Smart Innovation, Systems and Technologies

Volume 92

The Smart Innovation, Systems and Technologies book series encompasses the topics of knowledge, intelligence, innovation and sustainability. The aim of the series is to make available a platform for the publication of books on all aspects of single and multi-disciplinary research on these themes in order to make the latest results available in a readily-accessible form. Volumes on interdisciplinary research combining two or more of these areas is particularly sought.

The series covers systems and paradigms that employ knowledge and intelligence in a broad sense. Its scope is systems having embedded knowledge and intelligence, which may be applied to the solution of world problems in industry, the environment and the community. It also focusses on the knowledge-transfer methodologies and innovation strategies employed to make this happen effectively. The combination of intelligent systems tools and a broad range of applications introduces a need for a synergy of disciplines from science, technology, business and the humanities. The series will include conference proceedings, edited collections, monographs, handbooks, reference books, and other relevant types of book in areas of science and technology where smart systems and technologies can offer innovative solutions.

High quality content is an essential feature for all book proposals accepted for the series. It is expected that editors of all accepted volumes will ensure that contributions are subjected to an appropriate level of reviewing process and adhere to KES quality principles.

More information about this series at http://www.springer.com/series/8767

Sergey V. Zykov

Managing Software Crisis: A Smart Way to Enterprise Agility

Springer

Sergey V. Zykov
Higher School of Economics
National Research University
Moscow
Russia

ISSN 2190-3018 ISSN 2190-3026 (electronic)
Smart Innovation, Systems and Technologies
ISBN 978-3-319-77916-4 ISBN 978-3-319-77917-1 (eBook)
https://doi.org/10.1007/978-3-319-77917-1

Library of Congress Control Number: 2018938776

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG
part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To God, my teachers, and my family

Foreword

As computers and machines have become increasingly intelligent and capable of performing complex activities, the software that lies at the heart of every action is under more pressure to deploy successfully each and every time it is called upon. Yet companies rush the production cycle in order to be first to market; machines are called upon to operate in environments for which they were not designed; and enterprises lose business with computer software crashes. The resulting unhappy customers, expensive opportunity costs, and potentially unsafe operations have led to a crisis in software production. Thus, this book is timely in addressing one of the most challenging problems in technology and one that is mostly hidden—developing reliable flexible software that works "out of the box."

Indeed, software development has been studied for decades. Project management techniques have been applied to the software development life cycle and have led to recommendations for methods such as waterfall, agile, object-oriented, scrum, lean, iterative. However, rather than focusing on a single method, this book takes a broader approach and investigates software production complexity resulting from the interplay between software quality characteristics, technological factors, and human-related factors. Issues and best practices for software development are illustrated with case studies.

This book will be a valuable and thought-provoking read for anyone interested in software development. The authors are experts who have studied both the problems and the successes associated with software. Their combined wisdom will benefit the community and hopefully contribute to better software in the future.

Baltimore, MD, USA

Dr. Gloria Phillips-Wren, Ph.D.
Professor and Chair
Department of Information Systems
Law and Operations
The Sellinger School of Business and Management
Loyola University Maryland

Acknowledgements

I would like to thank the colleagues of mine who essentially contributed to this book. They clarified initially vague ideas and helped with translation, copyediting, proofreading, diagramming, etc. Many of them are the students who did their master/Ph.D. theses under my supervision. Some of their papers findings and takeaways were transformed and included into this book as case studies on agility improvement and crisis responses. They are: Vlad Abdulmianov, Eunice Agyei, Artem Aslanyan, Vera Ermakova, Nikita Fomichyov, Ramis Gabeydulin, Prof. Alexander Gromoff, Alexandra Gureeva, Mikhail Kupriyanov, Maria Mamontova, Dinara Nikolaeva, Isheyemi Olufemi, Victor Rotari, Gaurav Sharma, Grigory Shilin, Alexander Sivtsov, and Sabina Supibek.

