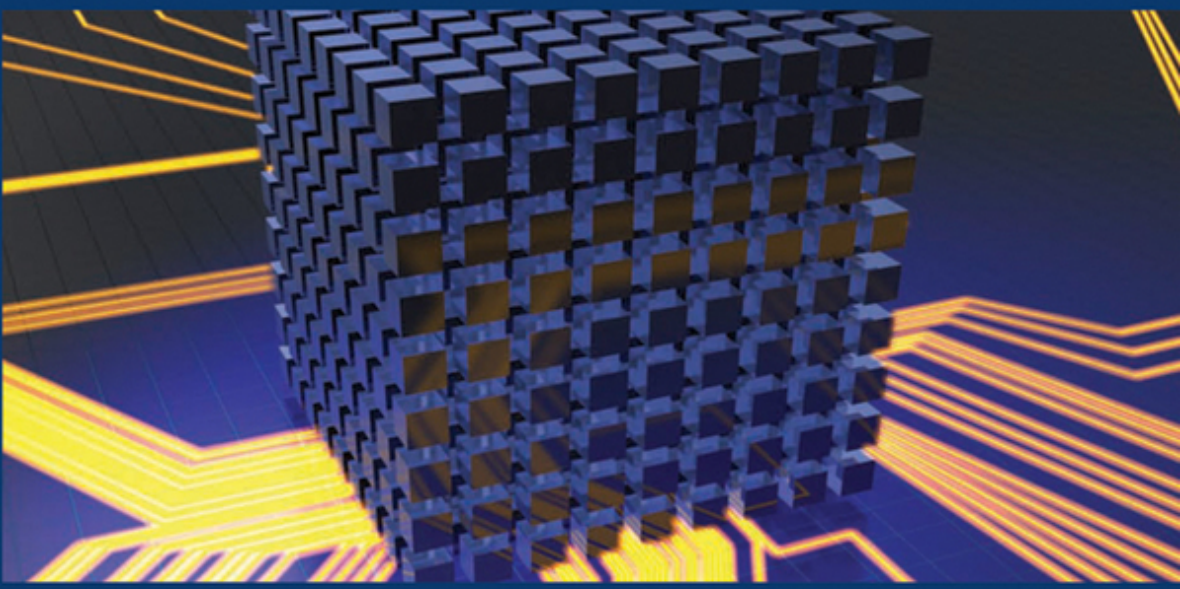

Managing Complexity of Information Systems

The value of simplicity

Pirmin Lemberger
Médéric Morel

ISTE
WILEY

Managing Complexity of Information Systems

Managing Complexity of Information Systems

The value of simplicity

Pirmin Lemberger
Médéric Morel

ISTE

WILEY

First published 2012 in Great Britain and the United States by ISTE Ltd and John Wiley & Sons, Inc.

ISTE Ltd
27-37 St George's Road
London SW19 4EU
UK

www.iste.co.uk

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA

www.wiley.com

Library of Congress Cataloging-in-Publication Data

Lemberger, Pirmin.
Managing complexity of information systems : the value of simplicity (TBC) / Pirmin Lemberger, Médéric Morel.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-84821-341-8
1. Management information systems. 2. Technological complexity. 3. Information technology. I. Morel, Médéric. II. Title.
T58.6.L447 2011
658.4'038011--dc23

2011042577

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
ISBN: 978-1-84821-341-8

Printed and bound in Great Britain by CPI Group (UK) Ltd., Croydon, Surrey CR0 4YY

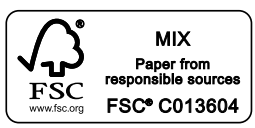

Table of Contents

Foreword

"Science does not think!"

In this famous and controversial statement[1], the philosopher Martin Heidegger certainly did not mean that scientists were stupid or that science was irrational. He was rather expressing the fact that science did not take time to think about itself, meaning its own goals and practices. This can inevitably lead to excesses or undesired results.

The book you are holding could be entitled "IT doesn't think", or "IT does not think enough!" Starting from scratch, it rethinks the goals of a "good" information system, asking what a "good" information system is, beyond choosing the "best" technology and beyond meeting deadlines or budget constraints.

The answer proposed here, "simplicity of the IS", relies on a thorough analysis of the countless sources of complexity that tend to make the IS into a chaotic jumble, which is hard to maintain and even more difficult to improve. This situation penalizes companies striving to remain viable in a fast-moving and competitive environment.

---

1 *What is called thinking?*, Martin Heidegger, Harper Perennial, 1976.

