Centro de Informática
U·F·P·E

Introdução ao Guia SWEBOK

Vítor Andrade
vagar@cin.ufpe.br

AGENDA

O SWEBOK (2004)

Guide to the
SoftWare
Engineering
Body of
Knowledge
(SWEBOK)

Patrocinadores:

Editores:

Alain Abran

Université du Québec

James W. Moore

MITRE

Pierre Bourque

Université du Québec

Robert Dupuis

UQÀM

O QUE É ENGENHARIA DE SOFTWARE?

O IEEE define **Engenharia de Software** como:

"(1) a aplicação de uma abordagem sistemática, disciplinada e quantificável de desenvolvimento, operação e manutenção de software; que é a aplicação de engenharia a software

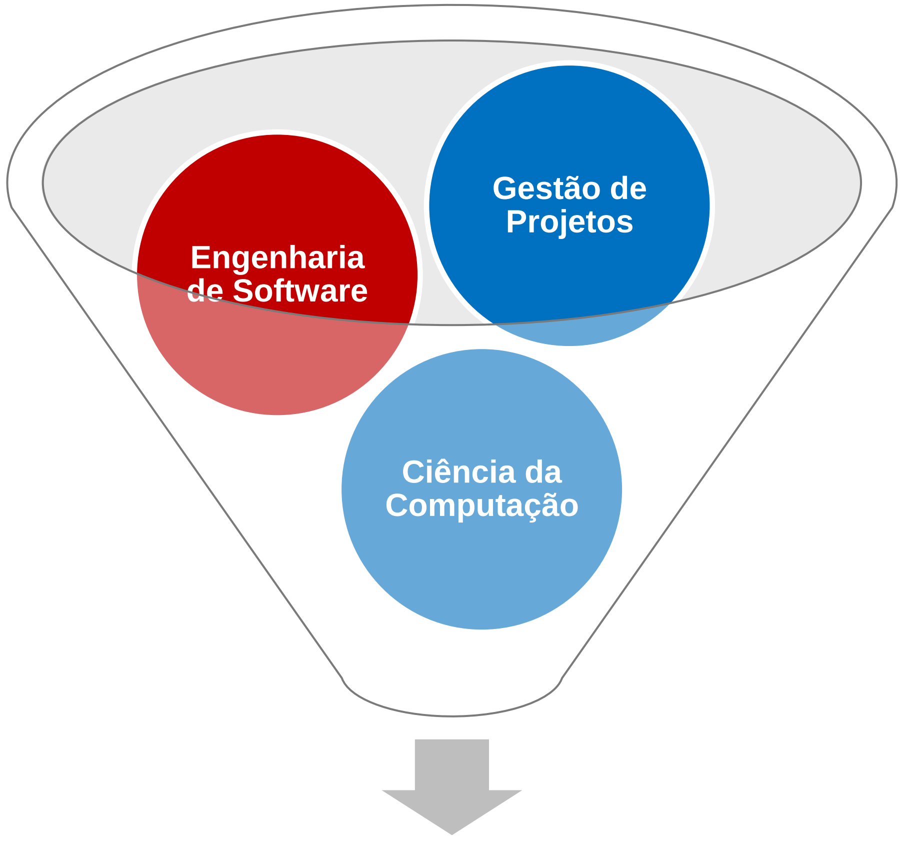
(2) O estudo de abordagens de (1)."

Fonte: SWEBOK, 2004

O SWEBOK

- O Guia cobre o conhecimento de engenharia de software necessário, mas não suficiente ao engenheiro de software.
- **NÃO** foca em assuntos específicos como, por exemplo, linguagens de programação, bancos de dados relacionais e redes não são cobertos no SWEBOK
- E **SIM** no conhecimento essencial que suporte a seleção da tecnologia apropriada, no tempo e na circunstância apropriados.

Exemplo:



Conhecimento Exigido do Engenheiro de Software

Fonte: SWEBOK, 2004

O IEEE

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS

- O IEEE é a maior associação profissional dedicada ao avanço da inovação tecnológica e excelência em benefício da humanidade. O IEEE e seus membros inspiram uma comunidade global por meio de publicações relevantes, conferências, padrões e atividades profissionais e educacionais (IEEE, 2012).

- Website: http://www.ieee.org/index.html?WT.mc_id=hpf_logo

Fonte: IEEE, 2012

OBJETIVOS DO SWEBOK

O "Guide to the Software Engineering Body of Knowledge" foi criado com 5 objetivos:

