The Enterprise Engineering Series

More information about this series at http://www.springer.com/series/8371

Boris Shishkov

Designing Enterprise Information Systems

Merging Enterprise Modeling and Software Specification

Springer

Boris Shishkov
Faculty of Information Sciences
University of Library Studies
and Information Technologies
Sofia, Bulgaria

Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Sofia, Bulgaria

Interdisciplinary Institute for Collaboration
and Research on Enterprise Systems and Technology
Sofia, Bulgaria

ISSN 1867-8920 ISSN 1867-8939 (electronic)
The Enterprise Engineering Series
ISBN 978-3-030-22440-0 ISBN 978-3-030-22441-7 (eBook)
https://doi.org/10.1007/978-3-030-22441-7

Prologue

We arrive at the truth not by the reason only but also by the heart
—Blaise Pascal

I have written this book for:

- Analysts who are interested in the construction and the operation of enterprises.
- Enterprise engineers who are challenged by the needs for adequate models to be used either as a basis for enterprise engineerings/reengineerings and/or in support of software specifications.
- IT architects who are inspired to improve the way in which they are specifying enterprise information systems.
- Software engineers who are aware of the necessity for better establishing the computation-independent models that concern the software system-to-be.

Even though delivering adequate enterprise models used to be a challenge, the emerging enterprise engineering discipline and its underlying social theories seem to be giving a good basis for modeling complex real-life (organizational) processes. Next to that, software engineering and corresponding computing paradigms have advanced and represent a good basis for developing software, starting from a computation-independent model of the software system-to-be. Nevertheless, bringing together enterprise modeling and software specification is still a challenge, although the enterprise-software gap has been broadly discussed for more than 15 years already. For this reason, it is not surprising that currently many software projects are reaching failure and/or going overbudget and/or bringing insufficient satisfaction to users and so on. Hence, the modeling featuring enterprises and their processes (on the one hand) and the software specifications (on the other hand) need to be better aligned and considered as one integrated task. Otherwise, computation-independent software models that lack adequate enterprise modeling background would keep on leading to the development of software that would only partially fit its real-life (enterprise) environment. I provide further elaboration and justification concerning those claims in the following two paragraphs.

THE IMPORTANCE OF ENTERPRISE MODELING

For centuries already, business processes of different kinds have been part of our societies. For decades already, global enterprises have been offering products and services around the globe. Currently, organizations are experiencing increasing needs for adequate ENTERPRISE MODELING because of the following: (1) Often (distributed) organizational processes are becoming too complex to be grasped intuitively. (2) Change impact analysis is often needed, but it requires structured enterprise data as input. (3) Enterprise innovations are considered of key importance for gaining competitive advantages, but this would often assume enterprise reengineering activities that in turn "ask" for legacy models and corresponding information. (4) Introducing new technology and/or accommodating automations can only be really successful if the technology is well aligned with the original business processes; this in turn can only be possible if we are able to adequately describe and model those processes, such that we know the "design restrictions" with regard to the introduced technology. Those are only four claims justifying the importance of enterprise modeling; I can provide even more justification in that direction. Nevertheless, we observe insufficient maturity as it concerns enterprise modeling: (a) Many analysts conduct intuitive enterprise modeling that is not scientifically justified—this often leads to modeling of low quality. (b) They often fail to be exhaustive in their modeling—some of them would only focus on modeling behavior and others would only focus on modeling data, and so on. (c) Some analysts would mix up essential business things (e.g., John paid for his service subscription) with information exchange that is not featuring essential business things (e.g., John entered his PIN incorrectly while using an ATM). (d) Other analysts would be unaware of the importance of communicative acts in real-life communication, through which commitments are generated, that are in turn crucially important with respect to the processes within an organization. (e) Many analysts would overlook regulations and public values as key restrictors with regard to the functioning of an organization. Hence, a sound and exhaustive conceptual reference is needed and the emerging ENTERPRISE ENGINEERING discipline is expected to be the solution.

USING ENTERPRISE MODELS AS A BASIS IN GENERATING SOFTWARE SPECIFICATIONS

Businesses changed when computers first appeared on the scene and enterprises had to accommodate (partial) automation of their business processes. Businesses changed again when web services and cloud infrastructures appeared, and enterprises faced the challenge of making (some) internal business processes external. Nothing was the same anymore—two "worlds" emerged: the domain experts, "playing on the enterprise field," and the technical experts, "playing on the software field." Nevertheless, more than 15 years of discussions (as mentioned above) featuring the mismatch between enterprise modeling and software design did not help—the two "worlds" are still separated. The hopes for changes are justified by the rapid enterprise-technological developments (see the beginning of the current paragraph) and it is expected that it will be those developments that would make the

