RESEARCH

Bernhard Turban

Tool-Based Requirement Traceability between Requirement and Design Artifacts

Springer Vieweg

Tool-Based Requirement Traceability between Requirement and Design Artifacts

Bernhard Turban

Tool-Based Requirement Traceability between Requirement and Design Artifacts

Foreword by Prof. Dr. Christian Wolff

Springer Vieweg

Bernhard Turban
Nabburg, Germany

Turban, Bernhard: Tool-Based Requirements Traceability between Requirement and Design Artifacts for Safety-Critical Systems
Zugl.: Regensburg, Univ., Diss., 2011

This work was accepted as a Ph. D. dissertation thesis by the Faculty of Languages, Literature and Cultural Studies of the University of Regensburg in 2011.

D 355

ISBN 978-3-8348-2473-8 ISBN 978-3-8348-2474-5 (eBook)
DOI 10.1007/978-3-8348-2474-5

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Library of Congress Control Number: 2013933878

Springer Vieweg
© Springer Fachmedien Wiesbaden 2013
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law. The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use. While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer Vieweg is a brand of Springer DE.
Springer DE is part of Springer Science+Business Media.
www.springer-vieweg.de

While the discerning layman understands that
in the design of large constructions,
a new town or an airport, the problems are overwhelming,
he probably does not realise so clearly
that there are problems just as pressing
and difficult for the designer
in the design of almost any trivial product.
A bad town will do more harm than a bad toothbrush
but the designer of either will experience his job
as the necessity to make a series of decisions
between alternative courses of action,
each affecting the decisions which come after it;
and if no life hangs on the outcome
of the series of decisions about the toothbrush,
the livelihood of several people does.
David Pye [Py78; p.75]

Foreword

What is the way design decisions are made in Software design and implementation? What is the relationship between a software artifact and customer requirements? What are the reasons, what is the rationale for a specific technical solution? How should design decisions be documented? These are only some of the questions which Bernhard Turban tackles in his dissertation on *Tool-Based Requirements Traceability*.

One of the major merits of this book is the successful bridging from design theories to practical tool design for embedded real-time software: Bernhard Turban actually puts design theory to work, in a way from which software designers and engineers may directly benefit. At the same time, this effort is firmly rooted in current software engineering standards like SPICE (Software Process Improvement and Capability Determination, ISO/IEC 15 504).

Tackling the documentation needs for software design decisions by implementing a tool using a specific algorithm or forwarding these decisions shows the authors inventiveness: For a problem many software engineers are constantly confronted with, this solution provides an innovative solution. At the same time, this approach generates traceability-relevant information.

In addition, the author does not only present a plausible and functional algorithm for documenting design decisions across different levels of the development process, he also realizes a complex interactive interface tool which seamlessly adds to the functionality of modeling tools. Based on this work, a commercial software development tool was created.

This work was developed not in an academic context, but in an industrial setting within a group of software engineers working in the domain of automotive embedded real-time systems. Thus, the author can draw all examples for his work from immediate observations in the development projects he was working on. This adds to the credibility of the work presented here, and I am sure that both academia as well as industrial software design can learn a good deal lot from Bernhard Turban's work.

With the complexity of software projects still rising, the demand for better documentation and traceability will grow beyond typical fields like the engineering of embedded systems. Therefore, it is to be hoped for that many software projects will benefit from Bernhard Turban's theoretical approach towards design decisions as well as from the tool solutions he has created.

Prof. Dr. Christian Wolff

Acknowledgements

This work would not have been accomplished without the support of so many people. I would like to thank them all for their support.

University of Regensburg
To begin with, I would like to thank my supervisors at the University of Regensburg Professor Dr. Christian Wolff and Professor Dr. Rainer Hammwöhner for their constant support. I specially thank them for giving me the chance to write this thesis.

University of Applied Sciences in Regensburg
Particularly, I would like to thank Professor Dr. Athanassios Tsakpinis from the University of Applied Sciences Regensburg and director of the Competence Center for Software Engineering. Without his very significant support the results described here might not have been accomplishable.