The value of this book is not in this or that specific recommendation but rather in the global vision that justifies the recommendations that are being made.

One of the major insights of this book is its repeated emphasis on the human factor, whether from the point of view of end-users or that of IT Departments.

IT people are often considered introverts who are uninterested in anything other than the latest techno hype. Because they also generally have a hard time meeting deadlines and budgets restrictions, IT management inundates them with technological, architectural, and organizational dictates. The necessity to master this vast array of tools, languages, and methods can leave them feeling disempowered.

The authors of this book argue that one should trust those on the frontlines and that some freedom should be given back to them, because the authors believe that this is the only way to build an IS with a minimum of common sense, which is to say, by never losing sight of the idea that simplicity, the essence of an IS, is not a goal but an ongoing journey.

It is not possible, unfortunately, to drop everything and start over from scratch: our hope is thus to determine where we want to go, to map out our journey, to regularly check that we are not straying from the chosen path, while giving ourselves the leeway to gaze at the panoramas before us and perhaps choose yet another path.

This book should be considered a sort of “survival guide”, simple enough to be usable and thorough enough to be relevant.

Pascal Grojean
Managing Director, Emoxa
Co-author of the books *SOA Architect’s Guide and Performance of IT Architectures*.

Preface

Many organizations are now reaching the same conclusion: mastering technical and organizational complexity is today the primary difficulty to overcome for their IT departments, much more so than reaching some critical magnitude in IT investments. At best, poorly managed complexity will prevent any reliable predictions for possible future evolutions of the system. At worst, the sustainability of the system as a whole could be put at stake. It would obviously be illusory, if not naive, to attempt to remove complexity altogether from the IS. The aim is rather to master the growth of complexity and to make sure that it stays in reasonable proportion to the actual usefulness of the IS to its various stakeholders. More precisely, the goal is to avoid an uncontrolled proliferation of "useless" complexity to ensure the scalability of the system and to maintain the satisfaction of its users.

This book develops the point of view according to which mastering complexity implies two essential steps: first, we must develop a *clear understanding* of the real nature of complexity within the IS; second, we must *identify the primary causes*, which contribute to its uncontrolled growth and organize these into a logical framework, in order to define efficient countermeasures. We also consider that any serious explanation for IT complexity should deal with both technical and psychological causes of complexity.

Two themes make up the main thread of our book: complexity and value. Both themes are quite common when considered separately. Their interplay, however, has remained a largely unexplored topic.

Our approach to IS complexity combines theoretical analysis with practical field experience. This kind of comprehensive analysis differs, we believe, from both academic works, which focus mostly on theoretical computing and also from so-called pragmatic approaches that simply list catalogs of recipes without bothering to provide a sound conceptual basis for them.

Target audience

This book will be of interest to CIOs as well as to enterprise architects and project managers. Parts of it are written on a more conceptual level than most IT books. This will perhaps require some readers to postpone somewhat their legitimate desire to rush out and apply simplicity rules to real life. We believe, however, that this postponement is worthwhile and the reader will be rewarded with a deeper, and thus more efficient, understanding of the true origins of unmanageable complexity in the IS.

Acknowledgments

This book would not have been possible without the support of SQLI CEO Julien Mériaudeau.

The authors would like especially to express their gratitude to several colleagues, who kindly agreed to share their expert knowledge and experience. Special thanks go to: Mr. Manuel Alves, director of Alcyonix Paris, an Enterprise Architect whose extensive experience in software engineering and project management, and sharp critical

mind proved invaluable when it came to confronting theoretical analysis with practical IT issues.

Mr. Simon-Pierre Nolin, senior consultant in IT infrastructure at Alcyonix, provided his deep insights and extensive field experience regarding how simplicity principles could be implemented in IT operations.

The authors thank Dr. Julian Talbot from the Laboratory of Theoretical Physics of Condensed Matter at Pierre et Marie Curie University in Paris for his critical proofreading of an early version of the manuscript.

The authors thank especially Mr. Jean-Luc Raffaëlli, Strategic Project Director at Groupe La Poste and Mr. Pierre Bonnet co-founder of Orchestra Networks for their insights and feedbacks.

Last but not least, Mr. J. Patterson Waltz, consultant in processes improvement at Alcyonix, reviewed the whole manuscript with an impressive dedication and thoroughness. Obviously, any remaining inaccuracies or typos remain the sole responsibility of the authors.

Chapter 1

Why Simplicity?