1 Promover uma visão consistente da engenharia de software mundialmente;

2 Esclarecer o lugar – e definir uma fronteira – da engenharia de software em relação a outras disciplinas.

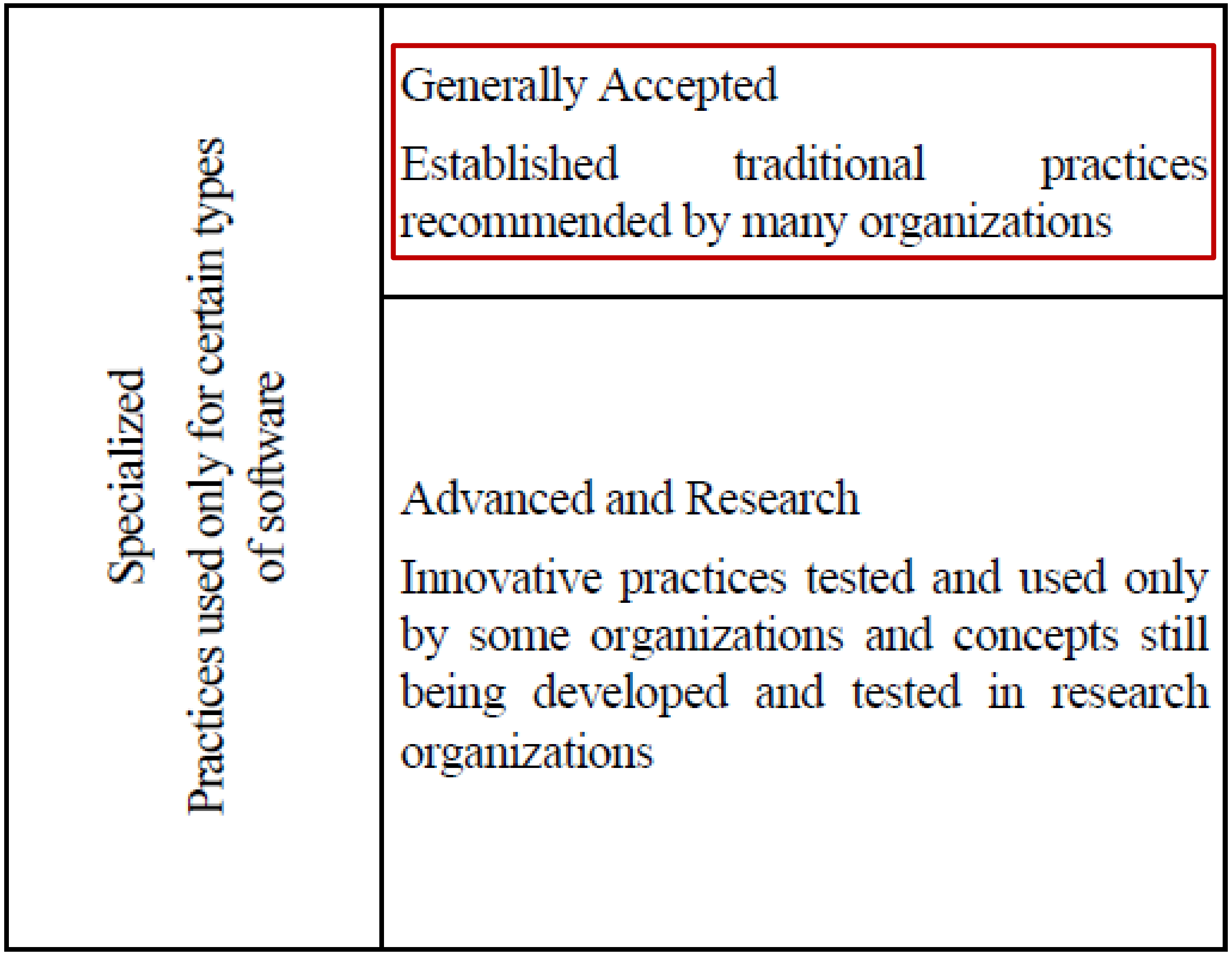
3 Caracterizar os conteúdos da disciplina de engenharia de software;

4 Proporcionar acesso topificado do conjunto de conhecimento na área de Engenharia de Software;

5 Prover uma base para desenvolvimento de um currículo, para certificação de profissionais e licenciamento de materiais;

Fonte: SWEBOK, 2004

FOCO DO SWEBOK



Specialized

Practices used only for certain types of software

Generally Accepted

Established traditional practices recommended by many organizations

Advanced and Research

Innovative practices tested and used only by some organizations and concepts still being developed and tested in research organizations

PÚBLICO-ALVO

- **Organizações públicas e privadas** que necessitavam de uma visão consistente sobre a engenharia de software para definição de requisitos de formação e treinamento, classificar vagas, desenvolver políticas de avaliação de desempenho ou até mesmo especificar atividades de desenvolvimento de software;
- **Engenheiros de software**;
- **Autoridades** responsáveis por elaborar políticas públicas;
- **Sociedades profissionais e educadores** para definição de regras de certificação, políticas de acreditação para currículos acadêmicos e orientações para a prática profissional.
- **Estudantes de engenharia de software**

Fonte: SWEBOK, 2004

O PROJETO SWEBOK – 3 FASES

1998 — **2001** — **2004**

STRAWMAN

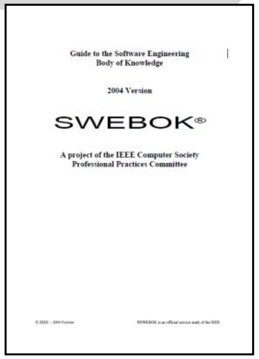
Apresentou um protótipo de como o projeto seria organizado