I further want to thank Professor Dr. Markus Kucera and Professor Dr. Bernhard Kulla for their advice and support.

Former Micron Electronic Devices AG
Further, I would like to thank Peter Schiekofer and Jörg Aschenbrenner for giving me a chance to perform my doctor's thesis with the Micron Electronic Devices AG and specially thank them for their open-mindedness to the vague ideas I first sketched to them seeing the innovative potential within the ideas.

Mercedes Benz technology (MBtech)
At the MBtech Group, I would like to thank Dr. Nico Hartmann for giving the R2A-project a home, after the integration of Micron Electronic Devices AG.

The PROVEtech:R2A development team
I also would like to thank the R2A development team for their good work and enthusiasm.

My Editors
I also want to thank Florian Weiss and my brother Andreas Turban for cross-reading my thesis. Futher, I especially want to thank Anita Wilke from Springer Fachmedien Wiesbaden GmbH for helping me bringing this thesis to a book.

Family and Friends
Last but not least, I would like to especially thank my parents, grandparents and all my friends for their patience and encouragements in difficult situations.

Bernhard Turban

Abstract

Developing *safety-critical systems* imposes special demands for ensuring quality and reliability of the developed systems. Process standards such as SPICE (ISO15504) or CMMI have been developed to ensure high quality processes, leading to the development of high quality systems. Central principles of these standards are demands for *requirements traceability*. *Traceability* means comprehensible documentation of all origins and later influences of a requirement throughout the complete development endeavor. Among other uses ascribed, the *traceability* concept tries to ensure that every requirement is adequately considered in development and that if changes on the requirement are needed, *impacts* of these changes can be adequately estimated and consistently implemented later on. Even though the *traceability* concept seems promising in theory, it faces substantial problems in practice. One problem is that despite the needed efforts, the perceived benefits for developers are often low because the quality of captured *traceability* information is often coarse grained, does not prove helpful in the situational context, or has already degraded.

This thesis tries to show that *traceability* between requirements and design is an especially difficult problem. To analyze the problem context, the thesis at first analyzes theories, in which the problem is cross-cutting. These are *embedded systems development*, *systems engineering*, *software engineering*, *requirements engineering* and *management*, *design* theory and *process standards* for *safety-critical systems*.

This analysis mainly identifies a twofold gap between the requirements and the design domain. Obviously a tooling gap exists because different tools are used for the requirements and design domain. However, more important, between requirement descriptions and designs a substantial inherent gap exists because design is a creative decision process of designers often guided by *intuition* and *tacit knowledge* thus difficult to trace by current *traceability* concepts. To prove this argumentation, the author analyzes four design theories (*symbolic information processing* (Simon), *wicked problems* (Rittel), *reflective practice* (Schön) and *patterns* (Alexander)). As a solution to the gap problem, the thesis introduces a tool-based *traceability* method that supports designers in their thinking, avoids disturbing designers in their intuitive phases of creativity, allows establishing *traceability* nearly as a *by-product*, provides *early benefit* to designers, improves collaboration between designers and extends usual *traceability* concepts by two *integrated decision models* allowing further decision information (*rationale*) to be documented. The *decision models* also allow deriving new design internal

"requirements" (*design constraints* and *budgeted resource constraints*) as consequences. In this way, it is possible to clearly distinguish real *requirements* originating from customers from 'requirements' arising from internal decision processes during design leading to the definition of a '*requiremental items taxonomy'*. As the thesis further shows, these concepts also prove to be helpful to avoid unnecessary redundancies in the *artifact process models* of SPICE (ISO15504) or CMMI, where different requirement (*system requirements*, *hardware requirements* and *software requirements*) and design artifacts (s*ystem design*, *hardware design* and *software design*) are considered in their interplay. Last but not least, mechanisms for *graphical impact analysis*, *consistency management* and *supplier management* complete the approach.

Through funding of the support program IUK-Bayern, the results presented here could be integrated into a commercial tool solution called PROVEtech:R2A, now offered by the MBtech Group as a decisive means to significantly improve requirement-based design processes with improved support to achieve real benefit from the *traceability* concept.