Simplicity is the ultimate sophistication
Leonardo da Vinci

1.1. Solving conflicting requirements

Information systems (ISs) are now ubiquitous in nearly all large companies and organizations. They provide a permanently available online store to customers. They automate an ever-increasing proportion of business processes and tasks, thus contributing to the rationalization effort and cost reduction required by the globalization of competition. Senior executives use ISs to perform business activity monitoring that allows them to react quickly in fast-moving markets, where reducing the time to market is more important than ever. ISs have thus truly become an essential tool for sound decision-making as well as for selling or providing goods and services.

We might naively think that such a strategic position would logically call for putting maximal effort into designing robust and perennial systems. However, as most likely any

reader of this book will know by experience, such is hardly ever the case. Unlike road networks or buildings, most ISs are not really built or designed to last. Rather, they grow much more like living organisms, responding to a set of fluctuating and contradictory forces while trying to adapt in an open environment. A common situation is one in which the number of users grows, both inside (employees and IT personnel) and outside (customers) the company, while at the same time those same users all become more demanding. They expect more speed, more reliability, more flexibility, and a better user experience and all of these simultaneously.

The most acute conflict between these expectations is probably less between speed and reliability than between flexibility and reliability. Speed could certainly be achieved, at least in principle, by using mere brute force, which means by allotting sufficient technological and human resources to designing and operating the IS. Flexibility, on the other hand, could probably not be achieved even if we had an infinite amount of resources available. The fact that brute force will not do is a hint that what we are facing here is a deeper issue than achieving mere performance. More flexibility typically involves meeting unclear and fluctuating user requirements. Often it also means providing improved customization to all stakeholders. Agility and fast turnaround are thus the key requirements here. Building reliability, on the other hand, requires a lengthy design phase, deep understanding of the interdependence of subsystems, performing many tests, and gathering extensive feedback about the system's behavior. Building reliability means building human understanding, which is in essence a slow process.

At least two other factors often contribute to make the situation even worse. First, there is the successive technological hype for such things as "EAI", "SOA", "EJB",

"MDM", or any other acronym you might have heard floating around in recent years. This succession of technologies will progressively generate uncontrolled complexity in the IS. Second, under such difficult circumstances, some key employees with technical or business skills might simply want to quit and look for a better working environment. Now, sum up all the previously mentioned forces that shape an IS: the need for flexibility, the multifaceted techno-hype, and perhaps a high turnover, and this will quite soon result in an organizational and technological nightmare that is probably best described as chaos! As physicists tell us, chaos is a situation which is unpredictable. This is the exact opposite of why the IS was built in the first place. In such near-chaotic situations, nobody has a clear picture of what the system is really doing, what the information feeds contain, how the data are structured, and which hardware processes are running. Not surprisingly either, nobody wants to assume the responsibility for making any decisions or changes. Incidentally, it is not by chance that most system architecture endeavors start by mapping the existing system because nobody really knows what the system is made of! Does this sound familiar?

This apparently uncontrollable increase in entropy of computing systems is by no means new. The recent need for opening older systems to the web and the plethora of technologies that pretend to be optimal in this respect only exacerbated the existing tendency for computing systems to grow out of control. For nearly half a century, however, software languages, architecture principles, and development processes have been designed to solve this apparent contradiction of building computing systems that are both maintainable, meaning well-structured and understandable by human minds, and, at the same time, flexible enough to accommodate changing requirements. Let us briefly review some of these here.

On the software engineering side, object-oriented programming (OOP) was probably one of the most significant such attempts. In non-technical terms, what OOP in principle does is to allow constructing a larger system from smaller ones by progressive and controlled aggregation. Traditional procedural languages were notoriously bad at achieving such a goal and OOP was, no doubt, a major breakthrough.

Architectural principles were also proposed, with the aim of organizing and decoupling as much as possible the various processing layers. They all involve the idea of using components, which are reusable pieces of software that should be as autonomous and decoupled from the others as possible. The best known example here is probably the three-tier architecture where components in charge of the presentation logic are clearly separated from those in charge of implementing the business rules, which are in turn decoupled from those responsible for recording the data in permanent storage.

More recently, we saw the advent of the so-called service-oriented architecture (SOA), motivated by the need for business-process flexibility and reusing legacy components. SOA proposes a component architecture, not just in terms of the software architecture for one application, but for the whole IS.

Finally, iterative engineering processes were designed, such as *extreme programming* or *Lean Software Development*, to provide controlled methods for dealing with unclear and quickly changing user requirements.

Each of these topics will be treated in depth in later chapters. For now, let us note that this continuous struggle explains why, during the early years of ISs, management was mostly driven by technological innovation. This is the first topic of the following section where we take some time