savings in development swallowed up in the long term by significantly higher costs for maintenance, support, and further development. Long-term goals are being sacrificed for the sake of short-term targets. Therefore, it is no surprise that software is being widely criticized for not being agile enough to adapt to the business strategies set out by management; we are overwhelmed by the task of untangling the complex web of increased dependencies between end applications and the system landscape.

In this book, I present an approach for implementing an architecture for end applications that strives to find a sensible balance between development and maintenance costs, sustainability, scalability, and error tolerance; an approach that meets flexibility requirements, while at the same time not being too complex itself; an approach that keeps the end application as abstract as possible from the system landscape in which it operates. In my search for this solution, I happened to come across the process modeling standard BPMN. This had started life as a graphical notation for modeling business processes, but had since developed into an outstanding instrument for implementing application architectures just like those I was faced with. My understanding of application architecture is a solution for end applications based on heterogeneous IT landscapes, which differentiates business and integration-centric processes. The most recent version of the BPMN standard (version 2.0) also had semantic enhancements that made it possible to execute process models created in this way, so I decided to start using BPMN to create and run complete application architectures. I do not use BPMN to model just the business processes of the end application, as the “B” in BPMN might suggest; I also use it to model and execute the integration processes between the systems. Although seemingly contradictory, these two uses of BPMN do fit together, as I will explain in this book.

With the application architecture I propose, I hope to present architects and software developers with a detailed blueprint, whose principles they can then use to plan and implement distributed applications in their companies. If application development adopts this approach, the solutions developed will help to gradually release companies from the stranglehold of dependencies and system connections mentioned earlier. In this regard, I see process-driven applications as contributing towards enterprise architecture management (EAM); however, it must also be said that this book is not about EAM; it covers areas outside the EAM spectrum.

The book is aimed at software architects, IT managers, managers with a technical interest, software developers, project managers, and also students of information

and business technology. I assume that readers have a basic knowledge of BPMN. I introduce the notation briefly, but I do not go into detail about each BPMN element. Likewise, I also assume my readers have a basic understanding of service-oriented architectures, as my approach also relies on reusable services in back-end systems.

Walldorf, Germany

Volker Stiehl



<http://www.springer.com/978-3-319-07217-3>

Process-Driven Applications with BPMN

Stiehl, V.

2014, XVII, 334 p. 221 illus., Hardcover

ISBN: 978-3-319-07217-3