I would like to thank the Springer executive editor Dr. Thomas Ditzinger and the Springer project coordinator for books production Mr. Ayyasamy Gowrishankar, for their continuous availability and prompt assistance.

In addition, I would like to express my deep appreciation and sincere gratitude to the editors in chief of the Springer series in Smart Innovation, Systems and Technologies, Prof. Lakhmi C. Jain and Prof. Robert J. Howlett, for their cooperative efforts in supporting my initiative.

Contents

Acronyms

3D	Three dimension
AAC	Airline Administrative Control
ABS	Automated banking system
ACARS	Aircraft Communications Addressing and Reporting System
ACDM	Architecture-centric design method
AI	Artificial intelligence
AOC	Aeronautical operational control
APEC	Asia-Pacific Economic Cooperation
API	Application programming interface
APL	Application programming language
ARIS	Architecture of integrated information systems
AST	Abstract syntax tree
ATAM	Architecture tradeoff analysis method
ATC	Air traffic control
BPM	Business process modeling
CABS	Centralized automated banking system
CAD	Computer-aided design
CASE	Computer-aided software engineering
CD	Continuous delivery
CEO	Chief executive officer
CERN	Conseil Européen pour la Recherche Nucléaire
CI	Continuous integration
CID	Common interface definition
CIDL	Common interface definition language
CMU	Carnegie Mellon University
CORBA	Common Object Request Broker Architecture
CPU	Central processing unit
CQRS	Command Query Responsibility Separation
CRM	Customer relationship management

CSV	Comma-separated values
DB	Database
DBMS	Database management systems
DOM	Document Object Model
DPU	Data processing unit
DRE	Direct-recording electronic
DSL	Domain-specific language
DSM	Domain-specific modeling
DSML	Domain-specific modeling language
EA	Enterprise architecture
EAI	Enterprise application integration
EMF	Eclipse Modeling Framework
EPL	Eclipse Public License
EQCS	Education quality control system
ER	Enterprise resource
ERP	Enterprise resource planning
ETL	Extract–transform–load
FIFA	Federation International of Football Association
FTP	File Transfer Protocol
GDP	Gross domestic production
GMF	Graphical Modeling Framework
GRPC	Google Remote Procedure Call
GUI	Graphical user interface
HTML	Hypertext Markup Language
IAAS (aka IaaS)	Infrastructure as a service
IDE	Integrated development environment
IDL	Interface definition language
IEEE	Institute of Electrical and Electronics Engineers
IIOP	Internet Inter-ORB Protocol
IOT (aka IoT)	Internet of Things
IS	Information system
ISO	International Organization for Standardization
IU	Innopolis University
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
KAIST	South Korea Advanced Institute of Science and Technology
KPI	Key performance indicator
KT	Knowledge transfer
LED	Light-emitting diode
LISP	List processing
LMS	Learning management system
LOM	Learning Object Metadata

LOP	Language-oriented programming
LW	Language workbench
MEPHI (aka MEPhI)	Moscow Engineering and Physics Institute
METAR	Meteorological Aerodrome Report
MIT	Massachusetts Institute of Technology
ML	Machine learning
MODS	Metadata Object Description Schema
MOOC	Massive open online course
MS	Microsoft
NATO	North Atlantic Treaty Organization
NLP	Natural-language processing
NPP	Nuclear power plant
ODBC	Open Database Connectivity
OER	Open educational resources
OOOI	Out, Off, On, In
ORB	Object request broker
OS	Operating system
PAAS (aka PaaS)	Platform as a service
PDF	Portable Document Format
PL	Production life cycle
PLM	Production life cycle management
PR	Public relations
PROLOG	PROgramming in LOGic
R&D	Research & Development
RAM	Random-access memory
RDF	Resource Description Framework
REST	Representational State Transfer
RPC	Remote procedure call
RUR	Russian ruble
SAAS (aka SaaS)	Software as a service
SE	Software engineering
SML	Standard Meta Language
SMS	Short Message Service
SOA	Service-oriented architecture
SOAP	Simple Object Access Protocol
SQL	Structured query language
SSL	Secure Sockets Layer
SVM	Support vector machine
TCP/IP	Transmission Control Protocol/Internet Protocol
TF.IDF	Term Frequency–Inverse Document Frequency
UI	User interface
UML	Unified Modeling Language
VHF	Very high frequency
VLDB	Very large database