STONEMAN

Publicação de uma versão *Trial* e início de sua utilização

500 revisores, 42 países

IRONMAN



Guide to the Software Engineering
Body of Knowledge

2004 Version

SWEBOK®

A project of the IEEE Computer Society
Professional Practices Committee

© IEEE – 2004 Version
SWEBOK is an official service mark of the IEEE

120 revisores, 42 países

10 brasileros

Fonte: SWEBOK, 2004

COMPOSIÇÃO DO SWEBOK

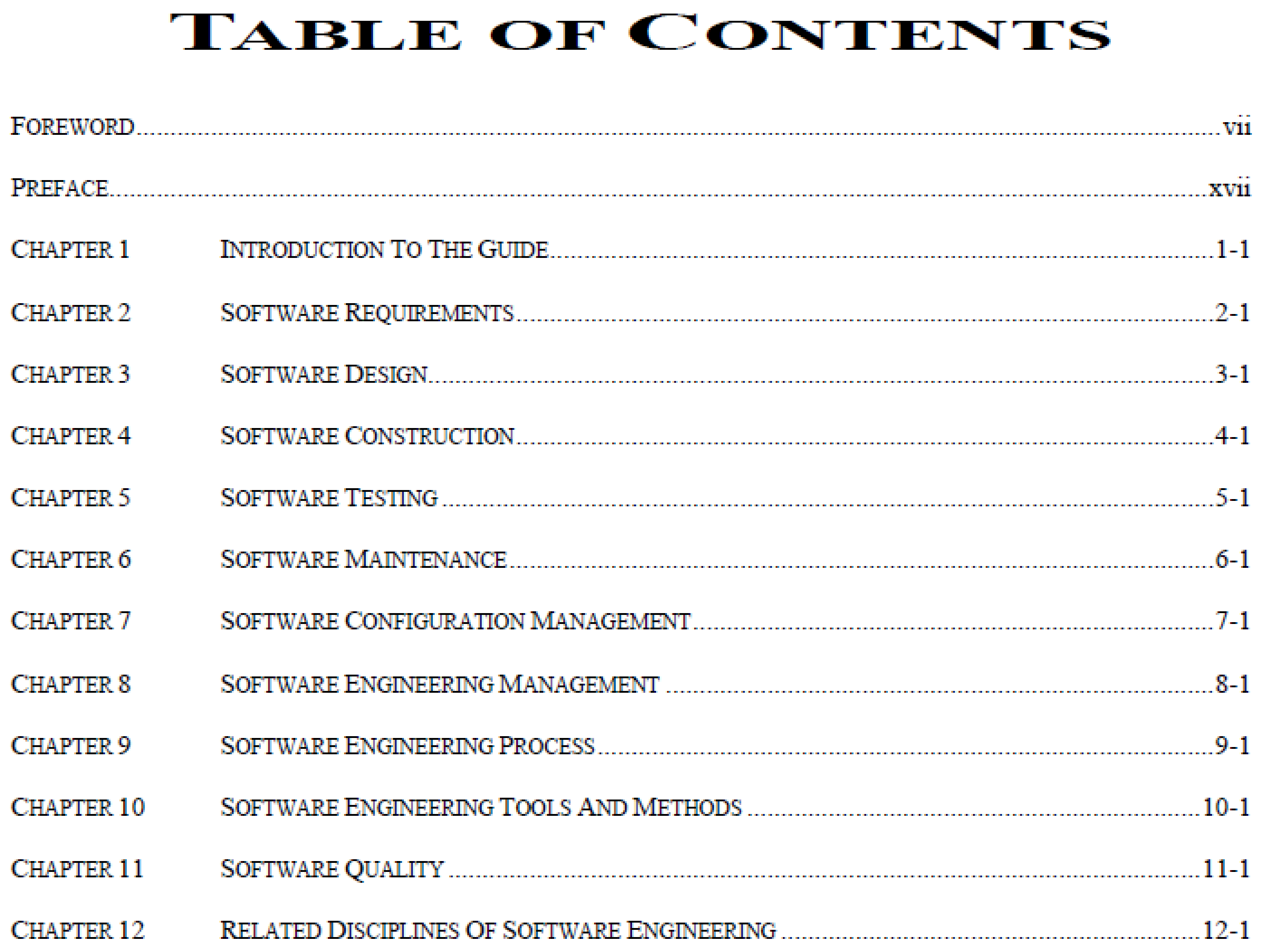


TABLE OF CONTENTS

