SQuBOK Project Team

Concise version:

Guide to the Software Quality Body of Knowledge (SQuBOK®)

Version 1

Originally published in Japanese by Ohmsha Ltd. (Tokyo, Japan), November 2007

This is an <u>ABRIDGED TRANSLATION</u> of the original Japanese version. SQuBOK® Project Team has investigated the feasibility of publication in other countries with this abridgment and will really appreciate your comments including answers to questionnaire in the end.

Please send your comments to: juse@juse.or.jp or Fax: +81-3-5378-1220

* PDF file of this document can be downloaded from: http://www.juse.or.jp/software/squbok-eng.html

JUSE: The Union of Japanese Scientists and Engineers

JSQC : The Japanese Society for Quality Control

ISBN 978-4-274-50162-3 (original)

The Union of Japanese Scientists and Engineers (JUSE)

1-2-1, Koenji-Minami, Suginami-ku, Tokyo 166-0003 JAPAN

http://www.juse.or.jp/

Mail: juse@juse.or.jp

Phone: +81-3-5378-9812

Fax: +81-3-5378-1220

Foreword:

On the publication of SQuBOK Guide Version 1

Yoshinori Iizuka

SQiP Steering Board (formerly SPC Steering Board) Chair

Professor, University of Tokyo

The Union of Japanese Scientists and Engineers (JUSE) established its Software Production Control Board (“SPC Board”) in 1980 in response to the application of quality management to software—in short, the marriage of software engineering and TQM (total quality management). Since its establishment, the SPC Board has focused on spreading valuable methodologies for enhancement of the competitiveness of the Japanese software industry as part of its efforts to systematize the practical study of software quality. Until the mid-1990s, the Board’s activities consisted primarily of research into the application of the TQM philosophy, methodology, and techniques established primarily in the manufacturing industry to software. The Board went on to accommodate structural changes in the software industry symbolized by the move in the early 1990s toward networking, open systems, downsizing, and multimedia, and later adjusted its approach in response to the building of new development and maintenance paradigms.

Today, quality is only becoming more important as we move toward a world characterized by the ubiquitous use of embedded software. We changed the Board’s name from SPC to SQiP (Software Quality Profession; pronounced “skip”) in September 2007 as a way of reaffirming the significance of continuing to contribute to the field of software quality and our resolution to do so. It is a commitment that evokes a profession concerned with software quality and the professionals that work in it.

Looking back, the development of a body of knowledge concerning the software quality as applied to a new age in software development was to prove to be an essential part of SPC activities. Whatever the particular product or service, achieving quality requires a motivation to achieve quality, thought and value systems concerning quality, technologies (methodologies) that give expression to quality, management methodologies that make the most of technology, and methodologies for encouraging the appetites, knowledge, and technical skills of the people involved in associated operations. No exception, software requires a body of knowledge related to these concepts and methodologies with a focus on quality.

In keeping with an awareness of these facts, the SPC Steering Board set forth the clearly defined objective of developing the SQuBOK (Guide to the Software Quality Body of Knowledge; pronounced “sku-bok”) in the spring of 2005 to serve as the means of access the body of knowledge, and Akira Sakakibara (IBM Japan), Yasuharu Nishi (The University of Electro-Communications), Juichi Takahashi (Sony Computer Entertainment), and Makoto Nonaka (Toyo University) embarked on a preparatory examination of the issues. Based on these efforts, the SQuBOK Project Team was formed in September 2005 to begin investigating the content of the work with Yasuko Okazaki (IBM Japan) as leader and Nishi and Nonaka as sub-leaders. This work proceeded in the form of a joint project in conjunction with the Software Group of the Japanese Society for Quality Control led by Takeshi Kaneko (Musashi Institute of Technology).

In their activities, the project set out to improve awareness of software quality technologies and offer support to organizations seeking to establish a software quality process by formalizing Japan’s implicit knowledge concerning software quality in a way that would play a useful role in training software quality engineers in line with the SQiP philosophy, and by organizing and systematizing the most recent software quality themes.

The SQuBOK Guide was conceived to serve as an exhaustive source of Japanese-language documents readily available in Japan, including good domestic case studies that had been publicized, in an effort to provide a structural visualization of useful knowledge accumulated by the Japanese software quality community.

The Alpha Version was compiled in April 2006 thanks to the energetic efforts of all those involved. This choice of edition name indicated that the guide was not yet a complete work. The pre-print was provided to a limited audience of experts capable of determining whether the content, still at an early stage, was sufficiently developed. This group of the following 13 individuals provided some 350 comments: Motoei Azuma (Waseda University), Takehisa Okazaki (IBM Japan), Hideto Ogasawara (Toshiba), Kenji Ogawa (IT Skill Standards Center, Information-Technology Promotion Agency), Ryuzo Kaneko (NEC Communication Systems), Takeshi Kaneko (Musashi Institute of Technology), Kiyohiro Kawai (ASP Communications), Masanori Kikumoto (Japan Novel), Motomu Komura (System SWAT), Susumu Sasabe (NEC Communication Systems), Shunichi Fukuyama (Tottori University of Environmental Studies), Fumiaki Hotta (Japan Advanced Institute of Science and Technology), and Toshiyuki Doi (Kyowa Exeo).

To create the Version 1, the project team of 14 individuals that put together the Alpha Version was expanded to 27, and the project reorganized the body of knowledge and compiled additional documentation over the course of a year and several months based on the comments that had been received for the Alpha Version. The result of their efforts was the Beta Version, which was released in September 2007 for public comment. Then the project incorporated feedback from these comments as appropriate and added documentation for the lowest level of topics (approximately 200 items) to complete the Version 1. During this time, the team received receptive yet rigorous reviews from advisors Mitsuru Oba (Hiroshima City University), Tohru Matsuodani (Debug Engineering Institute and Hosei University), and Katsuyuki Yasuda (Tsukuba International University).

Looking back, I have been interested in software quality and involved with the SPC Board for more than 20 years, and I have served as the SPC chairman for more than 10 years, all of them marked by the ongoing transformation in the software industry. Words fail to express the satisfaction of seeing this document that I had long known we must compile—a document that is absolutely essential to the SPC Board and indeed to the larger software quality community—come to fruition.

No field of endeavor starts out with a system of knowledge ready-made. This is especially true for the practical sciences. Such systems are developed by accumulating, visualizing, and structuring theory related to the basic methodologies necessary for achieving the goals of the field and its development as well as countless nuggets of often fragmentary knowledge expressed by a great many exceptional individuals who have come before us, some of them teachers by negative example. What we call learning is by no means the product of an intellectual game played by scholars, but rather the systematization of practice to facilitate the reuse of knowledge. I cannot help but feel pride in being able to present on behalf of the new SQiP Software Quality Committee the Version 1 of this work and overwhelming emotion as one who is interested in this field. I feel this way because when I sought to make some kind of contribution to the field of software quality, this work was necessary above all else.

I believe we will be able to take a significant stride towards the establishment of a software quality methodology by using this SQuBOK Guide as a foundation. It will be possible to present the framework for an educational and training curriculum. It can also serve as a basis for building models for planning human resources development and career paths.

In addition to taking this opportunity to once again thank all of the individuals who took responsibility for reviewing the Alpha Version draft, Alpha Version, Beta Version draft, and Beta Version so that the Version 1 could be published, I would like to ask for your continued understanding of the essence of SQuBOK—the ongoing development and evolution of this body of knowledge—and to request that you continue to provide rigorous yet constructive comments to that end.

I admire the ability of Yasuko Okazaki to lead so many occasionally fractious veterans in order to bring the project to this level of completion. I am also moved by the diligence of the members of the SQuBOK Formulating Subcommittee of the SQiP Software Quality Committee and the SQuBOK Study Group of the Japanese Society for Quality Control’s Software Group. I would like to take this opportunity to thank you all.

SQuBOK will play a central role in the practical research and popularization projects of the SQiP Software Quality Committee. I look forward to your continued support and leadership for both SQiP and the SQuBOK.

SQuBOK Project Team (at completion of the Version 1)

Leader:

Yasuko Okazaki (IBM Japan, Ltd.)

Sub-leaders:

Yasuharu Nishi (The University of Electro-Communications)

Makoto Nonaka (Toyo University)

Keizo Tatsumi (Fujitsu Limited)

Members:

Shinji Fukui (Omron Corporation)

Satoshi Fushimi (Information and Mathematical Science Laboratory Inc.)

Naomi Honda (NEC Corporation)

Shuji Honma (CSK Systems Corp.)

Noriko Iizumi (Hitachi High-Technologies Corporation)

Akira Ikeda (Hitachi Information & Communication Engineering, Ltd.)

Kazuo Kawai (Nil Software Corp.)

Keiko Koga (Hitachi, Ltd.)

Yoshinobu Machida (NTT Data Corporation)

Kouichi Miyagi (Osaka Gas Information System Research Institute Co., Ltd.)

Kiyoshi Mukai (Sumisho Computer Systems Corporation)

Takamasa Nara (NARA Consulting)

Keiko Nishio (Panasonic Mobile Communications Co., Ltd.)

Yoshiko Ogawa (Bank of Creativity Co., Ltd.)

Susumu Ohno (Nihon Kohden Corporation)

Tetsutaro Okawa (Nihon Unisys, Ltd.)

Kenji Onishi (Mamezou Co., Ltd.)

Akira Sakakibara (IBM Japan, Ltd.)

Tatsuya Shinozawa (INES Corporation)

Hideyuki Tabuchi (Mizuho Information & Research Institute, Inc.)

Hironori Washizaki (National Institute of Informatics)

Yoshimichi Watanabe (University of Yamanashi)

Tsuneo Yamaura (Tokai University)

Advisors

Tohru Matsuodani (Debug Engineering Institute, Hosei University)

Mitsuru Oba (Hiroshima City University)

Katsuyuki Yasuda (Tsukuba International University)

Introduction

Yasuko Okazaki

1. Regarding the SQuBOK Project Team

The SQuBOK Project Team, which launched its activities with a kick-off meeting on September 20, 2005, is a volunteer-based group that started out with 10 participants. Even when our ranks had swelled to 14 by the time we completed the 64-page Alpha Version on April 28, 2006, we didn’t have enough people working on the project, and we made an appeal before beginning to compile the Beta Version to companies and people involved in the industry who were not yet participating, ultimately increasing the size of our team to the current 27 people. Group members belong to either the SQuBOK Formulating Subcommittee of the Union of Japanese Scientists and Engineers’ SQiP Software Quality Committee (previously the SPC Board) or the SQuBOK Study Group of the Japanese Society for Quality Control’s Software Group, but the SQuBOK Project Team is at the core of actual work.