Contents

List of Figures

List of Tables

Abbreviations

The following lists the most common abbreviations used in this thesis over several chapters:

AIS	Actual Impact Set
AN	Abstraction Node – a concept of R2A (cf. ch. III.15)
ANH	Abstraction Nodes Hierarchy – a concept of R2A (cf. ch. III.15)
A-SPICE	Automotive SPICE (cf. ch. I.7.4)
BRC	Budgeted Resource Constraint – a concept of R2A (cf. ch. III.21)
CCB	Change Control Board
CMMI	Capability Maturity Model integrated (cf. ch. I.7)
COTS	Commercial Off The Shelf
CRS	Customer Requirements Specification
CTM	Conceptual Traceability Model
CusSysDes	The Customer's System Design
DC	A Design Contraint as a concept of R2A (part III)
DEC	A conflict based Decision a concept of R2A (part III)
DOD	United States Department of Defense
DRL	Decision Representation Language an RatMan approach (cf. ch. II.9)
DXL	DOORS eXtension Language
ECU	Embedded Control Unit
EEPROM	Electrically Erasable Programmable Read Only Memory
EIS	Estimated Impact Set
FR	Functional Requirement
GUI	Graphical User Interface
GUID	General Unique IDentifier
HMI	Human Machine Interface

HIS	Hersteller Initiative Software – Standardization Board of German Automotive OEMs (cf. ch. I.7)
HW	Hardware
HW_RS	Hardware Requirements Specification
IBIS	Issue Based Information System an RatMan (cf. ch. II.9) approach (see also gIBIS)
IDE	Integrated Development Environment
ISO	International Standards Organization
MF	Measurement Framework (see SPICE)
NFR	Nonfunctional Requirement
OCL	Object Contraint Language
PAM	Process Assessment Model (see SPICE)
PRM	Process Reference Model (see SPICE)
QOC	Questions, Options, Criteria an *RatMan* approach (cf. ch. II.9)
R2A	PROVEtech:R2A – The tool environment resulting from this research (part III)
RatMan	Rationale Management (cf. ch. II.9)
RDP	Requirements Dribble Process a heuristic supported by R2A (part III)
REM	Requirements Engineering and Management
REQ	Requirement from the customer as a concept of R2A (part III)
RI	Requiremental Item a concept of R2A (part III)
RIF	Requirement Interchange Format
RIS	Requirement Influence Scope a concept of R2A in connection with the RDP (part III)
ROM	Read Only Memory
RUP	Rational Unified Process
RE	Requirements Engineering
RM	Requirements Management
REM	Requirement Engineering and Management
RMS	Rationale Management System

RSD	Requirement Source Document as a concept of R2A (part III)
RTF	Rich Text Format
SE	Software Engineering
SEI	Software Engineering Institute (SEI) of the Carnegie Mellon University in Pittsburg
SIL	Safety Integrity Level as described in IEC 61508
SIS	Starting Impact Set
SPICE	Software Process Improvement Capability dEtermination (IS0 15504), (cf. ch. I.7)
SysEng	Systems Engineering (ch. I.4)
SYS_RS	System Requirements Specification
SuppRS	Supplier Requirements Specifications
SW	Software
SW_RS	Software Requirements Specification
SysML	System Modeling Language
TQM	Total Quality Management (cf. ch. I.7)
UML	Unified Modeling Language

Introduction

Nothing is more powerful in the world than an idea whose time has come.
Victor Hugo (*)

Introduction to the Topic

Usually, systems developed by humans are not developed for their own sake of existence. Instead, these systems shall help to achieve certain human goals or purposes. Goals or purposes, however, are often very abstract and vague in the same way as the usage situations of these systems are manifold and complex. Correspondingly, a more precise definition of what a system must exactly perform is needed. This leads to the need for defining the exact requirements of a system. Then, such a system must just be designed and constructed to fulfill the defined requirements.