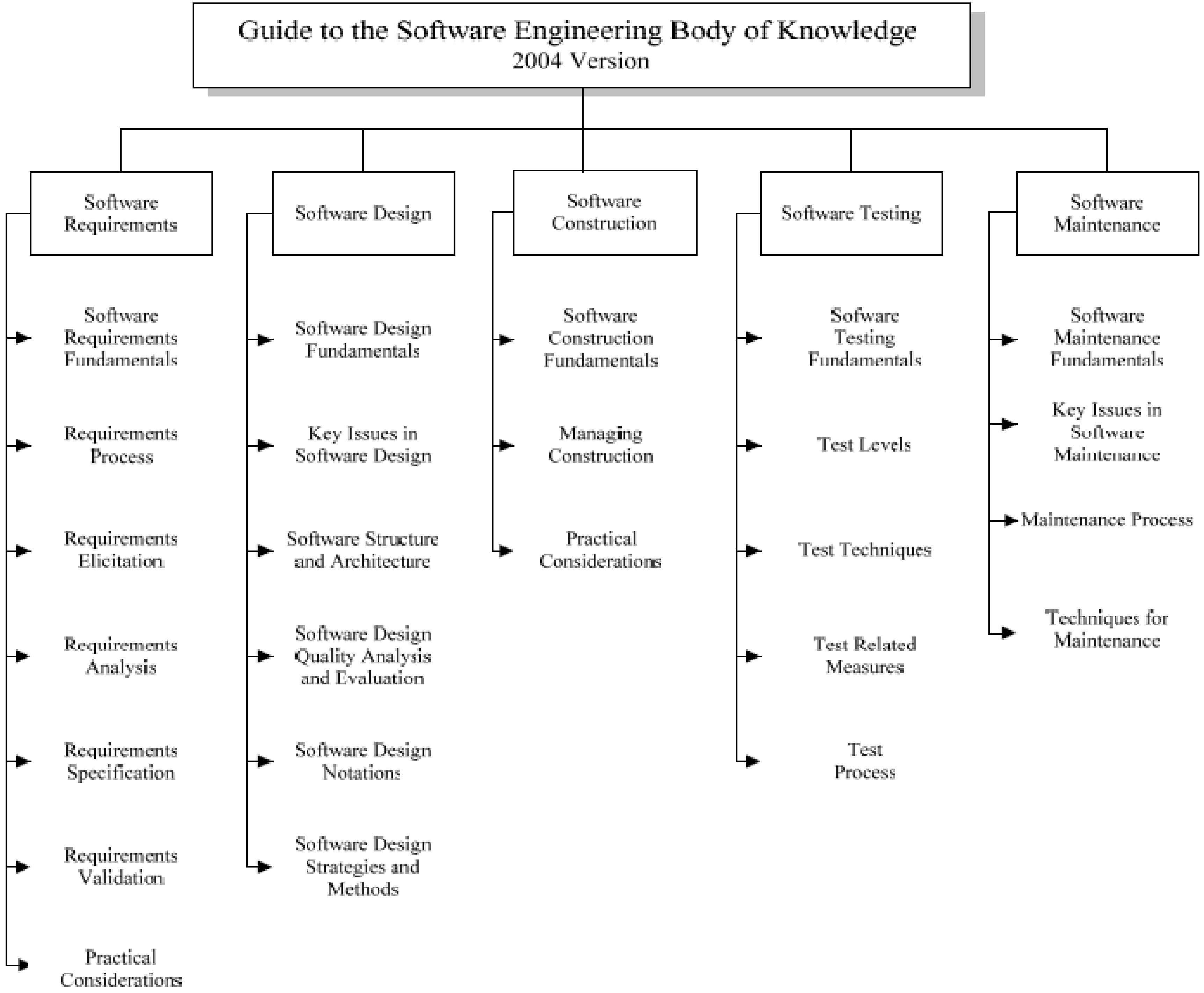
ÁREAS DO CONHECIMENTO

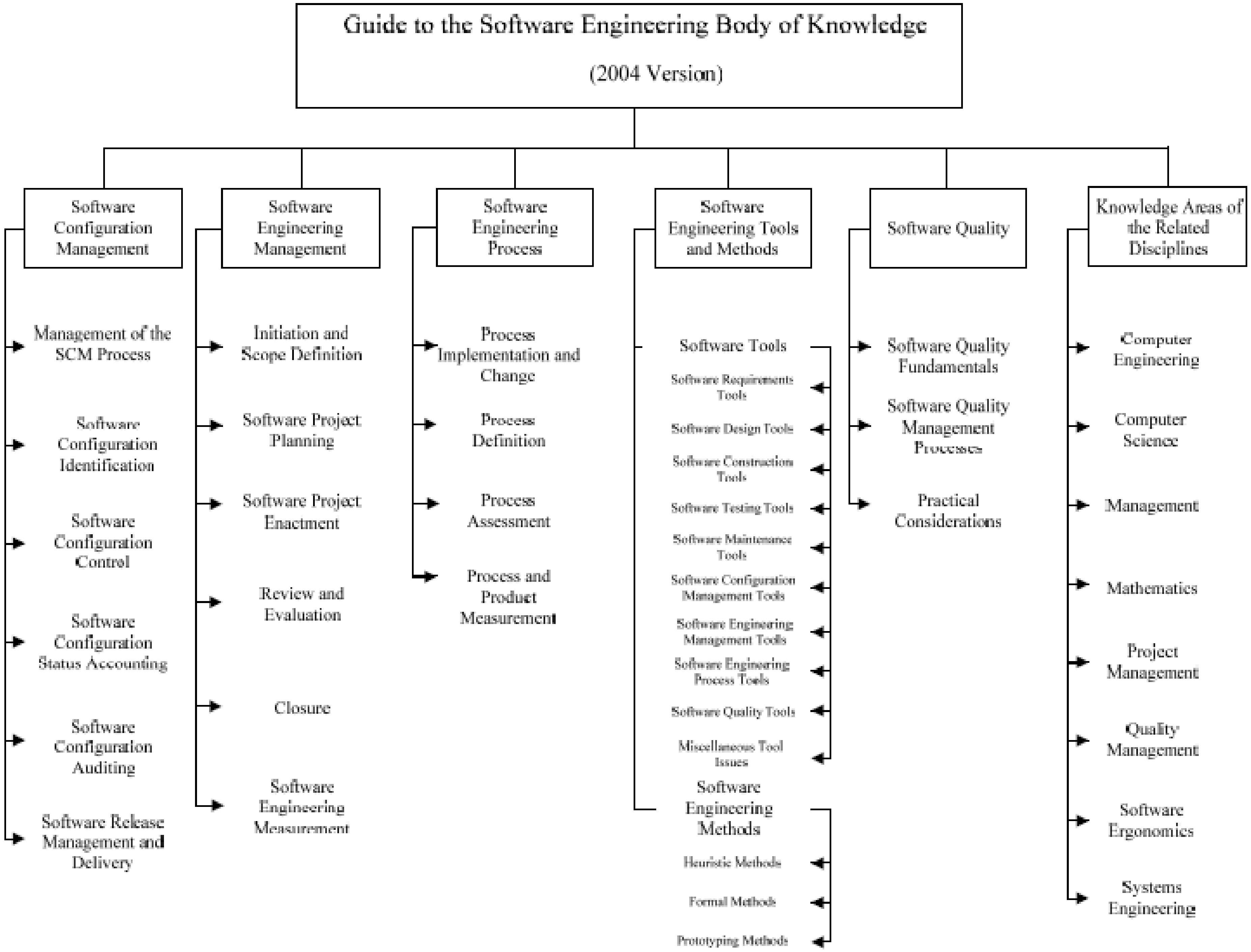
AS ÁREAS DO CONHECIMENTO DO SWEBOK

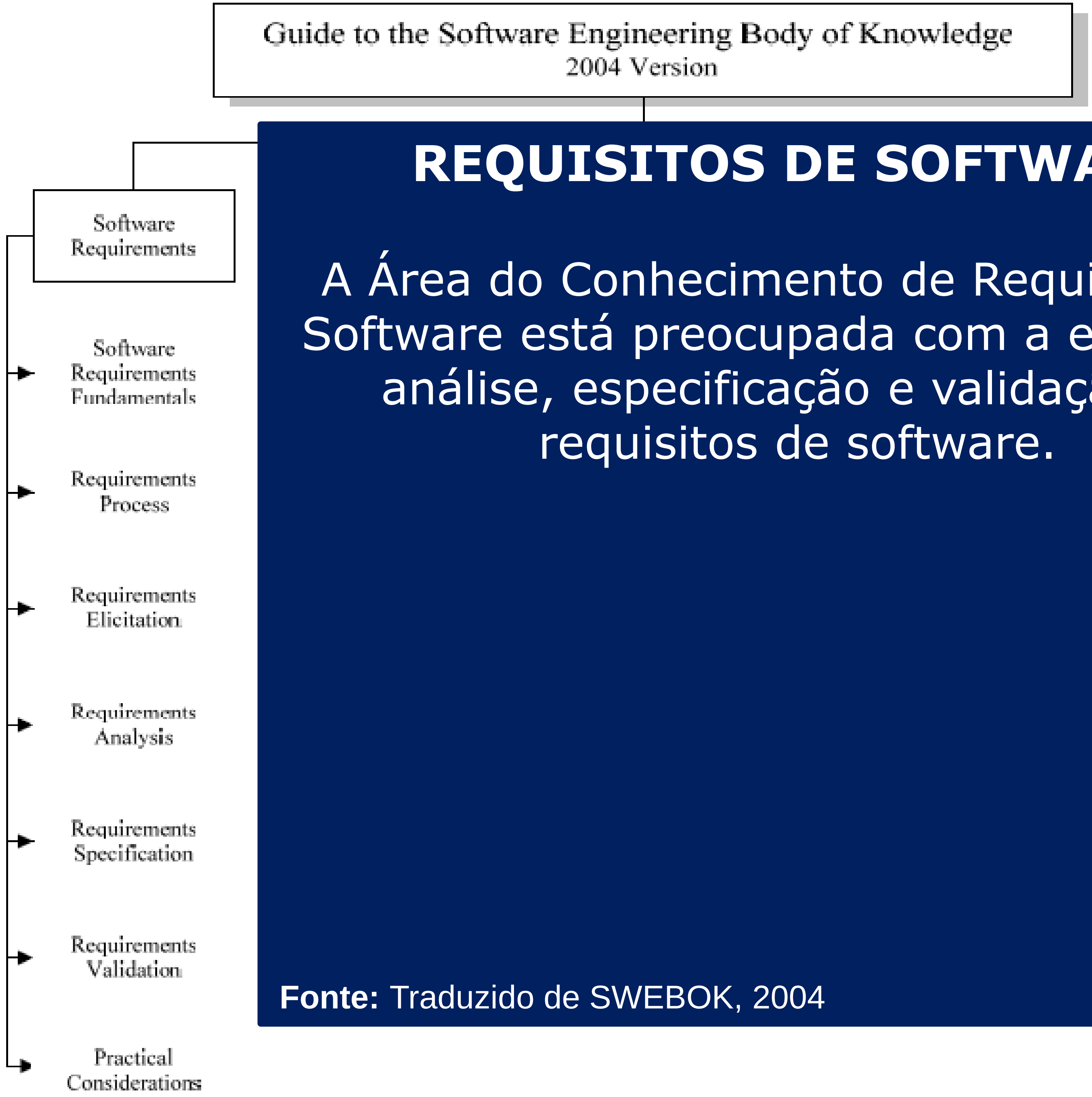
- São 10 as áreas de conhecimento do SWEBOK (knowledge areas – Kas):