2. Objectives

We had the following five objectives in compiling the Version 1 of the SQuBOK Guide as a Japan’s original BOK guide:

1. To help train individuals involved with quality assurance
2. To formalize Japan’s implicit knowledge concerning software quality
3. To organize and systematize new themes concerning software quality
4. To improve awareness of software quality technologies
5. To assist organizations seeking to establish software quality assurance processes

The first objective, helping to train individuals involved with quality assurance, indicates that the Version 1 of the SQuBOK Guide was compiled for an assumed audience of engineers involved with quality assurance. It was necessary to limit the scope of the first edition in order to ensure our ability to bring the SQuBOK Guide to fruition during a short period of time, and we elected to start with knowledge areas related to quality assurance activities and assume a readership of individuals involved with quality assurance. We are examining whether the scope of the work can be enlarged for second and subsequent editions. Areas where we are examining such changes for second and subsequent editions are indicated with asterisks (*) in the tree diagrams provided in Figures 1 through 4 of the introductory chapter.

The second objective, formalizing Japan's implicit knowledge concerning software quality, we sought to formalize the exceptional knowledge and experience of the field's domestic forerunners, information that is not included in BOK (body of knowledge) guides published in Europe and the United States. To achieve this goal, we solicited the cooperation and participation of as broad a group of companies and universities as possible when organizing the SQuBOK Project Team. Facing the mass retirement of baby boomer IT experts, we also sought to take advantage of this opportunity to visualize their knowledge and experience.

The third objective, organizing and systematizing new themes concerning software quality, indicates that the SQuBOK Guide should systematize a variety of scattered knowledge related to software quality and provide a means of accessing existing systems of knowledge. With IT now playing an important role in our social infrastructure, recent years have seen a rapid increase in the knowledge and domains with which IT engineers must be familiar, including service level agreements (SLAs), the Information Technology Infrastructure Library (ITIL), security, IT skill standards, agile development as expressed in eXtreme Programming (XP), software product lines, aspect oriented programming (AOP), *Guidelines for Improving Reliability of Information Systems* as published by the Ministry of Economy, Trade and Industry, and newly published international standards. Although several good dictionary-like resources concerning software quality management have been available in the past, there is little in the way of a resource with comprehensive content that also includes recent information. In light of this fact, we have sought to enable the Guide to serve as a hub for accessing knowledge related to software quality. It is by no means intended to supplant other systems of knowledge that include software quality knowledge (for example, PMBOK, a project management BOK, or SWEBOK, a software engineering BOK [ISO/IEC TR 19759: 2005]), but rather includes parts extracted from those works relating to software quality.

The fourth objective, improving awareness of software quality technologies, indicates our intention to provide a proper assessment to workers involved with software quality by informing others of the fact that software quality is underpinned by extensive and specialized technologies.

The fifth and last objective is to assist organizations seeking to establish software quality assurance processes. We hope that organizations seeking to establish such assurance processes in the future will be able to make use of this Guide.

3. Target audience

As described in relation to our first objective, the Version 1 assumes a readership of engineers involved with quality assurance. However, engineers with design and programming responsibilities can use it to gain an understanding from the standpoint of how to best evaluate the merits of the specifications and code they develop. We trust that the information it contains will help point the way toward the design and development of high-quality specifications and code.

……

Contents

Introductory Chapter: SQuBOK Guide Outline

(1) SQuBOK tree diagrams

Figures 1 through 4 provide tree diagrams describing the structure of SQuBOK.

The SQuBOK Guide divides knowledge areas into the three general categories of “Fundamental Concept of Software Quality”, “Software Quality Management”, and “Software Quality Methods” (see Figure 1). The initial category of “Fundamental Concept of Software Quality” classifies fundamental concepts and approaches concerning software quality. The next category of “Software Quality Management” classifies activities for managing quality. The final category of “Software Quality Methods” classifies specific methods, ranging from metrics and quality planning techniques to operational and maintenance techniques.

Each category is organized into a hierarchy of knowledge areas, knowledge sub-areas, and topics. More specifically, the initial category of “Fundamental Concept of Software Quality” (see Figure 2) consists of the two knowledge areas of “Quality Concept” and “Quality Management”. The “Quality Concept” knowledge area consists of the six knowledge sub-areas of “Definition of Quality Concept (History)”, “Software Quality Model”, “Dependability,” “Security”, Usability”, and “Safety”. Additionally, the “Definition of Quality Concept (History)” knowledge sub-area includes 10 topics ranging from “Definition of Quality (Gerald M. Weinberg)” to “Definition of Quality (ISO/IEC 25000)”.

The “Software Quality Management” category (See Figure 3) includes three sub-categories between the category and knowledge area level due to the large amount of content it covers: “Organizational Level”, organizing knowledge areas where management and action often occur at the organizational level; “Project-level (Common)”, organizing knowledge areas that are common to various development phases; and “Project-level (Specific)”, organizing knowledge areas describing specific development phases (see Figures 3 and 5). Under the sub-categories is the same hierarchy of knowledge areas, knowledge-sub areas, and topics.

The “Software Quality Methods” category (see Figure 4) uses the same hierarchical structure with knowledge areas, knowledge sub-areas, and topics.

In this way, the tree diagrams have four or five levels, consisting of categories, (sub-categories), knowledge areas, knowledge sub-areas, and topics.

(2) SQuBOK Guide Organization

Chapters 1, 2, and 3 explain the sub-categories (S-CA), knowledge areas (KA), and knowledge sub-areas (S-KA) for each of the three categories (CA). The description from sub-categories to knowledge sub-areas both serves as a pointer to the lower levels of the hierarchy and includes simple definitions and objectives. Explanatory information for the lowest level of topics (T) includes headings such as “Overview”, “Related topics and knowledge areas”, “References”, and “Further Readings”.

Readers seeking more information are recommended to consult the “References” and “Further Readings” sections as well as the “List of Recommended Reading/Papers” appendix A. Items under the “Related topics and knowledge areas” heading for each topic are also a useful resource for additional information. To get the chapter and section numbers listing the “Related topics and knowledge areas” heading, see Appendix E, “Index”.

Appendix A, “List of Recommended Readings/Papers” provides a list of carefully selected papers recommended by authoring team members. Authors worked to provide an extensive selection of papers written in Japanese for the convenience of domestic Japanese engineers.

Appendix B, “List of References/Further Readings” introduces the sources cited or referenced in the Guide as well as Further Readings. Some content overlaps with Appendix A. Appendix C, “List of standards” presents standards that have been covered as topics or referenced in the text, and that the authors determined should be presented in the form of a list based on their content or degree of influence.

Appendix D, “List of Award-winning Papers” lists papers that have been recognized at symposia and in other academic settings.

Appendix E, “Index” provides a list of all knowledge areas, knowledge sub-areas, and topics. To learn more about a given knowledge area, knowledge sub-area, or topic, consult Appendix E or the tree diagrams presented in Figures 1 through 4 to find the relevant chapter and section number.

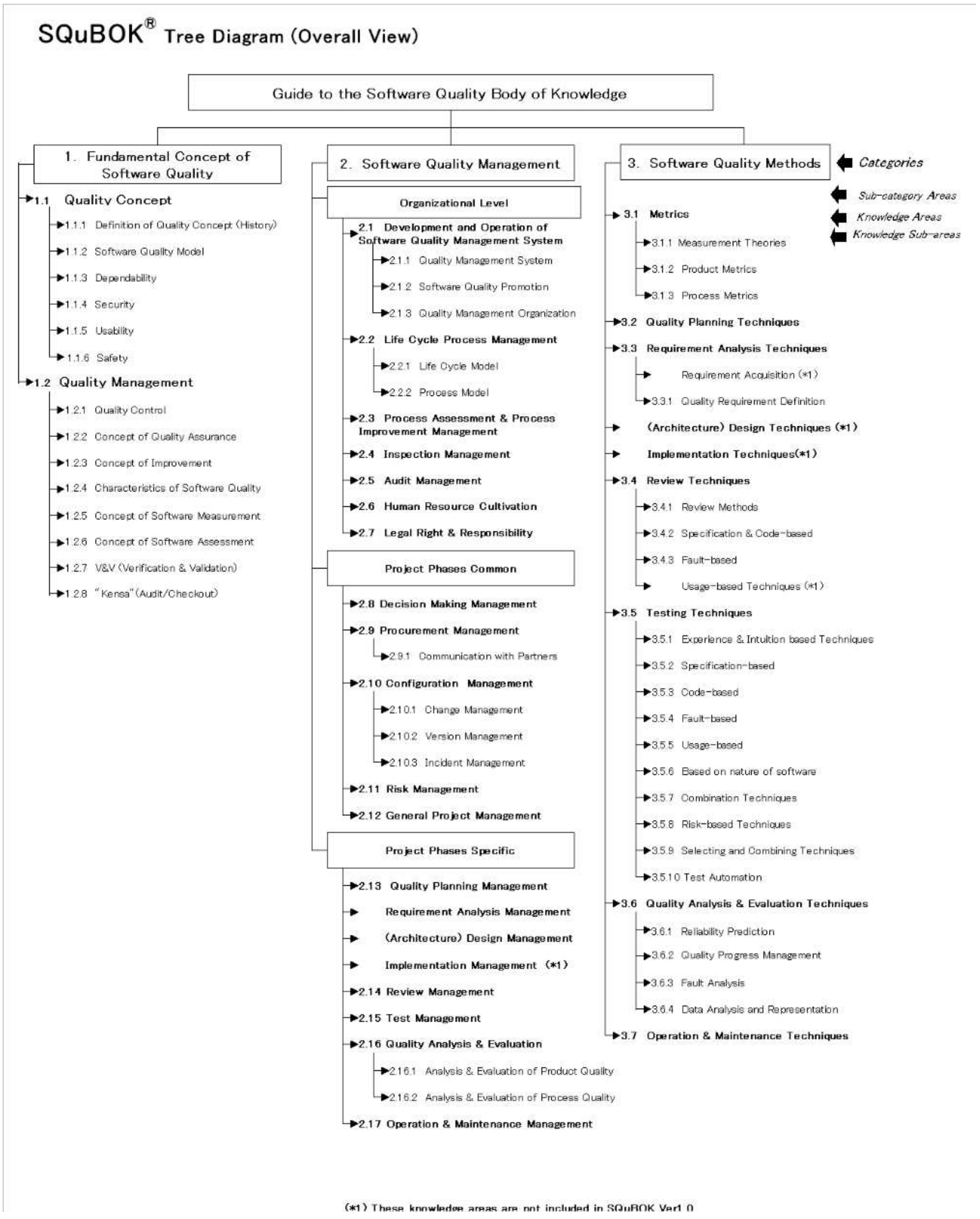
SQuBOK® Tree Diagram (Overall View)
Guide to the Software Quality Body of Knowledge
1. Fundamental Concept of Software Quality
2. Software Quality Management
3. Software Quality Methods
Categories
Sub-category Areas
Knowledge Areas
Knowledge Sub-areas
1.1 Quality Concept
1.1.1 Definition of Quality Concept (History)
1.1.2 Software Quality Model
1.1.3 Dependability
1.1.4 Security
1.1.5 Usability
1.1.6 Safety
1.2 Quality Management
1.2.1 Quality Control
1.2.2 Concept of Quality Assurance
1.2.3 Concept of Improvement
1.2.4 Characteristics of Software Quality
1.2.5 Concept of Software Measurement
1.2.6 Concept of Software Assessment
1.2.7 V&V (Verification & Validation)
1.2.8 "Kensa" (Audit/Checkout)
Organizational Level
2.1 Development and Operation of Software Quality Management System
2.1.1 Quality Management System
2.1.2 Software Quality Promotion
2.1.3 Quality Management Organization
2.2 Life Cycle Process Management
2.2.1 Life Cycle Model
2.2.2 Process Model
2.3 Process Assessment & Process Improvement Management
2.4 Inspection Management
2.5 Audit Management
2.6 Human Resource Cultivation
2.7 Legal Right & Responsibility
Project Phases Common
2.8 Decision Making Management
2.9 Procurement Management
2.9.1 Communication with Partners
2.10 Configuration Management
2.10.1 Change Management
2.10.2 Version Management
2.10.3 Incident Management
2.11 Risk Management
2.12 General Project Management
Project Phases Specific
2.13 Quality Planning Management
Requirement Analysis Management
(Architecture) Design Management
Implementation Management (*1)
2.14 Review Management
2.15 Test Management
2.16 Quality Analysis & Evaluation
2.16.1 Analysis & Evaluation of Product Quality
2.16.2 Analysis & Evaluation of Process Quality
2.17 Operation & Maintenance Management
3.1 Metrics
3.1.1 Measurement Theories
3.1.2 Product Metrics
3.1.3 Process Metrics
3.2 Quality Planning Techniques
3.3 Requirement Analysis Techniques
Requirement Acquisition (*1)
3.3.1 Quality Requirement Definition
(Architecture) Design Techniques (*1)
Implementation Techniques(*1)
3.4 Review Techniques
3.4.1 Review Methods
3.4.2 Specification & Code-based
3.4.3 Fault-based
Usage-based Techniques (*1)
3.5 Testing Techniques
3.5.1 Experience & Intuition based Techniques
3.5.2 Specification-based
3.5.3 Code-based
3.5.4 Fault-based
3.5.5 Usage-based
3.5.6 Based on nature of software
3.5.7 Combination Techniques
3.5.8 Risk-based Techniques
3.5.9 Selecting and Combining Techniques
3.5.10 Test Automation
3.6 Quality Analysis & Evaluation Techniques
3.6.1 Reliability Prediction
3.6.2 Quality Progress Management
3.6.3 Fault Analysis
3.6.4 Data Analysis and Representation
3.7 Operation & Maintenance Techniques
(*1) These knowledge areas are not included in SQuBOK Ver1.0