abovementioned mismatch socially unacceptable anymore. Said otherwise, the abovementioned developments have been justifying more and more the necessity for bringing together enterprise modeling and software specification since a domain (enterprise) expert alone is insufficiently capable of grasping the technical complexity of the enterprise's IT system and its reach outside through software services, while a software engineer would only have superficial enterprise-specific domain knowledge. What complicates things further is that software engineering and the emerging enterprise engineering discipline have developed separately. Most enterprise modelers would not "step in the shoes" of software designers, while most software engineers would not "broaden their horizon" beyond the software artifact being developed—even the computation-independent modeling of MDA (model-driven architecture) is just about the software system-to-be. Anything outside it would often be considered by software engineers as an "abstract environment," and the only thing to do about it is specifying "interfaces." Many software engineers would be insufficiently focused on questions, such as: What is the construction of the broader enterprise environment? How can the software system under development be adaptable with regard to possible environmental changes? How can the software system-to-be achieve user-perceived effectiveness? How can the software system-to-be be adequately aligned with the enterprise norms and broader societal regulations? How can human responsibility and authority be aligned to what the software system-to-be actually does? How can public values (such as privacy, accountability, and transparency) be properly "translated" into (software) requirements? The lack of adequate "answers" to those questions justifies the claim that the alignment between enterprise modeling and software specification is still uncertain, and this in turn often leads to enterprise information systems of low quality—software applications and information systems are not as effective as they should be; this leads to software failures as mentioned above. It is therefore not surprising that the latest technological innovations (e.g., innovations concerning Internet-of-things and machine learning) are only partially "brought" to enterprises via software applications. I argue that only the ENTERPRISE-MODELING-DRIVEN SOFTWARE GENERATION could be the solution as it concerns the mismatch between enterprise engineering and software design.

Bridging that gap is thus considered important as it concerns enterprise information systems, inspiring me to propose a modeling approach. For this reason, on the one hand, the proposed approach steps on a conceptual invariance (embracing concepts whose essence goes beyond the barriers between social and technical disciplines), while, on the other hand, the approach builds upon that "common ground" to accommodate a modeling duality featuring (1) technology-independent enterprise modeling that is rooted in social theories and (2) software specifications that are rooted in computing paradigms. On top of that, the approach's guidelines and related notations further grease such an enterprise-software modeling by facilitating modeling generations and transformations: starting from unstructured business information, coming through enterprise models, and reaching as far as the specification of software. The alignment between enterprise modeling and software specification is realized in a component-based way, featuring a potential re-use of

modeling constructs, such that the modeling effectiveness and efficiency are stimulated. Finally, I provide a case study and illustrative examples for the sake of "grounding" my studies and demonstrating some strengths and limitations of the proposed modeling approach.

Nevertheless, this book is not about the approach itself. This book is about bringing together enterprise modeling and software specification (maybe also by using another approach and/or other notations) such that an enterprise-modeling-driven software generation is achieved. Even though the book is supposed to be telling you how to sort this out, it does not give you an "A to Z" recipe as in cooking. Instead, it raises awareness and provides directions. I have stressed upon the mismatch between enterprise modeling and software design, and I have proposed solution directions accordingly, featuring on the one hand a conceptual invariance and on the other hand a modeling duality (see above). I believe that reading this book will inspire thoughts and ideas that are useful as it concerns YOUR way of analyzing enterprises and/or YOUR way of specifying software. Hence, I have not only presented social theories (Chap. 4) and computing paradigms (Chap. 5), but I have also introduced a common conceptual background for them, touching upon systemics (Chap. 2) on the one hand and context-awareness (Chap. 3) on the other hand. Further, by introducing (in Chap. 6) an approach (see above), namely the SDBC approach, and by considering accordingly a case study and illustrative examples, I have brought forward some justification on the adequacy and feasibility of merging enterprise engineering and software engineering. It is up to you to reflect those ideas, guidelines, and examples in your work, such that you usefully enrich the approaches you follow as it concerns enterprise modeling and/or software specification.

As it concerns readability, I have emphasized issues through various fonts and styles, for example: <Courier New 9 Points, bold>emphasis</Courier New 9 Points, bold>, <bold>emphasis</bold>, <underlined>emphasis</underlined>, <uppercase>emphasis</uppercase>, and so on. This allows me to emphasize in different "levels" distinguishing, for example, between different parts, then between key issues concerning each of the parts, then between different concepts considered accordingly, and so on.

This book is based on my previous book *Enterprise Information Systems - A Modeling Approach* which was published in 2017 but has never been really distributed and is no longer available. Even though the current book is essentially based on the 2017 book, it also contains new and reworked material, and is the only one that now counts as a summary of my work over the last years.

Anyway, more effort is still needed as concerns the mismatch between enterprise modeling and software design—this has been a challenge for many years already (as mentioned above), and bridging that gap is not just a matter of innovative ideas but also of changing attitudes. This challenge is of an interdisciplinary essence and it needs bringing together enterprise engineers and software developers, inspiring them to join interdisciplinary projects and discussions. I believe that the current book represents a small contribution in that direction. Hopefully, such efforts would be embraced by relevant communities and adequate solutions would be materialized. If you want to join activities in that direction, visit the website www.is-bmsd.org

featuring BMSD—the international symposium on business modeling and software design.

In carrying out those studies, I have been inspired by joint work and collaborations with my colleagues from Delft University of Technology, the University of Twente, the University of Reading, and IICREST, to whom I am happy and honored to extend my gratitude and compliments. Further, I am privileged as well to be leading for 9 years already (as General Chair + Program Chair) the prestigious BMSD symposium (see above), whose community's feedback is always so stimulating. Finally, I dedicate this book to the memory of my father, Blagovest; it is really special to me that it would have been my dad's 82nd birthday today. He used to be a wonderful father and a brilliant scientist, always inspiring!

I hope you will find the current book interesting and will consider it a helpful reference with regard to enterprise information systems.

Sofia, Bulgaria Boris Shishkov
6 April 2019

Contents