Cap. 2	**Requisitos de Software**	**Gerenciamento de Configuração de Software**	**Cap. 7**
Cap. 3	**Projeto de Software**	**Gerenciamento de Engenharia de Software**	**Cap. 8**
Cap. 4	**Construção de Software**	**Processo de Engenharia de Software**	**Cap. 9**
Cap. 5	**Teste de Software**	**Ferramentas e Métodos de Engenharia de Software**	**Cap. 10**
Cap. 6	**Manutenção de Software**	**Qualidade de Software**	**Cap. 11**

Fonte: SWEBOK, 2004


Guide to the Software Engineering Body of Knowledge
2004 Version
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Requirements Fundamentals
Requirements Process
Requirements Elicitation
Requirements Analysis
Requirements Specification
Requirements Validation
Practical Considerations
Software Design Fundamentals
Key Issues in Software Design
Software Structure and Architecture
Software Design Quality Analysis and Evaluation
Software Design Notations
Software Design Strategies and Methods
Software Construction Fundamentals
Managing Construction
Practical Considerations
Software Testing Fundamentals
Test Levels
Test Techniques
Test Related Measures
Test Process
Software Maintenance Fundamentals
Key Issues in Software Maintenance
Maintenance Process
Techniques for Maintenance

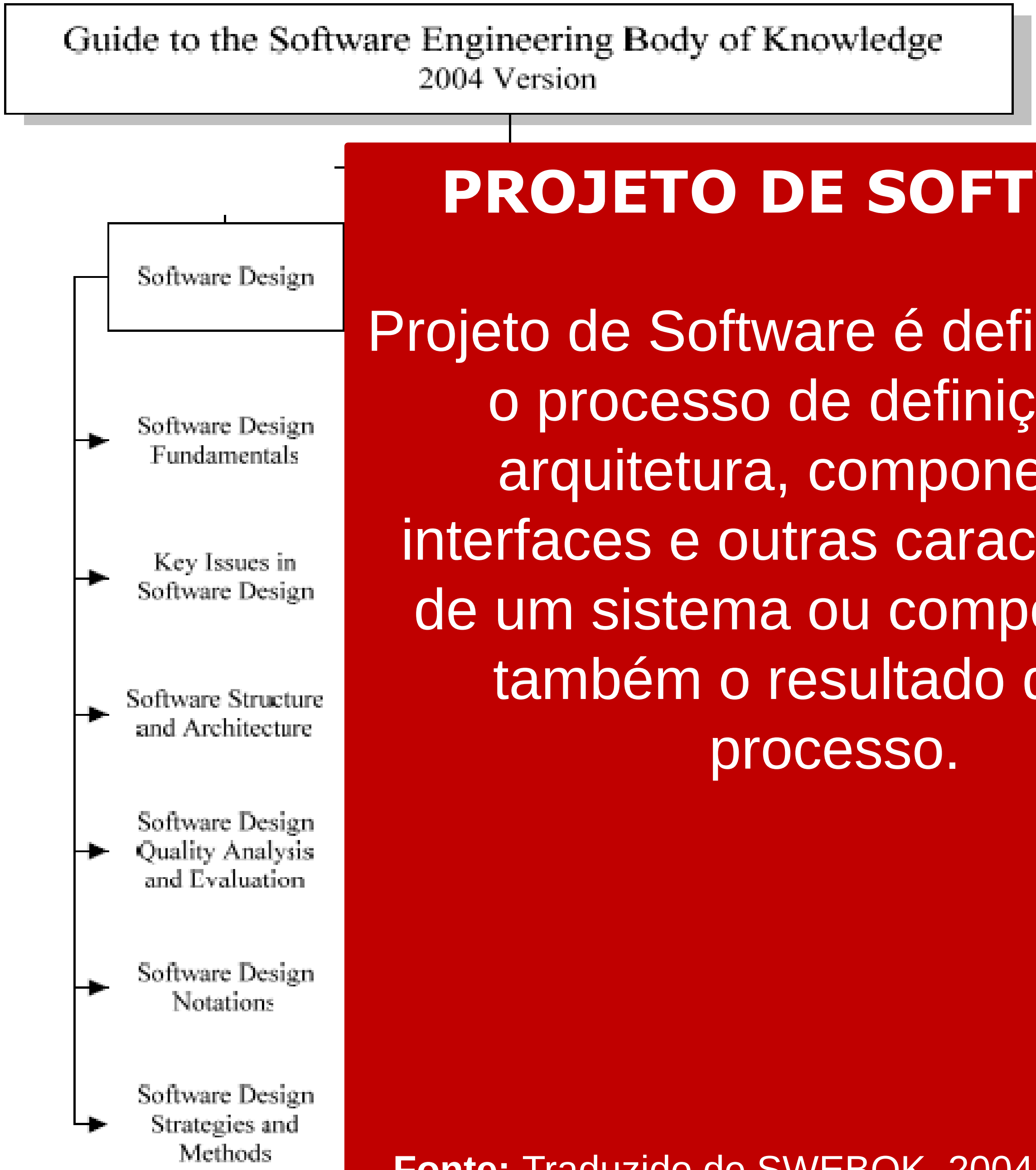

Guide to the Software Engineering Body of Knowledge
(2004 Version)
Software Configuration Management
Management of the SCM Process
Software Configuration Identification
Software Configuration Control
Software Configuration Status Accounting
Software Configuration Auditing
Software Release Management and Delivery
Software Engineering Management
Initiation and Scope Definition
Software Project Planning
Software Project Enactment
Review and Evaluation
Closure
Software Engineering Measurement
Software Engineering Process
Process Implementation and Change
Process Definition
Process Assessment
Process and Product Measurement
Software Engineering Tools and Methods
Software Tools
Software Requirements Tools
Software Design Tools
Software Construction Tools
Software Testing Tools
Software Maintenance Tools
Software Configuration Management Tools
Software Engineering Management Tools
Software Engineering Process Tools
Software Quality Tools
Miscellaneous Tool Issues
Software Engineering Methods
Heuristic Methods
Formal Methods
Prototyping Methods
Software Quality
Software Quality Fundamentals
Software Quality Management Processes
Practical Considerations
Knowledge Areas of the Related Disciplines
Computer Engineering
Computer Science
Management
Mathematics
Project Management
Quality Management
Software Ergonomics
Systems Engineering



REQUISITOS DE SOFTWARE

A Área do Conhecimento de Requisitos de Software está preocupada com a elicitação, análise, especificação e validação da requisitos de software.

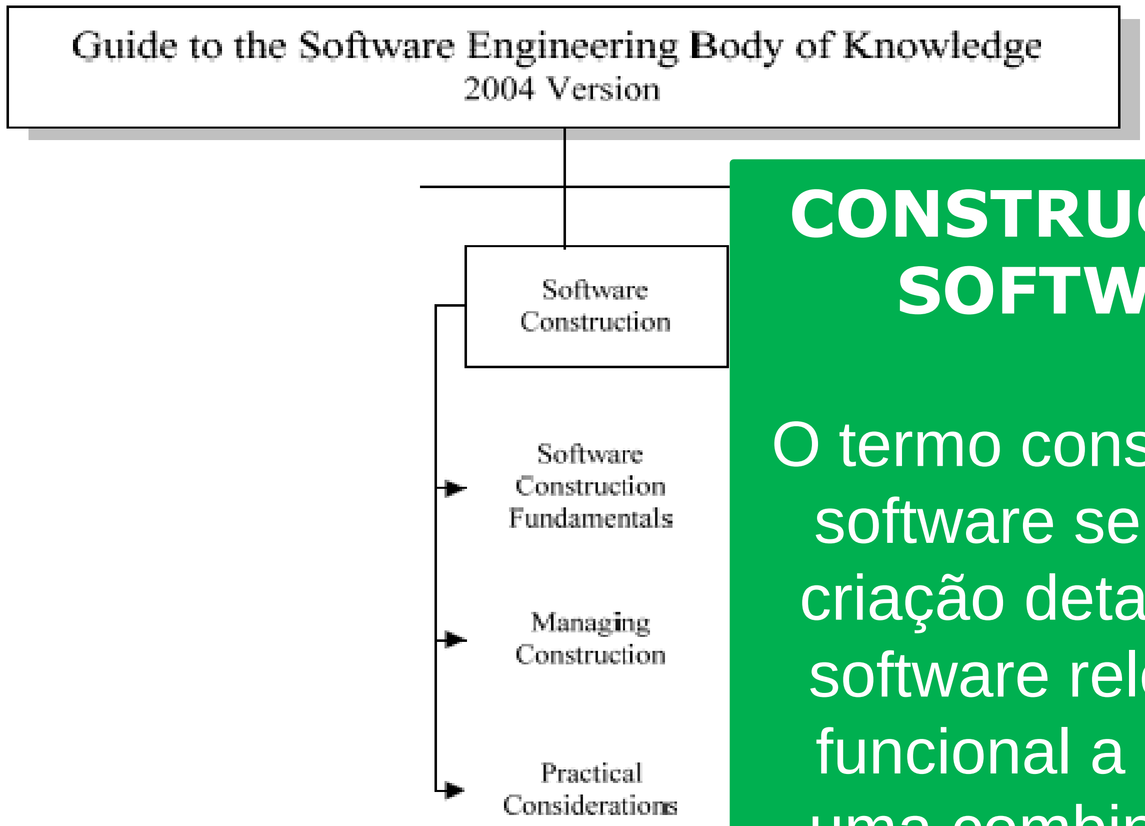
Fonte: Traduzido de SWEBOK, 2004


Guide to the Software Engineering Body of Knowledge
2004 Version
Software Design
Software Design Fundamentals
Key Issues in Software Design
Software Structure and Architecture
Software Design Quality Analysis and Evaluation
Software Design Notations
Software Design Strategies and Methods

PROJETO DE SOFTWARE

Projeto de Software é definido como o processo de definição da arquitetura, componentes, interfaces e outras características de um sistema ou componente e também o resultado desse processo.