Figure 1. "Overall View of Tree Diagram" (upper 4 layers, from category level to sub-knowledge area level)

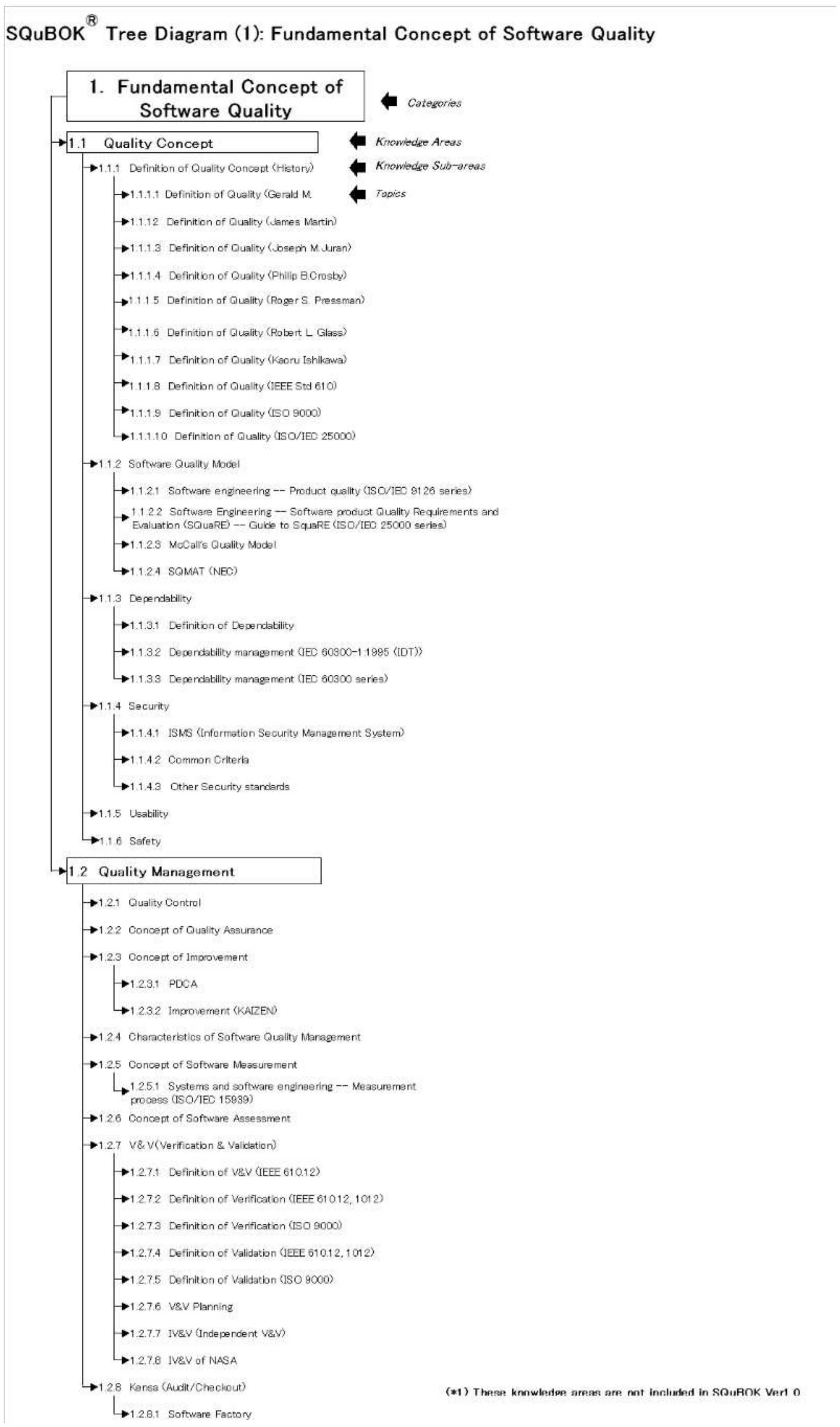
SQuBOK® Tree Diagram (1): Fundamental Concept of Software Quality
1. Fundamental Concept of Software Quality
Categories
1.1 Quality Concept
Knowledge Areas
1.1.1 Definition of Quality Concept (History)
Knowledge Sub-areas
1.1.1.1 Definition of Quality (Gerald M.
Topics
1.1.12 Definition of Quality (James Martin)
1.1.1.3 Definition of Quality (Joseph M.Juran)
1.1.1.4 Definition of Quality (Philip B.Crosby)
1.1.1.5 Definition of Quality (Roger S. Pressman)
1.1.1.6 Definition of Quality (Robert L. Glass)
1.1.1.7 Definition of Quality (Kaoru Ishikawa)
1.1.1.8 Definition of Quality (IEEE Std 610)
1.1.1.9 Definition of Quality (ISO 9000)
1.1.1.10 Definition of Quality (ISO/IEC 25000)
1.1.2 Software Quality Model
1.1.2.1 Software engineering -- Product quality (ISO/IEC 9126 series)
1.1.2.2 Software Engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SquaRE (ISO/IEC 25000 series)
1.1.2.3 McCall's Quality Model
1.1.2.4 SQMAT (NEC)
1.1.3 Dependability
1.1.3.1 Definition of Dependability
1.1.3.2 Dependability management (IEC 60300-1:1995 (IDT))
1.1.3.3 Dependability management (IEC 60300 series)
1.1.4 Security
1.1.4.1 ISMS (Information Security Management System)
1.1.4.2 Common Criteria
1.1.4.3 Other Security standards
1.1.5 Usability
1.1.6 Safety
1.2 Quality Management
1.2.1 Quality Control
1.2.2 Concept of Quality Assurance
1.2.3 Concept of Improvement
1.2.3.1 PDCA
1.2.3.2 Improvement (KAIZEN)
1.2.4 Characteristics of Software Quality Management
1.2.5 Concept of Software Measurement
1.2.5.1 Systems and software engineering -- Measurement process (ISO/IEC 15939)
1.2.6 Concept of Software Assessment
1.2.7 V& V(Verification & Validation)
1.2.7.1 Definition of V&V (IEEE 610.12)
1.2.7.2 Definition of Verification (IEEE 610.12, 1012)
1.2.7.3 Definition of Verification (ISO 9000)
1.2.7.4 Definition of Validation (IEEE 610.12, 1012)
1.2.7.5 Definition of Validation (ISO 9000)
1.2.7.6 V&V Planning
1.2.7.7 IV&V (Independent V&V)
1.2.7.8 IV&V of NASA
1.2.8 Kensa (Audit/Checkout)
1.2.8.1 Software Factory
(*1) These knowledge areas are not included in SQuBOK Ver1.0

Figure 2. "Fundamental Concept of Software Quality" Category (all layers, from category level to topic level)