VM	Virtual machine
VS	Visual Studio
XML	Extensible Markup Language
XPATH (aka XPath)	XML Path Language
ZKP	Zero-knowledge proof

Abstract

This book is about enterprise agility and crisis-resistant software engineering.

Chapter 1 gives an overview of the concept of enterprise agility and its impact on organizational flexibility. This includes an in-depth evaluation of modern enterprise-scale projects and case studies on their non-agility, which often results in disastrous consequences. The ways we suggest to avoid these mission-critical errors are based on the existing research results in the field. Another key aspect is the agility improvement by means of the architectural trade-offs. We outline a few types of crisis recovery strategies and determine their links to the software architectures. We consider mission-critical systems for deeper understanding of how to increase scalability and avoid potential design errors. We provide case studies of bridge construction as examples of careless crisis management and foundations for agility improvement. We also focus on the digital integrity protection and particularly blockchain technology, and its application to secure voting. Afterward, we discuss decentralized applications and using their bit-torrent sharing networks in cryptocurrencies. We present a case study on sentiment analysis and its application to the agility issues in crisis.

Chapter 2 discusses the concept of programming languages and their use in application development. We analyze the evolution of programming languages from primitive to high-level ones. We examine the concept of domain-specific languages and a few problem domains for their testing and verification under specific environments, the focus being on inter-process communication with CORBA and ICE technologies. This chapter also describes the embedded systems and how they promote agility in mission-critical systems.

Chapter 3 describes the best practices in service-oriented enterprise software development. We define Microservices and illustrate how they improve organizational agility. We analyze how service-oriented architectures and particularly Microservices differ from the monolithic approach and identify their potential application areas. We discuss implementations of continuous delivery and continuous integration of the application life cycle and illustrate their importance. We present a case of Microservices for the banking sector and investigate how it helps

to meet the requirements. We examine the integration of customer relationship management and geo-marketing and identify the business value of this synergy. We discuss the cloud services (and particularly virtual machines) as an agility booster; this includes in-depth testing of the proposed architecture.

Chapter 4 focuses on how the best software development practices depend on human-related factors; we further investigate the pattern-based approach as an agility driver. We discuss the principles of knowledge transfer and detect the key factors, which promote agile transfer in crisis. We look at the application of these factors to a Russian start-up, Innopolis University, and analyze the implications on the project flexibility improvement. We discuss patterns and anti-patterns of crisis-resistant development for increasing agility in mission-critical systems.

Keywords Agility · Architecture · Crisis resistance · Blockchain
Decentralized application · Life cycle model · Software product
CASE tool · Functional · Logical · Object-oriented · Domain-specific language
Reusability · Embedded system · Enterprise software development
Microservice · Customer relationship management · Banking · Cloud service
Best practice · Layer-based approach · Knowledge transfer · Design pattern

Introduction: Agility or Extinction

The focus of this book is smart agility management in crisis for large-scale software systems.

Marx explained crises and their nature. He stated that crises result from misbalanced production and the realization of a surplus value on the market [1]. The root cause of this misbalance is the separation between the producers and the means of production [2]. In software development, the nature of crises is somewhat different; a crisis is typically a disproportion between client's expectations and the actual product behavior.