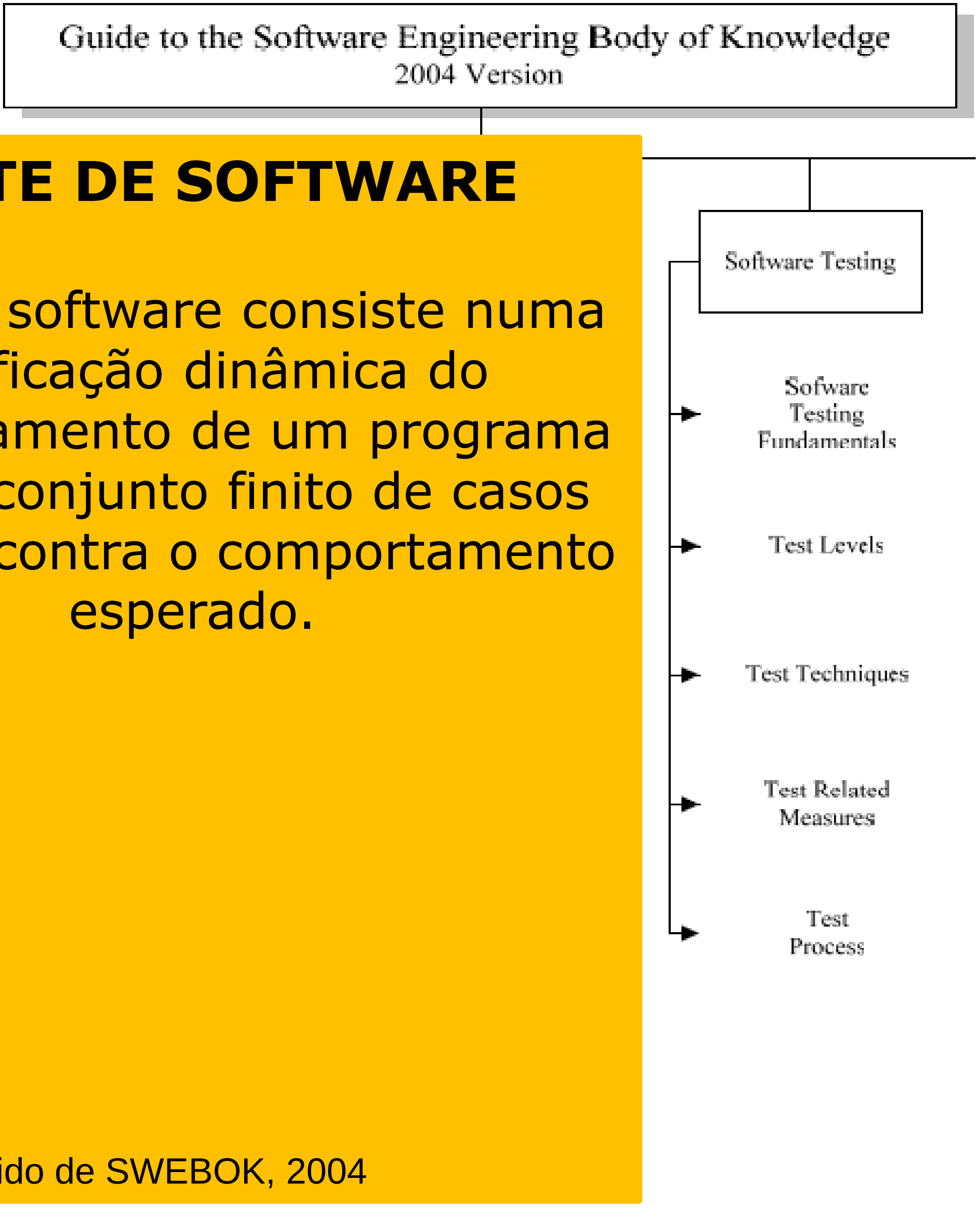
Fonte: Traduzido de SWEBOK, 2004


Guide to the Software Engineering Body of Knowledge
2004 Version
Software Construction
Software Construction Fundamentals
Managing Construction
Practical Considerations

CONSTRUÇÃO DE SOFTWARE

O termo construção de software se refere à criação detalhada de software relevante e funcional a partir de uma combinação de codificação, verificação, teste unitário, teste integrado e debugging.