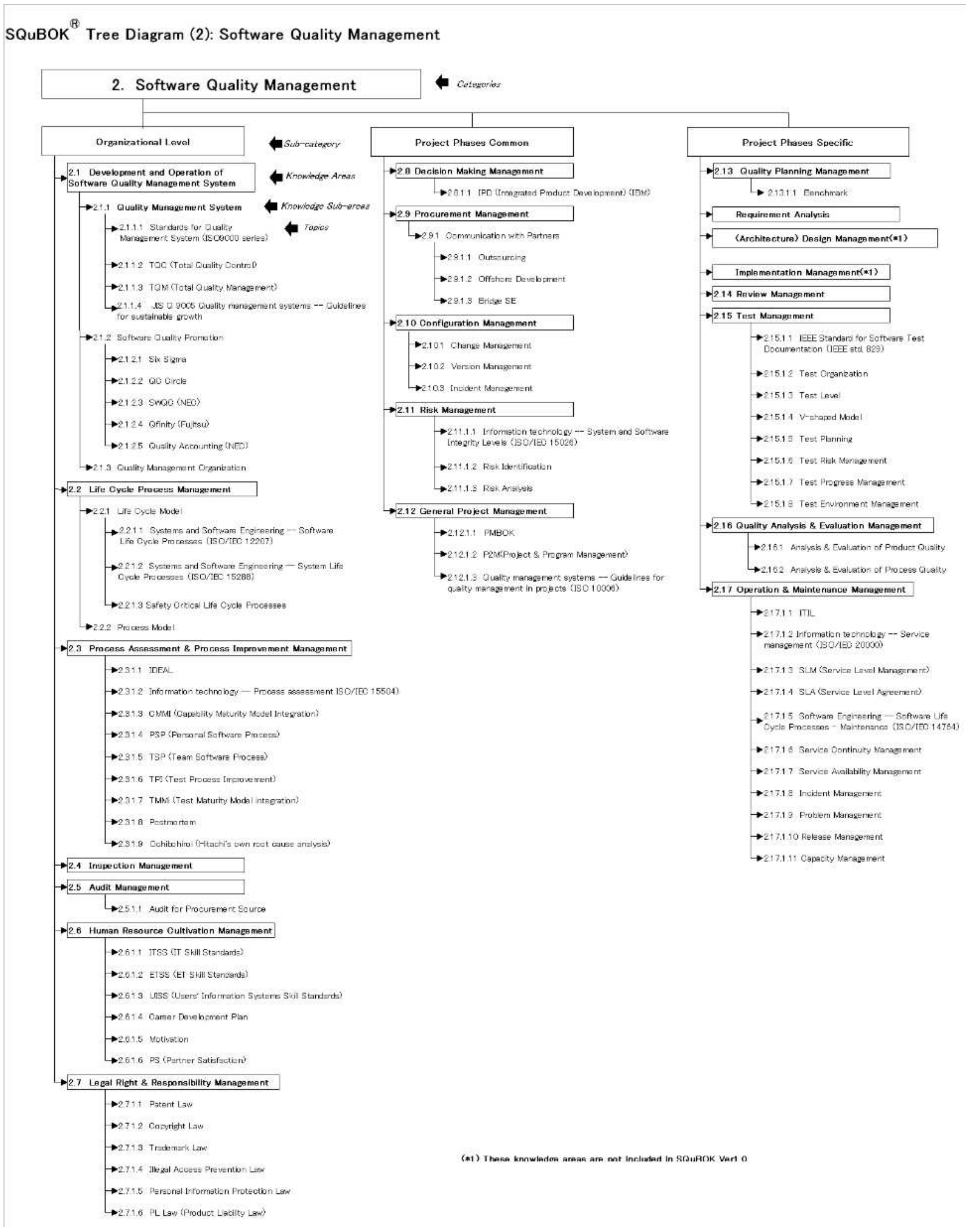
SQuBOK® Tree Diagram (2): Software Quality Management
2. Software Quality Management
Categories
Organizational Level
Sub-category
Project Phases Common
Project Phases Specific
2.1 Development and Operation of Software Quality Management System
Knowledge Areas
2.1.1 Quality Management System
Knowledge Sub-areas
2.1.1.1 Standards for Quality Management System (ISO9000 series)
Topics
2.1.1.2 TQC (Total Quality Control)
2.1.1.3 TQM (Total Quality Management)
2.1.1.4 JIS Q 9005 Quality management systems -- Guidelines for sustainable growth
2.1.2 Software Quality Promotion
2.1.2.1 Six Sigma
2.1.2.2 QC Circle
2.1.2.3 SWQC (NEC)
2.1.2.4 Qfinity (Fujitsu)
2.1.2.5 Quality Accounting (NEC)
2.1.3 Quality Management Organization
2.2 Life Cycle Process Management
2.2.1 Life Cycle Model
2.2.1.1 Systems and Software Engineering — Software Life Cycle Processes (ISO/IEC 12207)
2.2.1.2 Systems and Software Engineering — System Life Cycle Processes (ISO/IEC 15288)
2.2.1.3 Safety Critical Life Cycle Processes
2.2.2 Process Model
2.3 Process Assessment & Process Improvement Management
2.3.1.1 IDEAL
2.3.1.2 Information technology — Process assessment ISO/IEC 15504)
2.3.1.3 CMMI (Capability Maturity Model Integration)
2.3.1.4 PSP (Personal Software Process)
2.3.1.5 TSP (Team Software Process)
2.3.1.6 TPI (Test Process Improvement)
2.3.1.7 TMMi (Test Maturity Model integration)
2.3.1.8 Postmortem
2.3.1.9 Ochitchiroi (Hitachi's own root cause analysis)
2.4 Inspection Management
2.5 Audit Management
2.5.1.1 Audit for Procurement Source
2.6 Human Resource Cultivation Management
2.6.1.1 ITSS (IT Skill Standards)
2.6.1.2 ETSS (ET Skill Standards)
2.6.1.3 UISS (Users' Information Systems Skill Standards)
2.6.1.4 Career Development Plan
2.6.1.5 Motivation
2.6.1.6 PS (Partner Satisfaction)
2.7 Legal Right & Responsibility Management
2.7.1.1 Patent Law
2.7.1.2 Copyright Law
2.7.1.3 Trademark Law
2.7.1.4 Illegal Access Prevention Law
2.7.1.5 Personal Information Protection Law
2.7.1.6 PL Law (Product Liability Law)
2.8 Decision Making Management
2.8.1.1 IPD (Integrated Product Development) (IBM)
2.9 Procurement Management
2.9.1 Communication with Partners
2.9.1.1 Outsourcing
2.9.1.2 Offshore Development
2.9.1.3 Bridge SE
2.10 Configuration Management
2.10.1 Change Management
2.10.2 Version Management
2.10.3 Incident Management
2.11 Risk Management
2.11.1.1 Information technology -- System and Software Integrity Levels (ISO/IEC 15026)
2.11.1.2 Risk Identification
2.11.1.3 Risk Analysis
2.12 General Project Management
2.12.1.1 PMBOK
2.12.1.2 P2M(Project & Program Management)
2.12.1.3 Quality management systems — Guidelines for quality management in projects (ISO 10006)
2.13 Quality Planning Management
2.13.1.1 Benchmark
Requirement Analysis
(Architecture) Design Management(*1)
Implementation Management(*1)
2.14 Review Management
2.15 Test Management
2.15.1.1 IEEE Standard for Software Test Documentation (IEEE std. 829)
2.15.1.2 Test Organization
2.15.1.3 Test Level
2.15.1.4 V-shaped Model
2.15.1.5 Test Planning
2.15.1.6 Test Risk Management
2.15.1.7 Test Progress Management
2.15.1.8 Test Environment Management
2.16 Quality Analysis & Evaluation Management
2.16.1 Analysis & Evaluation of Product Quality
2.16.2 Analysis & Evaluation of Process Quality
2.17 Operation & Maintenance Management
2.17.1.1 ITIL
2.17.1.2 Information technology -- Service management (ISO/IEC 20000)
2.17.1.3 SLM (Service Level Management)
2.17.1.4 SLA (Service Level Agreement)
2.17.1.5 Software Engineering — Software Life Cycle Processes - Maintenance (ISO/IEC 14764)
2.17.1.6 Service Continuity Management
2.17.1.7 Service Availability Management
2.17.1.8 Incident Management
2.17.1.9 Problem Management
2.17.1.10 Release Management
2.17.1.11 Capacity Management
(*1) These knowledge areas are not included in SQuBOK Ver1.0

Figure 3. "Software Quality Management" Category (all layers, from category level to topic level)