Concerning the development of software-based systems, development experiences of the last decades have been rather disenchanting. Often, five out of six development projects are considered as rather unsuccessful [BMH+98; p.3], [St95], [St01], [Eb05; p.23ff]. One major issue identified through the years is that the developed systems often do not achieve the goals and purposes they were intended for, or if they fulfill them, the resulting system's development project significantly has exceeded planned budget and (resp. or) effort [St95], [St01].

Research on the causes for these problems is ongoing. Among others, three issues can be identified as root causes (cf. ch. I.5): Unclear requirements, often changing requirements and inadequate processes for handling.

One approach to solve the first problem is to spend extra effort on identifying and defining clear and adequate requirements upfront. Today, a whole set of artifacts, heuristics, practices and processes around the topic requirements are available summarized under the theory of *requirements engineering* (*RE*). However, development experiences have shown that even though extra focus and effort is spent upfront on the definition of requirements, changing requirements are still more the norm than the exception. As ch. I.5.6 shows, reasons are manifold.

B. Turban, *Tool-Based Requirement Traceability between Requirement and Design Artifacts,*
DOI 10.1007/978-3-8348-2474-5_1, © Springer Fachmedien Wiesbaden 2013

In the author's opinion, at least two essential causes for the requirements change problem exist:

1. Software (SW) and SW-based systems are abstract and thus essentially difficult to define comprehensively.

2. In addition, SW-based systems themselves with their intercorrelations with other systems and their embedding into processes infer a significant complexity leading to the problem that not all cases and eventualities can be considered beforehand.

These causes – among others described in ch. I.5.6 – significantly challenge the paradigm that the extensive specification and analysis of requirements upfront will tame the requirements change problem. They might rather be a good leverage to mitigate the problem, but changing requirements will still remain a decisive factor for projects. *RE*-theory also seems to have acknowledged this fact in the way that it more and more emphasizes the aspect that requirements must also be adequately managed (see ch. I.5.3). Thus, the author rather prefers to speak of *requirements engineering and management* (*REM*).

In *REM* theory, *requirements traceability* (in the following simply called *traceability*) is considered as central means to manage requirement changes. *Traceability* means "comprehensible documentation of requirements, decisions and their interdependencies to all produced information resp. artifacts from project start to project end" ([RS02; p.407 (*)]). Through recorded *traceability* information, *impact analysis* of changes is possible allowing estimating the *impact* of suggested requirement changes. This information allows project stakeholders to decide, whether the benefits of a requirement change outweigh its costs, thus avoiding disadvantageous changes. Once it is decided to perform a change, *traceability* helps to consistently propagate the change to all *impacted* locations in a project. Thus, consistently inferring the change into the project prevents dangers of forgetting to change affected locations leading to defects or even fatal consequences. In this way, the *traceability* concept is a promising means to improve *REM* and especially change management processes, thus avoiding inconsistencies – introduced during inevitably applied changes – leading to failures in the system, thus leading to significantly improved quality of developed systems.

Even though the *traceability* concept is already known for over 20 years and it always has seemed very promising to be a significant value gain in a project, it is still not very widely spread in development practice except for development projects under certain circumstances. As ch. II.10.5 tries to outline, this seems to

be the case, because it suffers from a general problem of efficiency and of low direct benefit perceived by the project members intended to capture the *traceability* information.

The quality of developed systems generally is a decisive factor. On the other side, ensuring quality involves significant efforts and costs. Even though quality must not necessarily be seen as a cost factor, but should rather be seen as a factor of investment, only finite resources can be spent for quality in order to ensure economic success. For once, this appeals to ensuring a high degree of effectiveness on quality assurance methods in general. For the other, demands for quality may differ concerning the purpose of the system. As an example, it may be an acceptable risk for PC-based SW systems that some minor bugs or other minor flaws remain undiscovered in a delivered system, because applying an update on a PC is acceptable as long as the number of updates is acceptable to the users and it is easy to apply the updates. Concerning embedded systems steering a technical equipment, it is much more difficult to perform SW-updates, as this in most cases implies a product recall to apply the new software update. Besides high costs, this is rather not acceptable for the users and often involves significant image losses for the involved companies. Beyond that, so called *safety-critical systems* exist, where a malfunction can lead to significant damages to values or even impose hazards for persons' health or lives. In these cases, even minimal probabilities of failures involving injury or death of persons must be best possibly eliminated.