Enterprise systems are typically complex; they combine a large number of hardware and software components. Managing the development of such complex software products, even under uncertainties and in crisis, is a key aim of software engineering. This discipline emerged in the 1960s as a response to the so-called software crisis. This term originated from the critical development complexity because of overwhelming computing power. In 1967, the issue became so critical that NATO had to arrange an invitation-only conference to find an immediate solution. The conference was held in Germany; its key participants were such gurus in computer science as A. Perlis, E. Dijkstra, F. Bauer, and P. Naur. These researchers and practitioners were Turing Award winners and the NATO Science Committee representatives from the USA, Germany, Denmark, and the Netherlands. At that time, the complexity of the hardware and software systems became unmanageable by the traditional methods and techniques. A large number of software products were late, over budget, or totally unsuccessful. To clarify this state of software production management, F. Bauer introduced the term "crisis" at the conference; E. Dijkstra used it later in his Turing Award lecture. The participants suggested "software engineering" (this term is also attributed to F. Bauer) as a remedy for that crisis. The idea was applying engineering methods of material production to the emerging large-scale software development in order to create better measurable and less uncertain products, i.e., to achieve a new agility level.

Now, we are on the same page concerning the crisis. However, what is agility? Intuitively, it is clear. For some experts, agile means flexible. For the other experts, agile means adjustable. Before giving a formal definition, let us turn to an example.

One can easily imagine a dancing puma or cat; these are typical instances of agile behavior or agility. However, can you imagine a waltzing elephant or a bigger (and more outdated) creature, like a mammoth or a dinosaur?

Why do we turn to large ancient beasts and waltzing? This is a *metaphor*. Metaphor is often used as an agile software engineering technique in order to disseminate a novel software system concept among the shareholders.

In our case, waltzing is an example of activity that requires agility. An elephant or a mammoth is a large-scale company which is an instance of a slow responsive and hard-to-adjust actor. The challenges of agile management are easier to track and monitor by means of large-scale company cases. An old-fashioned dinosaur example refers to an enterprise with complex and somewhat outdated management patterns and legacy computer systems. State-of-the-art business environment requires increased responsiveness, which is similar to following a tune while waltzing. Moreover, waltzing requires complex motion and constant coordination of the dancer's body parts.

Waltzing also requires a partner, which involves another level of coordination. This dancing style has a clearly recognizable pattern, which is certainly different from any other, such as salsa or polka. Though following a specific style might appear too difficult for an elephant or a mammoth, performing in any different style from the partner will likely result in a crash. This is due to an unexpected and/or unpredictable behavior, which in fact is an example of a crisis. Conversely, to perform harmoniously, i.e., to avoid a crisis, the partners need to coordinate (i.e., monitor and adjust) their actions following the same style. Therefore, to reach this agile harmony the partners need to master the waltzing process through training which requires both personal and team activities.

Currently, the Earth faces a global warming, which looks so slow that certain individuals do not notice it. As the warming accelerates, it may result dramatically for these non-responsive individuals in just a few decades. In crisis, it is very risky to remain old-fashioned and non-agile. Dramatic business climate change, uncertainties of resources, business and technical requirements, emerging and collapsing markets are radical and may result in critical consequences. In crisis, agility requires instant attention as insufficient agility is synonymous with extinction.

Agility is related to balancing business requirements and technology constraints. Many local crises result from misbalancing of these two aspects. Therefore, a well-balanced software solution means better agility. In other words, agility is a remedy for crisis. In crisis, agility is vital for any kind of a business structure. However, agility is better observed and understood in large-scale structures such as enterprises. Thus, building enterprise-scale systems requires a good balance. Not only should this balance be present in the design and construction, but also it should be present in each and every stage of the enterprise system life cycle. In the case of an enterprise, its agility is a concept that incorporates the ideas of "flexibility, balance, adaptability, and coordination under one umbrella" [3].

The early crisis in software development and the recent global economic crisis taught us a few lessons. One very important lesson is that the so-called human factor errors, which result from critical uncertainties and undisciplined life cycle

management, often dramatically influence the software quality, and project success. Our systematic approach to the impact of these human-related factors on agile enterprise system development embraces the perspectives of business processes, data flows, and system interfaces. These three perspectives correspond to dynamic, static, and system architectural views. For each of the perspectives, we identify a set of business management levels, such as strategic decision-making or everyday management. After we combine these perspectives and the business management levels, we get the enterprise engineering matrix (Fig. 1).