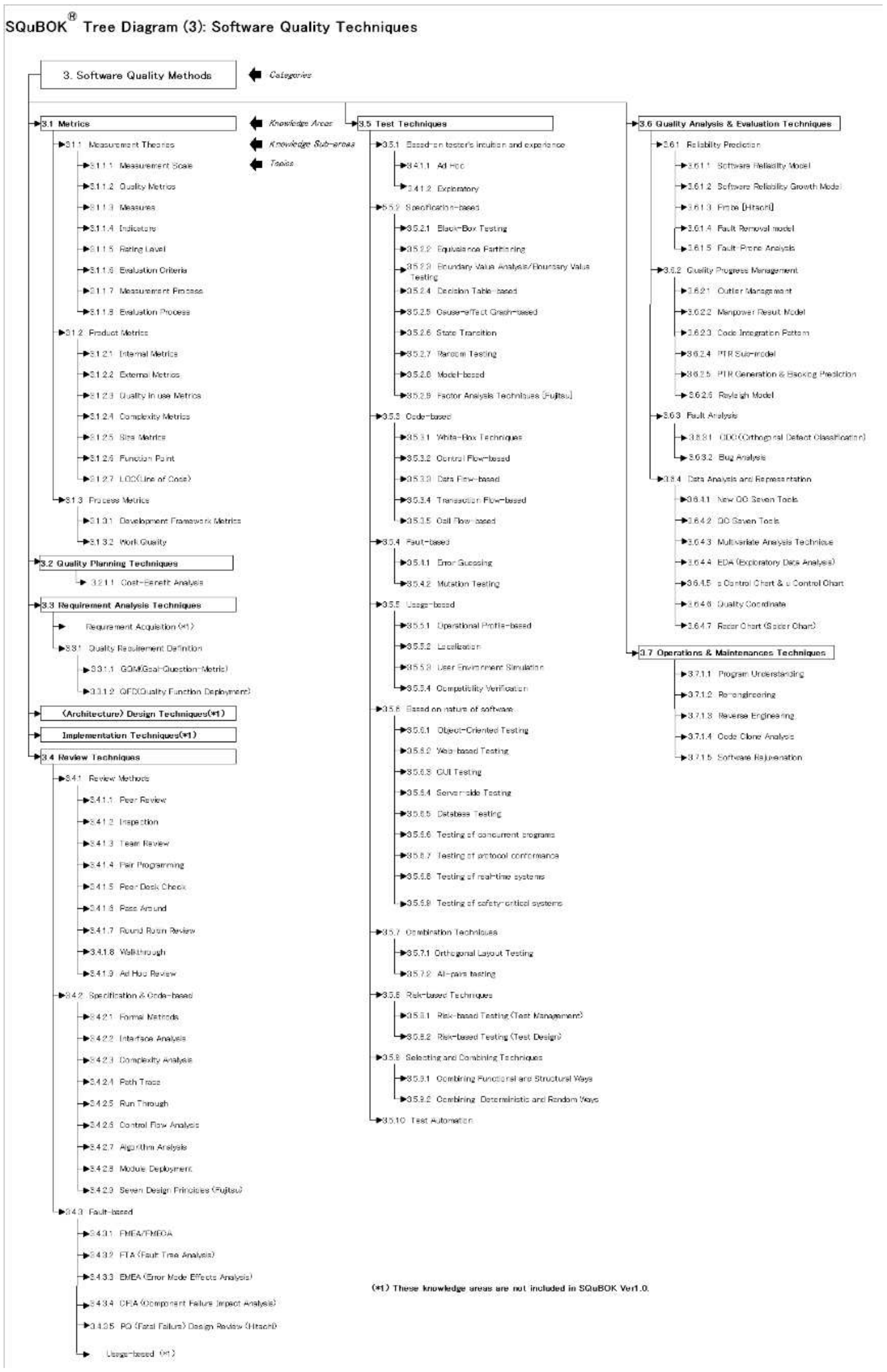
SQuBOK® Tree Diagram (3): Software Quality Techniques
3. Software Quality Methods
Categories
Knowledge Areas
Knowledge Sub-areas
Topics
3.1 Metrics
3.1.1 Measurement Theories
3.1.1.1 Measurement Scale
3.1.1.2 Quality Metrics
3.1.1.3 Measures
3.1.1.4 Indicators
3.1.1.5 Rating Level
3.1.1.6 Evaluation Criteria
3.1.1.7 Measurement Process
3.1.1.8 Evaluation Process
3.1.2 Product Metrics
3.1.2.1 Internal Metrics
3.1.2.2 External Metrics
3.1.2.3 Quality in use Metrics
3.1.2.4 Complexity Metrics
3.1.2.5 Size Metrics
3.1.2.6 Function Point
3.1.2.7 LOC(Line of Code)
3.1.3 Process Metrics
3.1.3.1 Development Framework Metrics
3.1.3.2 Work Quality
3.2 Quality Planning Techniques
3.2.1.1 Cost-Benefit Analysis
3.3 Requirement Analysis Techniques
Requirement Acquisition (*1)
3.3.1 Quality Requirement Definition
3.3.1.1 GQM(Goal-Question-Metric)
3.3.1.2 QFD(Quality Function Deployment)
(Architecture) Design Techniques(*1)
Implementation Techniques(*1)
3.4 Review Techniques
3.4.1 Review Methods
3.4.1.1 Peer Review
3.4.1.2 Inspection
3.4.1.3 Team Review
3.4.1.4 Pair Programming
3.4.1.5 Peer Desk Check
3.4.1.6 Pass Around
3.4.1.7 Round Robin Review
3.4.1.8 Walkthrough
3.4.1.9 Ad Hoc Review
3.4.2 Specification & Code-based
3.4.2.1 Formal Methods
3.4.2.2 Interface Analysis
3.4.2.3 Complexity Analysis
3.4.2.4 Path Trace
3.4.2.5 Run Through
3.4.2.6 Control Flow Analysis
3.4.2.7 Algorithm Analysis
3.4.2.8 Module Deployment
3.4.2.9 Seven Design Principles (Fujitsu)
3.4.3 Fault-based
3.4.3.1 FMEA/FMECA
3.4.3.2 FTA (Fault Tree Analysis)
3.4.3.3 EMEA (Error Mode Effects Analysis)
3.4.3.4 CFIA (Component Failure Impact Analysis)
3.4.3.5 FQ (Fatal Failure) Design Review (Hitachi)
Usage-based (*1)
3.5 Test Techniques
3.5.1 Based-on tester's intuition and experience
3.4.1.1 Ad Hoc
3.4.1.2 Exploratory
3.5.2 Specification-based
3.5.2.1 Black-Box Testing
3.5.2.2 Equivalence Partitioning
3.5.2.3 Boundary Value Analysis/Boundary Value Testing
3.5.2.4 Decision Table-based
3.5.2.5 Cause-effect Graph-based
3.5.2.6 State Transition
3.5.2.7 Random Testing
3.5.2.8 Model-based
3.5.2.9 Factor Analysis Techniques [Fujitsu]
3.5.3 Code-based
3.5.3.1 White-Box Techniques
3.5.3.2 Control Flow-based
3.5.3.3 Data Flow-based
3.5.3.4 Transaction Flow-based
3.5.3.5 Call Flow-based
3.5.4 Fault-based
3.5.4.1 Error Guessing
3.5.4.2 Mutation Testing
3.5.5 Usage-based
3.5.5.1 Operational Profile-based
3.5.5.2 Localization
3.5.5.3 User Environment Simulation
3.5.5.4 Compatibility Verification
3.5.6 Based on nature of software
3.5.6.1 Object-Oriented Testing
3.5.6.2 Web-based Testing
3.5.6.3 GUI Testing
3.5.6.4 Server-side Testing
3.5.6.5 Database Testing
3.5.6.6 Testing of concurrent programs
3.5.6.7 Testing of protocol conformance
3.5.6.8 Testing of real-time systems
3.5.6.9 Testing of safety-critical systems
3.5.7 Combination Techniques
3.5.7.1 Orthogonal Layout Testing
3.5.7.2 All-pairs testing
3.5.8 Risk-based Techniques
3.5.8.1 Risk-based Testing (Test Management)
3.5.8.2 Risk-based Testing (Test Design)
3.5.9 Selecting and Combining Techniques
3.5.9.1 Combining Functional and Structural Ways
3.5.9.2 Combining Deterministic and Random Ways
3.5.10 Test Automation
3.6 Quality Analysis & Evaluation Techniques
3.6.1 Reliability Prediction
3.6.1.1 Software Reliability Model
3.6.1.2 Software Reliability Growth Model
3.6.1.3 Probe [Hitachi]
3.6.1.4 Fault Removal model
3.6.1.5 Fault-Prone Analysis
3.6.2 Quality Progress Management
3.6.2.1 Outlier Management
3.6.2.2 Manpower Result Model
3.6.2.3 Code Integration Pattern
3.6.2.4 PTR Sub-model
3.6.2.5 PTR Generation & Backlog Prediction
3.6.2.6 Rayleigh Model
3.6.3 Fault Analysis
3.6.3.1 ODC (Orthogonal Defect Classification)
3.6.3.2 Bug Analysis
3.6.4 Data Analysis and Representation
3.6.4.1 New QC Seven Tools
3.6.4.2 QC Seven Tools
3.6.4.3 Multivariate Analysis Technique
3.6.4.4 EDA (Exploratory Data Analysis)
3.6.4.5 c Control Chart & u Control Chart
3.6.4.6 Quality Coordinate
3.6.4.7 Radar Chart (Spider Chart)
3.7 Operations & Maintenances Techniques
3.7.1.1 Program Understanding
3.7.1.2 Re-engineering
3.7.1.3 Reverse Engineering
3.7.1.4 Code Clone Analysis
3.7.1.5 Software Rejuvenation
(*1) These knowledge areas are not included in SQuBOK Ver1.0.

Figure 4. “Software Quality Methods” Category (all layers, from category level to topic level)

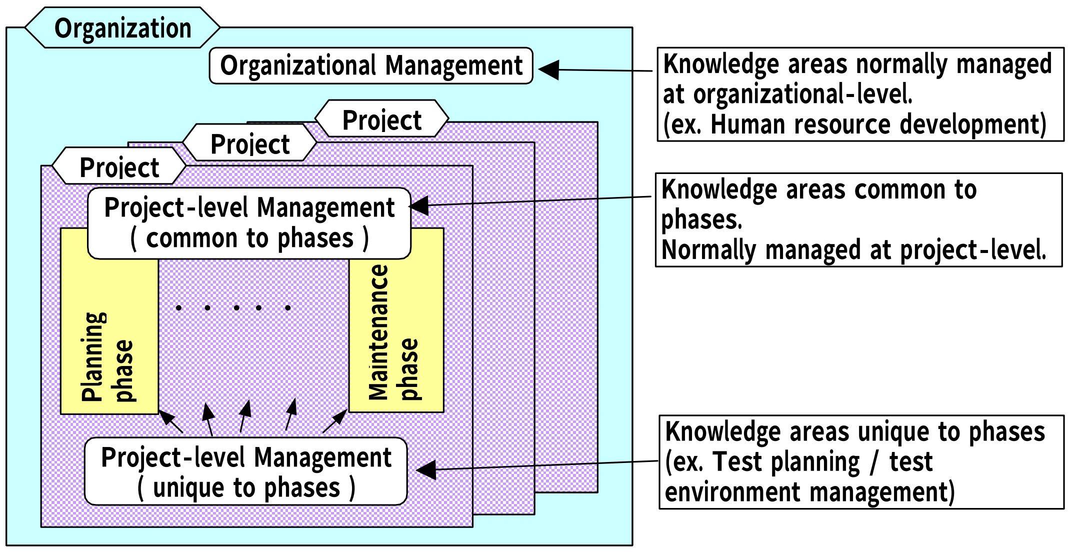
Organization
Organizational Management
Knowledge areas normally managed at organizational-level. (ex. Human resource development)
Project
Project
Project
Project-level Management (common to phases)
Knowledge areas common to phases. Normally managed at project-level.
Planning phase
. . . . .
Maintenance phase
Project-level Management (unique to phases)
Knowledge areas unique to phases (ex. Test planning / test environment management)

Figure 5. “Software Quality Management” Category Classification Method

……

Chapter 1: Fundamental Concept of Software Quality

The “Fundamental Concept of Software Quality” category describes fundamental concepts and approaches concerning software quality. This category is divided into the “Quality Concept” and “Quality Management” knowledge areas.

The “Quality Concept” knowledge area consists of six knowledge sub-areas: “Definition of Quality Concept (History)” exploring the development of the concept by various standards and researchers; “Software Quality Model” as typified by ISO/IEC 9126; “Dependability” describing the broad concept of reliability; and “Security” “Usability” and “Safety” areas that are attracting a new level of interest recently.

The “Quality Management” knowledge area consists of eight knowledge sub-areas: “Quality Control”, “Concept of Quality Assurance”, “Concept of Improvement” “Characteristics of Software Quality Management”, “Concept of Software Measurement”, “Concept of Software Assessment”, “V&V (Verification & Validation)”, and “‘*Kensa*’ (Audit/Checkout)”.

As described above, this category is intended to provide a description of the basic philosophy of software quality; explanations of specific approaches are left to subsequent categories (Chapter 2 and Chapter 3). For example, this chapter focuses on describing the general approach in the “Concept of Improvement” knowledge sub-area of this category’s “Quality Management” knowledge area. Readers wishing to explore more specific methods should consult the topics under “Process Assessment and Process Improvement Management” under the “Software Quality Management” category addressed in Chapter 2.

1.1 KA: Quality Concept

(1) Definition of Quality

Quality and software quality have been defined in various ways by researchers and ISO, JIS, and IEEE standards. Currently, general international agreement has been reached on user satisfaction as the ultimate goal. Here Japan's "consumer-centric" approach as espoused by Kaoru Ishikawa and others has been influential in Europe and the United States. The topics of this knowledge sub-area introduce well-known definitions, excluding deprecated standards.