Another important means to ensure good product quality is to employ good development processes. In the context of embedded projects and especially for *safety-critical* embedded projects, significant efforts have been undertaken to standardize the processes with their decisive characteristics to be performed in order to achieve high quality outcomes. Ch. I.7 describes these efforts and the demands for these processes. In these process standards, a demand crosscutting through all engineering processes is the demand for *traceability* of every requirement to the influences it imposes on every artifact developed in any engineering process.

The implementation of these demands in practice, however, often makes apparent that these demands themselves are difficult to implement and if they are implemented it is highly questionable whether the effort and resources spent really bring significant benefit to development projects. Instead, *traceability* demands are often rather performed to correspond to the standards' demands.

In this thesis, the author tries to identify several core reasons for these problems. Besides the *benefit problem* mentioned above, an essential problem is that different tools are used for different processes. This, however, implies that the *traceability* concept must somehow cross these tool gaps in order to connect the

information within the different tools. In the author's opinion, this actually is one essential cause for the *benefit problem*, as crossing these gaps generally requires higher efforts, decreases accuracy and significantly increases potentials for inconsistencies.

Unfortunately, the author considers one problem as even more essential: This problem origins from the fact that requirements describe a *problem space* that must be transformed into a solution. This transformation process is usually referred to as design. Usual *traceability* models rather assume that these connections between requirements and design artifacts are rather linear semantic allowing to trace these connections.

The author, however, believes that a semantic gap exists between the *problem space* described by requirements and the solution found. This gap exists, because design is a complex task of performing sequences of complex design decisions leading to the solution. There, the connections being rather nonlinear make it very difficult to record valuable *traceability* information.

As a way to address these problems identified, this thesis also introduces a tool environment called PROVEtech:R2A (R2A) to support *requirements traceability* to design with specific focus on diminishing both mentioned gaps. In this way, the author also hopes to diminish the benefit gap to a degree that collecting *traceability* information provides direct benefit for the designers thus hoping to really achieve the promises of the *traceability* concept.

Context of this Thesis Project

In order to provide a better understanding to the reader how the research results described in this thesis have emerged, this chapter provides a short overview about the history of this research project.

First ideas to some core problems and features addressed by R2A arose as a consequence of the direct development experiences of the author in an automotive ECU development project for lights steering with SPICE level two processes. At that time, the Micron Electronic Devices AG (MEDAG) and the Competence Center for Software Engineering (CC-SE) at the University of Applied Sciences Regensburg have begun a collaboration with the goal to improve the connection of theoretic research with industrial practice.

In the development project, from 2004 to 2005 the author worked as representative of the CC-SE at MEDAG where the author was at first responsible for

introducing *REM*-processes with the *REM-tool* IBM Rational DOORS[1] to be newly introduced into the company's project practice. During further development, the author was responsible for module design and implementation. In this way, the author was also responsible for maintaining the *requirements traceability* to the module design directly experiencing the shortcomings and problems involved.

These experiences have lead to the idea about a tool environment, where designers should directly benefit from gathered *traceability* information by making the influences of requirements on design directly visible to designers (basic ideas of ch. III.13, ch. III.15 and ch. III.18.2.2) and by improving the collaboration of all involved designers (basic ideas of ch. III.18.2.4).

In 2005 the identified key concepts have then been formulated in a theoretic outline with an extended theoretical case study being reviewed by representatives from MEDAG and CC-SE. The concepts proved promising. As the concepts also base on extensive user interaction, where usability is a key factor for success, the project made contact to the Institute for Media, Information and Cultural Studies at the University of Regensburg, where usability is one major research topic.

The three organizations have decided to form a partnership to realize the project. For this goal, the partners decided to develop a prototype tool evaluating the theoretical results by practical feedback and to apply for financial aid at the IUK[2]-program of the Bavarian Ministry of Economic Development.

During the application phase in 2006, the prototype tool implementation has been developed and has been continuously assessed by design practitioners of the partners to achieve immediate feedback of implemented features.