BUSINESS PROCESSES	DATA FLOWS	SYSTEM TYPES
STRATEGY	STRATEGY	BI / PORTAL
INTEGRATION / KNOWLEDGE ANALYSIS	*METAKNOWLEDGE = = WISDOM*	*METAKNOWLEDGE = = WISDOM*
RELATIONSHIP MGMT	RELATIONSHIP MGMT	CRM / SCM
INTEGRATION / DATA ANALYSIS	*METADATA = = KNOWLEDGE*	*METADATA = = KNOWLEDGE*
RESOURCE PLANNING	RESOURCE PLANNING	ERP
PRODUCTION PLANNING	*SUPPLIES / ORDERS*	*SUPPLIES / ORDERS*
ACCOUNTING, DAILY MGMT	ACCOUNTING, DAILY MGMT	MES
PRODUCTION MGMT (PLANT LEVEL)	*TECHNOLOGY MAPS*	*TECHNOLOGY MAPS*
SUPERVISORY CONTROL	SUPERVISORY CONTROL	SCADA
TELEMETRY DATA COLLECTION/ HARDWARE DEVICE MGMT	*CLEAN DATA*	*CLEAN DATA*
DATA STORAGE	DATA STORAGE	DB / DWH
ANALOG-TO-DIGITAL	*RAW DATA*	*RAW DATA*
DEVICES/ SENSORS	DEVICES/ SENSORS	SENSOR / BOT

Fig. 1 Enterprise agility matrix

Another and perhaps a better name for this is the enterprise agility matrix as it determines enterprise agility. This matrix allows the detection of mission-critical dependencies in human (and other) factors for the systemic properties of the software products. These dependencies are based on relationships between certain values of the matrix cells. As such, we can build a set of constraints to guard the software development process from critical design errors and give an early warning of risky decisions. The matrix allows for agility estimation in terms of process management, data integrity, and interface quality. Informally, it will indicate whether the system set to be designed will behave like a puma or like a mammoth in an unstable, uncertain, or crisis environment. It will also recommend how to design an agile system that naturally accommodates to digital transformation.

This book is organized as follows. Chapter 1 covers the key concepts, such as agility and crisis, in more detail; it outlines crisis-resistant agility improvements for the age of digital transformation including blockchain-based decentralized software for transparent voting and a sentiment analysis application. Chapter 2 describes the history of programming languages and their agility in terms of large-scale software development; it discusses languages for domain-specific applications. Chapter 3 investigates the evolution of services in software development; it focuses on agile approaches including cloud computing and Microservices. Chapter 4 addresses agile and crisis-resistant pattern- and practice-based software development; it also discusses the human factors which promote software-related knowledge transfer. The conclusion summarizes the key outcomes of the book; it suggests agile ways to smartly manage software development in crises.

This book will recommend crisis adjustments and improvements in order to maintain agility and competitiveness in the global environment change. These recommendations are typically real case-based. Certainly, this book will not give a universal solution for agility. However, it will recommend technologies and approaches to start the new style of agile "dancing."

References

1. Chakravarty et al. (2013). Supply chain transformation: Evolving with emerging business paradigms. In *Springer Texts in Business and Economics* 2014th Edition, Kindle Edition.
2. Lowry, P. B., & Wilson, D. (2016). Creating agile organizations through IT: The influence of internal IT service perceptions on IT service quality and IT agility. *Journal of Strategic Information Systems (JSIS), 25*(3), 211–226. Available at SSRN: https://ssrn.com/abstract=2786236.
3. Chen, C., Liao, J., & Wen, P. (2014) Why does formal mentoring matter? The mediating role of psychological safety and the moderating role of power distance orientation in the Chinese context. *International Journal of Human Resource Management, 25*(8), 1112–1130.