In this edition, instances such as the following have not yet been organized as topics. Garvin attempted to approach the quality of a given item (not limited to software) from the perspective of interested parties as follows (Garvin 1984):

- Transcendent perspective: Quality can be recognized but is difficult to define.
- User perspective: Does quality comply with the purpose?
- Manufacturer perspective: Does quality conform to the specifications?
- Product perspective: Does quality lead to unique product characteristics?
- Value-based perspective: Quality depends on the amount the customer will pay for value.

Using televisions and table clocks as subject matter, Kano proposed the categories of "expected quality", "one-dimensional quality", and "appealing quality" from the user's perspective using the two-dimensional concepts of satisfaction and material fulfillment (Kano 1984, Kano 1985).

- Expected quality elements: Quality elements the fulfillment of which is expected as a matter of course and the inadequacy of which causes dissatisfaction
- One-dimensional quality elements: Quality elements that cause satisfaction if fulfilled but dissatisfaction if inadequate
- Appealing quality elements: Quality elements that cause satisfaction if fulfilled but to the inadequacy of which the user reacts with a sense of resignation

Kaoru Ishikawa has proposed shortening the Japanese term for quality from "*hinshitsu*" (product quality) to simply "*shitsu*" (quality), a recommendation that is echoed by Kano and Iizuka (Kano 2000, Iizuka 2005). In an era when quality management was chiefly applied to tangible manufactured products, the term *hinshitsu* with its connotation of “product quality” was not problematic and was used without any sense of contradiction. In consideration of the facts that the number of service and other industries providing intangible value is increasing and that expressing the concept of quality as *hinshitsu* evokes the image of the quality of tangible products, these authors are recommending that the Japanese term “*shitsu”* (quality) be used to denote exactly the same meaning as the conventional expression *hinshitsu*, succinctly expressing the intended meaning in all industries. They are interested in the characteristics and features of needs in all areas, whether for products, services, systems, people, processes, or operations. *JIS Q 9005: 2005 Quality Management Systems: Guidelines for Sustainable Growth* (JIS Q 9005: 2005), which was developed by a committee chaired by Iizuka, was the first Japanese standard to use the term “*shitsu*” for quality management.

(2) Difficulty of satisfying users

Although quality is pursued in order to satisfy users, it is important to note that the quality desired by users is not universal but rather subject to constant change. For example, it was important for users of computer programs in the 1950s that those programs operate properly, but subsequently the focus began to shift to reliability and processing time. By the 1980s, reliability was taken as an expected requirement for quality, and attention was shifting to usability. Today, security is one area attracting significant user interest. Gradual, multi-year trends like these are augmented by extremely short-term changes in user needs and satisfaction standards (even while a given software product is being developed).

Additionally, it should be noted that user requirements and expectations are growing increasingly diverse, and the importance given to each quality characteristic varies with individual users (and stakeholders). In some cases, what is desirable to one user is not so to another. Glass, as described in one topic, explains that the priority given to individual quality characteristics should vary with the type of project, and Weinberg also discusses the difficulty of assessing user requirements.

ISO/IEC 9126-1 divides the difficulty of assessing user requirements into the following four areas: “(1) a user is often not aware of his real needs, (2) needs may change after they are stated, (3) different users may have different operating environments, and (4) it may be impossible to consult all the possible types of user, particularly for off-the-shelf software" (ISO/IEC 9126-1: 2001). "Quality during use" was added to the previous "internal quality" and "external quality" categories at the time of the standard's revision in 2001 in an effort to create a software quality model that better reflected user satisfaction.

……

1.2 KA: Quality Management

This knowledge area organizes and introduces approaches to quality management.

First, we will review the necessity and importance of quality management. Organizations are established and conduct activities to provide value to customers. In order to ensure that organizations can enjoy a stable existence over the long term, it is necessary to provide the products and services that are the chief output of the organization’s activities to customers, receive compensation in return, and reinvest the resulting profits to maintain the repeating production cycle of providing value. To accomplish this, the products and services provided by the organization must satisfy a wide range of customers over the long term. Quality management is a tool for accomplishing this goal. Quality management refers to the process of directing and managing organizations to supply products and services of good quality. Quality management is essential if an organization is to enjoy a stable existence over the long term. The fundamental philosophy of quality management consists of a customer-centric approach that seeks to provide satisfaction to customers on an ongoing basis.

Next, we will review the definition of quality management. In ISO 9000 (ISO 9000:2005), quality management is defined as “Coordinated activities to direct and control an organization with regard to quality” and consists of the four activities of quality planning, quality control, quality assurance, and quality “*kaizen*” (improvement). As is made clear by this definition, ISO 9000 makes a series of careful distinctions among quality management, quality control, and quality assurance. Quality management is a comprehensive term, while quality control and quality assurance are subordinate concepts. Quality planning involves the development of plans including standards and other means to satisfy quality requirements. Quality control consists of checking products against relevant standards to ensure compliance in accordance with those plans. By contrast, quality assurance refers to the evidence-based expression of the status of activities for checking quality. Finally, quality “*kaizen*” involves making improvements to products and processes throughout this series of activities.

Here we will review the Japanese approach to quality management. Modern Japanese quality management began with the aid of instruction from the United States after the end of World War II. Dr. W. Edwards Deming and Dr. Joseph M. Juran came to Japan to lecture in 1950 and 1954, respectively, and worked to popularize the practice of quality management through education. As the concept contributed significantly over time to Japanese industry, it underwent a significant development from TQC to TQM and from management focusing on production phases to management including a consideration of administration-level issues. At the same time, its application expanded from the manufacturing industry to the construction, power, service, and software industries, among others. The important philosophies that inform these activities—for example, reliance on "*Genchi Genbutsu*" (actual location and actual materials), small group activities, the participation of all employees, and the revitalization of the organization—crystallize the Japanese approach to quality management. It is here that the roots of the Japanese approach, consisting of "*Genba*"(site)-centered "*kaizen*" accomplished through the participation of all employees as opposed to the commitment-driven approach prevalent in the United States and Europe, are to be found (Iizuka 2005).

Through the process of this development, the focus of the approach taken to the implementation of these philosophies shifted from product inspections to process control and then to new product development (Ishikawa 1981). The inspection approach consisted of selecting products that meet standards by means of product inspections so that defective products are kept from entering the market. As this approach was being pursued, it became evident that selecting defective products is inefficient, and engineers began working to build standard-compliant products from the production process. This change marked the adoption of the process control approach, which sought to build in quality during the production phase. As this approach developed, companies began working to manufacture quality products from the design stage, giving birth to the new product development approach. The new product development approach seeks to build in quality through both product design and production processes. The transformation in these implementation approaches also influenced practices in Europe and the United States.

Quality management philosophies can be analyzed in terms of a propensity to focus on results and a propensity to focus on causes, with result-focused quality management epitomized by the inspection-centric approach that developed chiefly in Europe and the United States. This method, which is based on the imperative of preventing products of poor quality from entering the market by strengthening inspections of the results (products and services), seeks to clearly articulate the evaluation standards on which inspections are based and to avoid supplying products that do not meet standards to the market. By contrast, cause-focused quality management developed primarily in Japan as described above to emphasize the processes that create products and services. By identifying and then eliminating the reasons that poor-quality products are built, the process of creating processes to build high-quality products from the beginning assumes primary importance. To accomplish this goal, the conditions under which processes are performed are measured and analyzed, and processes are subject to “kaizen” improvement. Cause-focused quality management that emphasizes processes is gaining adherents in Europe and the United States as well, as exemplified by the CMMI and Six Sigma strategies. In Japan, a management system that gives a central role to quality in management strategy has been proposed as a further refinement of this approach in the form of *JIS Q 9005: 2005 Quality Management Systems– Guidelines for Sustainable Growth* (JIS Q 9005: 2005).

Next, we will examine quality management as it applies to software. Software has several characteristics that differentiate it from hardware. The most important of these include the fact that software is difficult to understand, including its development process; that it consists of accumulated logic; and that it does not have production processes in the manner of hardware. For these reasons, it is impossible to apply the production process-based quality management techniques used for hardware as-is to software. More so than the hardware production phase, the hardware design phase resembles the process of software development. Additionally, growth in the scale of software, as seen to a remarkable degree in the embedded software domain in recent years, has a significant effect on quality. Because software is an accumulation of logic, it becomes more difficult to obtain a clear view of its content as its scale increases, and developers lose their grasp of that content. At the same time, closeness of communication decreases and it becomes more difficult to facilitate teamwork due to increases in the number of project developers. In this way, one characteristic of software development is the extremely significant influence of the human element. Software quality management requires an understanding of these characteristics of software.

In this knowledge area, the basic concept of software quality management described above is augmented by discussions of software measurement, software evaluation, V&V, and "*kensa*" (audit/checkout).

References

(ISO 9000: 2005)
ISO 9000: 2005, *Quality Management Systems – Fundamentals and Vocabulary.*

(Iizuka 2005)
Yoshinori Iizuka, *Super-ISO Corporate Practice Series Introduction: Moving Beyond ISO* (Japanese version only), Japanese Standards Association, 2005.

(Ishikawa 1981)
Kaoru Ishikawa, *Japanese Quality Control: What is TQC?* (Japanese version), JUSE Press, 1981. (*What is Total Quality Control?: The Japanese Way*, Prentice Hall, 1985)

(JIS Q 9005: 2005)
JIS Q 9005: 2005, *Quality Management Systems – Guidelines for Sustainable Growth.*

Further Readings

(SPC International Task Force 2003)
Software Production Control Board SPC International Task Force, *Characteristics and Background of Japanese Software Quality Control* (Japanese version only), 22nd Software Production Quality Control Symposium/Union of Japanese Scientists and Engineers, pp. 427 to 450, 2003.

……

Chapter 2: Software Quality Management

The “Software Quality Management” category organizes activities for managing quality. Quality should be pursued through systematic activities extending through all layers of an organization, from the management level to the production floor, and there is a diverse range of activities related to quality management. This chapter organizes these activities by separating them into those that apply to organizations (companies or departments), which include multiple software development projects, and those that apply to individual projects, which are further divided into the following three sub-categories in relation to individual phases in the life cycle:

- Activities that are common throughout the organization, or that apply to the entire organization
- Activities that are undertaken in the context of projects, performed throughout the life cycle, and do not vary by phase
- Activities that are undertaken in the context of projects and vary by phase

Software Quality Management at Organizational Level

This sub-category organizes activities that are common throughout the organization, or that apply to the entire organization. It discusses the following knowledge areas:

- Development and Operation of Software Quality Management System (Section 2.1)
- Life Cycle Process Management (Section 2.2)
- Process Assessment and Process Improvement Management (Section 2.3)
- Inspection Management (Section 2.4)
- Audit Management (Section 2.5)
- Human Resource Cultivation Management (Section 2.6)
- Legal Right and Responsibility Management (Section 2.7)

2.1 KA: Development and Operation of Software Quality Management System

This knowledge area discusses the specific methods by which quality management in the software domain is implemented in organizations.