With these granted financial aids, a two years project for six persons could be realized to transfer the achieved theoretical and prototypical research results into a solution relevant for practice. The project has been performed from Feb. 2007 to Feb. 2009 leading to the commercial tool PROVEtech:R2A as it is discussed in this part. Because the tool's features have been considered as very innovative, where good usability at complex user interactions is essential, and because most core features have been extensively analyzed upfront by theoretical discussion and the prototype, the project members decided to develop the project using the evolutionary prototyping concept from agile development methods. Evolutionary prototyping means that the project started with a prototype where all identified features were successively integrated into the prototype so that the prototype

---

1 At that time called Telelogic DOORS

2 The IUK program (In German: Information Und Kommunikation (Information and Communication)) is a research funding program to support transferring newest research results into commercial solutions applicable in practice.

successively evolves to the final product. In this way, new features could at first be realized via a prototype implementation. These features then could be introduced to design practitioners to acquire direct feedback on the prototypical implementation. This feedback could then be used to improve and *refactor* the implementation to fully integrate it into the project's program base. Concerning the tool's architectural design, therefore, only an *architectural skeleton* has been developed sketching the core concepts of the tool environment and leaving details of the architecture open for change.

This proceeding may, at first, seem to contradict principles discussed in this thesis about *REM*, but, as discussed in ch. I.5.6 and ch. I.6.2.2, prototype-based requirement evaluation is a common practice to address the problem that highly innovative projects face a high volatility of requirements.

During the project in the midst of 2008, the MEDAG has been taken over by the MBtech Group GmbH & Co. KGaA (in the further simply called MBtech) a subsidiary company of the Daimler AG specialized on engineering services. The concepts and ideas of the project convinced the MBtech of the innovative potentials of the tool leading to a continued endeavor to develop the results to a commercial solution. In this way, the developed tool has been named PROVEtech:R2A[3] (called R2A in the following) and has been integrated into the PROVEtech tool family.

Currently, R2A is offered as commercial solution of the MBtech to address the *traceability* problems described in this thesis. It is continuously maintained and improved through a half-year release cycle. In this way, the project described here also is an example of how theoretic research results can be successfully brought into commercial project practice.

[3] R2A stands for Requirements 2 Architecture. Further information on PROVEtech:R2A can be found at the company homepage: http://www.mbtech-group.com/eu-en/electronics_solutions/tools_equipment/provetechr2a_traceability_management/traceability_management.html (Access: 2010/09).

General Remarks on this Thesis

Before stepping into the thesis, the reader should note some general remarks.

Registered Trademarks

The reader of this thesis should note that some mentioned techniques and tools referred to in this thesis are registered trademarks or under protection of copyright laws.

Argumentation

The thesis introduced here is not an empirical study, but rather a theoretical work. The work can be considered somewhere between *systems engineering* and *software engineering* theory. As a matter of fact, many of the mentioned theories and 'facts' presented in this thesis have no irrevocable evidence but are to a certain degree a 'fact' of experience, interpretation and believe. When the author collected these 'facts' from different sources, dangers of misinterpretation or selective interpretation by the author cannot be excluded. Facts found in a research paper cannot always be seen on their own. Often, these 'facts' are embedded in a certain context (e.g., a special research theory or project). Now, taking conclusions from these 'facts' should be done with a certain care. To address this problem, the author often considered not only to cite the pure 'fact' concluded somewhere, but also tried to outline the context where these 'facts' have arisen and he also tried to provide available possible alternative interpretations by other authors, or theories to allow the reader to derive his (her) own conclusions about the evidence and how cogent the author's argumentation is. As a matter of fact, however, most theories are not compatible or consistent to each other. Correspondingly, a technique to outline the context of some argumentation may also result in some inconsistency or contradictory statements. The reader should consider these inconsistencies or contradictions as phenomenon of the manifold complexity that research theories produce in their connection to each other and the limited capabilities of humans to completely cope with these complexities. Besides, the author generally doubts the potential existence of one grand unified theory about systems and software development. Rather the author considers inconsistencies and contradictions as spring of new knowledge in research.