(1) Quality management systems and the "*kaizen*" approach

ISO 9000 defines a quality management system as a "Management system to direct and control an organization with regard to quality" for products and services, the primary output of organizations (ISO 9000:2005). ISO 9001 (ISO 9001:2000), which defines the requirements that quality management systems must fulfill, recommends the adoption of a process approach for improving the effectiveness of such systems. The term "process approach" refers to the explicit and systematic expression of the processes used inside the organization, with particular emphasis on the interactions between processes, as well as their operational management. The application of a process approach to quality management systems enables both the system's constituent elements to be organized using the flow of "inputs → conversion activities → outputs" and the influence among those elements to be clearly articulated. This approach makes it easy to implement a cycle by which processes vital to the effectiveness of the system are identified and systematically improved in order to facilitate the improvement of the quality management system on an ongoing basis.

In Japan, activities typified by reliance on "*Genchi Genbutsu*" (actual location and actual materials), small group activities, the participation of all employees, and the revitalization of the organization has been practiced as part of TQC/TQM by the hardware manufacturing industries. These activities comprise production floor-centric "*kaizen*" activities based on the participation of all employees, and they represent an approach that differs significantly from activities that focus on the process definition described above. Quality management knowledge from these hardware-manufacturing industries has been incorporated into the software domain, leading to the current situation. In "*Genba*"(site)-centered "*kaizen*" activities based on the participation of all employees, all production floor workers embrace the quality goals and work together to achieve them. This approach has the advantages of enabling "*kaizen*" activities to reach all areas of quality management thanks to its production floor focus and of enabling an organization to be built by employees thinking and acting for themselves.

"*Kaizen*" activities that focus on the process definition and production floor-centric "*kaizen*" activities based on the participation of all employees share a common orientation in that they both seek to improve the effectiveness of quality management systems. It is important to understand the advantages of both approaches and to combine and implement them according to the characteristics of the organization in question.

(2) Software quality management system characteristics and correspondence

Software quality management systems are quality management systems in the software domain. This section discusses noteworthy aspects of quality management systems, particularly in the software domain.

Software is characterized by intellectual difficulty that extends to the development process, a preponderance of logic, and the presence of strong human elements such as teamwork that exert an influence on software quality. Accordingly, it is of key importance to develop software quality management systems with mechanisms that make it possible for workers to understand difficult phenomena and organize large amounts of logic from a technical perspective for practical use while also improving teamwork and motivation on the part of developers. Chapter 2, "Software Quality Management" and Chapter 3, "Software Quality Methods," discuss the constituent methods for implementing such systems.

Let us consider the requirements for an organization wishing to develop and operate a software quality management system. The organization must have responsibility for quality. There are two types of responsibility for quality: responsibility for products, and responsibility for the processes that give rise to products (including not only procedures but also all elements necessary to undertake the series of associated activities, including tools, technologies, and developers). The field of software requires an organizational format that is explicitly aware of these dual areas of responsibility. If the focus is exclusively on responsibility for product quality, it is difficult to create and take advantage of opportunities for introducing revolutionary technologies, for example those that can dramatically change processes. On the other hand, limiting the focus to responsibility for processes raises the possibility that problems may be overlooked due to the lack of direct confirmation of product quality. Put simply, a focus on "*kensa*" (audit/checkout) doesn't ensure that high-quality products will be manufactured in the first place, and a focus on "*kaizen*" doesn't provide a means for judging whether there are issues with the final deliverables. It does not matter whether responsibility for products and processes is vested in a single organization or in separate organizations. The important thing is that the software quality management system should be advanced by an organization motivated by having responsibility for both of these aspects of quality.

Additionally, it is important to provide mechanisms for improving the quality management system on an organizational and ongoing basis. In the software domain, it is both easy and effective to use the occurrence of faults as an opportunity to undertake a cycle of "*kaizen*". By analyzing the true cause of the fault, the process is improved so that the fault is not "built in" to the product again. Repeating this "*kaizen*" process both steadily improves process effectiveness and makes process "*kaizen*" part of the organizational culture. Establishing the objective and scope of "*kaizen*" efforts according to the sophistication of the quality management system is another important consideration in accomplishing effective "*kaizen*". This approach is the same as that asserted by CMM/CMMI, i.e., that maturity levels are defined as stages in the evolution of process "*kaizen*" and should not be skipped. Efforts by organizations whose quality management systems operate at a low level of maturity to introduce advanced technologies tend not to be successful. When a quality problem occurs while basic measures have not been implemented, the organization must first implement "*kaizen*" to ensure that those basics are reliably implemented.

Software quality management systems should be built and operated based on a consideration of the characteristics of software as described above.

In addition to basic concepts in developing and operating software quality management systems as described above, this knowledge area discusses quality management systems, software quality promotion, and approaches for creating quality management organizations.

References

(ISO 9000: 2005)

ISO 9000: 2005, *Quality Management Systems – Fundamentals and Vocabulary*.

(ISO 9001: 2000)

ISO 9001: 2000, *Quality Management Systems – Requirements*.

Further Readings

Software Production Control Board SPC International Task Force, *Characteristics and Background of Japanese Software Quality Control* (Japanese version only), 22nd Software Production Quality Control Symposium/Union of Japanese Scientists and Engineers, pp. 427 to 450, 2003.

2.1.1 S-KA: Quality Management System

Understanding quality management systems requires an awareness of the differences between the ISO9000 approach and the approach that evolved through the development of TQC/TQM in Japan.

In ISO 9000, quality management systems are defined as a "Management system to direct and control an organization with regard to quality" for the products and services, the primary output of organizations (ISO 9000: 2005). Quality management systems consist of the four activities of quality planning, quality control, quality assurance, and quality "*kaizen*" pursued in response to quality objectives set in accordance with a quality policy given direction by top management. The ISO 9000 standard spurred the penetration of the quality management system approach, which spread rapidly worldwide along with the registration system. However, it is not sufficient to develop a quality management system as defined by ISO 9000. Such systems seek customer satisfaction, but the quality requirements that should be fulfilled are limited chiefly to criteria to which customers have agreed, and little attention is paid to latent customer requirements.

Moreover, the ISO 9000 approach focuses on "*kaizen*" from the standpoint of quality management system effectiveness but does little to address "*kaizen*" from the perspective of improving the products and services themselves or exploring how they might be created most efficiently. This orientation is due in part to natural limits resulting from the need to employ evidence-based confirmation with explicit standards founded on the assumption of third-party examination through the registration system. In this way, ISO 9000 seeks to define a broad quality management system but ends up with a system with a limited scope.

By contrast, the approach to quality management that evolved through the development of TQC/TQM in Japan consists of "all activities associated with providing products that customers can use with confidence" (Iizuka 2005) and is founded on a meticulous pursuit of customer satisfaction and a program of quality-centric "*kaizen*" accomplished with the participation of all employees. The definitive difference with the European and American approach as represented by ISO 9000 lies in the distinctive

Japanese approach of focusing on putting the system in motion, even if it is inadequate, and having all employees strive to improve processes themselves through a program of "*kaizen*" as compared to the European and American approach of defining processes and then verifying whether they are being executed in accordance with their definitions. Overall, the Japanese approach functions as a system while using tools such as QC circles and statistical methods. The advantage of this Japanese-style management system lies in its ability to generate efficient, waste-free organizational operation by enabling all employees to work toward the same objective. *JIS Q 9005: 2005 Quality Management Systems – Guidelines for Sustainable Growth* (JIS Q 9005: 2005) proposed a further refinement of this Japanese approach in the form of a management system that places quality at the core of its management strategy.

For an organization to enjoy a stable existence over the long term, the products and services it provides must satisfy a wide range of customers over the long term. Quality management systems are an important tool in achieving that goal. It is desirable to pursue a program of "*kaizen*" that reflects the characteristics of the organization based on the understanding of the scope and limits of the ISO 9000 quality management systems and the understanding of the advantages of Japanese-style quality management systems.

References

(ISO 9001: 2000)
ISO 9001: 2000, *Quality Management Systems – Requirements.*

(ISO 9000: 2005)
ISO 9000: 2005, *Quality Management Systems – Fundamentals and Vocabulary.*

(JIS Q 9005: 2005)
JIS Q 9005: 2005, *Quality Management Systems – Guidelines for Sustainable Growth.*

(Iizuka 2005)
Yoshinori Iizuka, *Super-ISO Corporate Practice Series Introduction: Moving Beyond ISO* (Japanese version only), Japanese Standards Association, p.35, 2005.

Further Readings

Kaoru Ishikawa, *Japanese Quality Control: What is TQC?* (Japanese version), JUSE Press, 1981. (*What is Total Quality Control?: The Japanese Way*, Prentice Hall, 1985)

Yoshinori Iizuka and Jun'ichi Jido, *Super-ISO Corporate Practice Series: The TQM Philosophy* (Japanese version only), Japanese Standards Association, 2005.

……

2.1.1.4 T: JIS Q 9005 Quality Management Systems -- Guidelines for Sustainable Growth

JIS Q 9005 (JIS Q 9005: 2005) moves beyond the quality management system model provided by the ISO 9000 family by providing a quality management system model designed to enable organizations to adapt agilely to changes in their environments. Moving beyond the ISO 9000 family refers to approaching quality management systems from the standpoint of management and proposing an approach with self-initiated innovation that can be used to respond to changes in a variety of business environments. In this way, the approach seeks to enable long-lived, sustainable organizations. The standard underwent the JIS standardization process with the purpose of spreading its content widely as quality-oriented Japan.

The standard posits that organizations must have a capability for innovation and learning to enable them to deal with change if they are to successfully address changes in their environments and achieve sustainable growth. Additionally, it proposes the articulation of a specific vision of organizational capability as well as a three-level quality management model (consisting of a quality management system innovation level, a quality management system continual “*kaizen*” level, and a product and service continual “kaizen” level) in order to implement the organization’s business strategy. Furthermore, it defines 12 principles of quality management for implementing this approach. JIS Q 9006 provides guidelines for the self-assessment of quality management systems in an effort to facilitate convenient self-assessment by organizations implementing this approach (JIS Q 9006: 2005).

Objective
To enable companies to establish quality management systems capable of continuing to grow in all business environments.

Method
To use this standard as a benchmark for quality management systems.

Results
To enable companies to continue to create a high level of customer value, secure competitive advantage, and enjoy sustainable growth.

Key Points

This standard makes a point of using the Japanese term “*shitsu”* instead of “*hinshitsu”* (both terms translate as “quality” in English) in defining quality as “Overall characteristics of capability to meet needs or expectations” (JIS Q 9005: 2005). This decision derives from the fact that the term *hinshitsu* is limited to suggesting quality in the sense of “product quality” and reflects a desire on the part of the standard’s authors that there be no psychological barriers to the use of the term by industries that provide intangible value.

Related topics and knowledge areas

Standards related to quality management systems (ISO 9000 family)

References

(JIS Q 9005: 2005)

JIS Q 9005: 2005, *Quality Management Systems – Guidelines for Sustainable Growth*, page 2.

(JIS Q 9006: 2005)

JIS Q 9006: 2005, *Quality Management Systems – Guidelines for Self-assessment*.

Further Readings

Yoshinori Iizuka (editor) and JIS Q 9005/9006 Guidelines Editing Committee (author/editor), *Quality Management Systems for Achieving Sustainable Growth, with JIS Q 9005/9006 Guide Utilization Case Studies* (Japanese version only), Japanese Standards Association, 2006.

……

3.3.1.2 T: QFD (Quality Function Deployment)

Quality function deployment (QFD) is defined as “Methodology employed various types of transformation and deployment in order to realize quality objective (JIS Q 9000) for a product (JIS Q 9000), which may be abbreviated as QFD. INFORMATION: Term referring collectivity to quality deployment, engineering deployment, cost deployment, reliability deployment and job function deployment” (JIS Q 9025: 2003). The term quality function deployment includes the two meanings of the deployment of quality and the deployment of operational functions. Quality deployment refers to work undertaken to articulate the type of quality required by customers and to analyze the circumstances necessary to build that quality into manufactured products. Operational function deployment refers to work undertaken to articulate functions performed to build in the required level of quality. In short, quality function deployment acts as a tool for articulating the quality that is required in the products being manufactured and building operational mechanisms for building the articulated quality into the products.

Objective

Although one of the initial objectives of quality function deployment was to provide a methodology that could be used when developing new products with superior quality compared to competitors’ products, it has subsequently come to be used as a quality assurance mechanism. In short, its application is being expanded so that it acts as a tool for articulating the type of quality required during the planning and design phases, communicating that quality to and reflecting it in the manufacturing phase.

Method

First, market (customer) requirements are identified, converted into expressions of required functions and quality, and organized hierarchically by levels of abstraction. Then the technologies necessary to implement those requirements are identified, expressed in the form of quality elements and characteristic that indicate technical elements, and organized hierarchically by quality levels of abstraction. Next, the required functions and quality are combined with the quality elements and characteristics, and a list of quality is created to express the correspondence between the two. This clarifies the relationship of correspondence between the elements comprising the product and the desired quality and articulates considerations in building in the desired level of quality. This information is then used to determine the plan quality and design quality, articulating the operations that must be performed in order to reliably build the determined quality into the product. In performing these tasks, it is important to address not only quality but also to articulate all action that should be taken in each operation while considering factors such as cost and reliability.

Results

The introduction of quality function deployment can be expected to yield the following benefits:

- An ability to gather and organize requirement information for the market (customers)
- An ability to systematically organize the necessary technical elements in order to incorporate customer requirements
- An ability to provide mechanisms for reliably incorporating the quality required by customers
- An ability to reliably communicate quality elements integrated during the planning and design phases to the manufacturing phase

Key Points

It is a frequent misconception that quality function deployment consists of creating a list of quality. In fact, it is important to address the question of how customer requirements can best be incorporated into products and to communicate the quality elements determined during the planning and design phases to the manufacturing phase to manufacture the product. Rather, it is critical to ask how quality can be deployed through operations in order to build in quality as understood using the list of quality.

Related topics and knowledge areas

Quality Plan Management

References

(JIS Q 9025: 2003)

JIS Q 9025: 2003, *Performance Improvement of Management Systems – Guidelines for Quality Function Deployment*, page 3.

Further Readings

Shigeichi Moriguchi (editor), (Japanese version) *Software Quality Management Guidebook*, Japanese Standards Association, 1990. (English version) *Software Excellence: A Total Quality Management Guide*, Productivity Pr, 1997

Ayatomo Kanno and Tadashi Yoshizawa (supervising editors) and JUSE Software Quality Management Study Group (editor), *Software Quality Assurance Technologies for the 21st Century: Ten Years of Accomplishments from the JUSE Software Quality Management Study Group* (Japanese version only), JUSE Press, 1994.

Katsuyuki Yasuda, *Software Quality Assurance Approach and Practice: A Systematic Approach to the Open Source Era* (Japanese version only), JUSE Press, 1995.

Yoji Akao and Shigeru Mizuno, *Quality Function Deployment: A Companywide Approach to Quality Control* (Japanese version only), JUSE Press, 1978.

Yoji Akao, *Quality Function Deployment Utilization Manual 1: Introduction to Quality Deployment* (Japanese version only), JUSE Press, 1990.

Tadashi Ofuji, Michiteru Ono, and Yoji Akao, *Quality Function Deployment Utilization Manual 2: Quality Deployment Methods (I)* (Japanese version only), JUSE Press, 1994.

Tadashi Ofuji, Michiteru Ono, and Yoji Akao, *Quality Function Deployment Utilization Manual 3: Quality Deployment Methods (II)* (Japanese version only), JUSE Press, 1994.

……

3.5.7.1 T: Orthogonal Array Testing

Orthogonal Array testing is a technique used in designing combination tests for test cases that cover all combinations between two factors rather than covering all combinations by taking advantage of the property that the total number of level combinations is the same between any two factors for orthogonal Array used in the design of experiments method. Design of experiments is a method developed by R.A. Fisher of England in 1920 to rationalize agricultural experiments, based on which Dr. Genichi Taguchi developed a method known as quality engineering. Quality engineering was introduced to the United States in the early 1980s and entered into widespread use, chiefly in the automotive industry, where its techniques were known as Taguchi methods. The application of orthogonal layouts to software testing began around 1984 in Japan (at Fujitsu), with technical development continued primarily by AT&T engineers in the United States during the 1990s.

Objective

To reduce the number of test cases in a rational manner and design combination tests with a viable number of cases.

Method

(1) The parameters (factors) for the function targeted by the test and their respective values (levels) are organized.
(2) An orthogonal layout whose size reflects the number of factors and levels is created.
(3) The factors and levels are allocated to the orthogonal layout.
(4) When there are relationships where factors cannot be combined (prohibitions), these are avoided by changing the shape of the orthogonal layout.
(5) The results of the allocation process are used as test cases.

Results

There are examples of investigation reports indicating that many software faults are caused by either one (single-mode fault) or two (double-mode fault) factors. This technique enables reliable testing to detect faults of this scope.

Key Points

This method does not cover combinations of three or more factors, making it necessary to add test cases reflecting combinations of three or more factors as needed.

Related topics and knowledge areas
All-pairs Testing

References
None

Further Readings

Shinobu Sato and Hiroki Shimokawa, *Methods for Setting Software Test Parameters Using the Design of Experiments Method* (Japanese version only), *Proceedings of the 4th SPC Symposium*, JUSE Press, pp.1-8, 1984.

Madhav S. Phadke, “Planning Efficient Software Tests”, *CrossTalk*, October 1997, http://www.stsc.hill.af.mil/crosstalk/1997/10/planning.asp.

Genichi Taguchi, *Functional Evaluation for Robust Design: Methods for Efficient Development* (Japanese version only), Japanese Standards Association, 2000.

Lee Copeland, *A Practitioner’s Guide to Software Test Design*, STQE Publishing, 2004.

Koichi Akiyama, *Feature 3: Introduction to Combination Testing Using Orthogonal Array* (Japanese version only), Software Test Press/Gijutsu-Hyohron, Vol. 2, pages 89 to 107, 2006.

Masataka Yoshizawa, Koichi Akiyama, and Taro Sengoku, *Introduction to HAYST Software Testing: How to Use Orthogonal Array to Increase Quality and Productivity* (Japanese version only), JUSE Press, 2007.

Mitsuru Oba, *Software Quality Assurance: Techniques for Gathering and Analyzing Software Project Performance Data* (Japanese version only), Soft Research Center, 1993.

Kenji Onishi, *Practical Guide to Software Testing for Increased Reliability* (Japanese version only), Nikkei Business Publications, 2004.

Stephen H. Kan, *Metrics and Models in Software Quality Engineering* (2nd Edition), Addison-Wesley, 2003.

……

3.6.1.3 T: Quality Probe (Hitachi)

Quality Probe (QP) is a technique employed by Hitachi for intermediate quality audit to predict software reliability. Quality assurance department measures and evaluates software quality during the testing stage using sampling tests in advance of the product "*kensa*" (inspection). This approach provides an early assessment of quality and yields guideline for various measures to improve quality.

Objective

To accelerate the removal of defects by evaluating quality during the testing stage and obtain pointers to various measures for improving quality.

Method

(1) Some 10% to 20% of the "*kensa*" (inspection) items designated by the Quality Assurance Departments are sampled for use as QP items.

(2) Quality assurance department performs actual testing using the identified items.

(3) The number of remaining software defects is estimated using statistical methods (binomial probability paper) based on the defects detected during the testing.

Results

(1) Reliability is improved by having design departments detect defects in reference to the number of remaining defects estimated using QP.

(2) Engineers can ascertain defect trends and causes and discover the weaknesses of their software by analyzing and evaluating the nature of the defects detected by QP. Then design departments can strengthen their testing perspective and revitalize the testing process to enable the detection of more defects through testing.

Key Points

(1) The standards governing the process, for example the scale of the software to which QP is applied and the number of QP cycles to be performed, should be articulated.

(2) When performing QP, it is necessary to determine whether the software is at a

proper level of quality. If the technique is not performed after testing has made significant progress (normally, when 90% to 95% of testing is complete) and software quality has reached a certain level, the sampling test will not serve to estimate quality and will serve simply as a normal testing or a debugging process.

Related topics and knowledge areas

Software Reliability Growth Model

References

None

Further Readings

Katsuyuki Yasuda, *Software Quality Assurance Approach and Practice: A Systematic Approach to the Open Source Era* (Japanese version only), JUSE Press, 1995.

Chin-Kuei Cho, *An Introduction to Software Quality Control*, John Wiley & Sons, 1980

……

Questionnaire

1. If "Complete English version of SQuBOK guide" is published, will it be useful and widely accepted in your country or not?　(Y, N)

Why?

2. Please give us your comments or advice to the current topics organization (see tree diagram), description, references, and further readings (see this abridgment) from viewpoint of global deployment.

3. Your Profile

Name:

Country:

Mail:

Field:　Industry　　　　Academic　　　　Other:

Please send your answer/comments to following address:

Mail:　juse@juse.or.jp

Fax:　+81-3-5378-1220