Fonte: Traduzido de SWEBOK, 2004



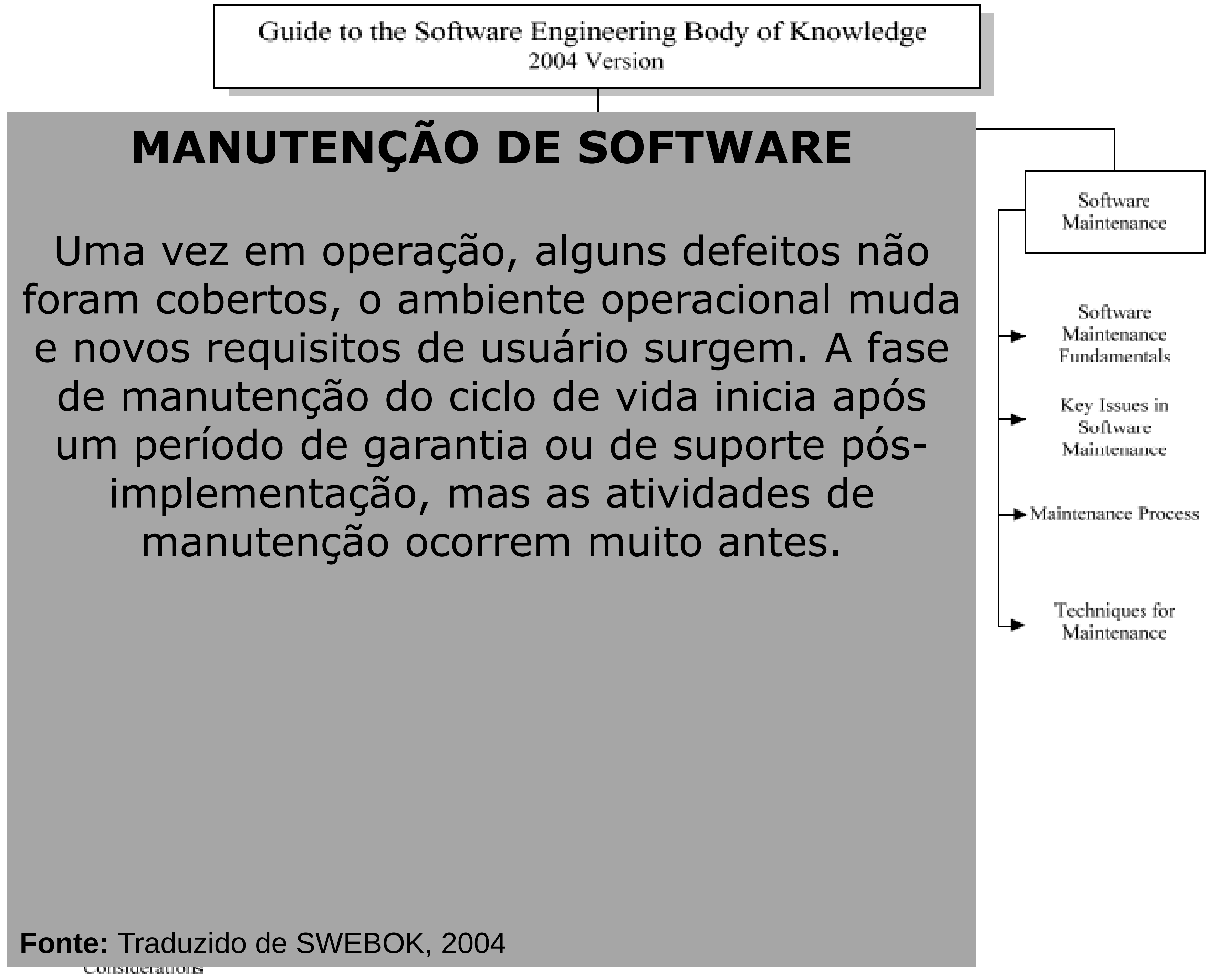
TESTE DE SOFTWARE

Teste de software consiste numa verificação dinâmica do comportamento de um programa em um conjunto finito de casos de teste contra o comportamento esperado.

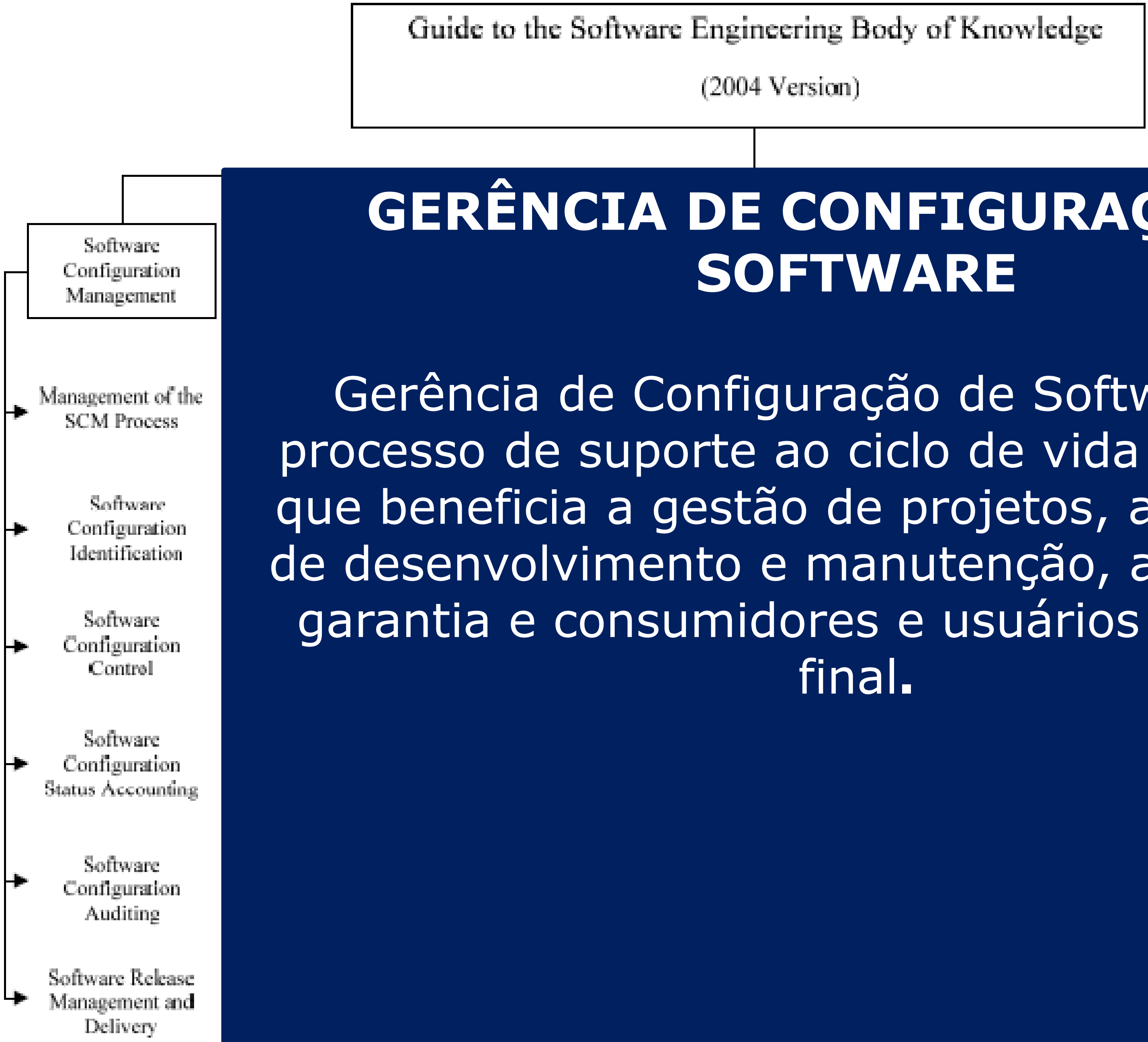
Fonte: Traduzido de SWEBOK, 2004

MANUTENÇÃO DE SOFTWARE

Uma vez em operação, alguns defeitos não foram cobertos, o ambiente operacional muda e novos requisitos de usuário surgem. A fase de manutenção do ciclo de vida inicia após um período de garantia ou de suporte pós-implementação, mas as atividades de manutenção ocorrem muito antes.


Guide to the Software Engineering Body of Knowledge
2004 Version
Software Maintenance
Software Maintenance Fundamentals
Key Issues in Software Maintenance
Maintenance Process
Techniques for Maintenance
Considerations

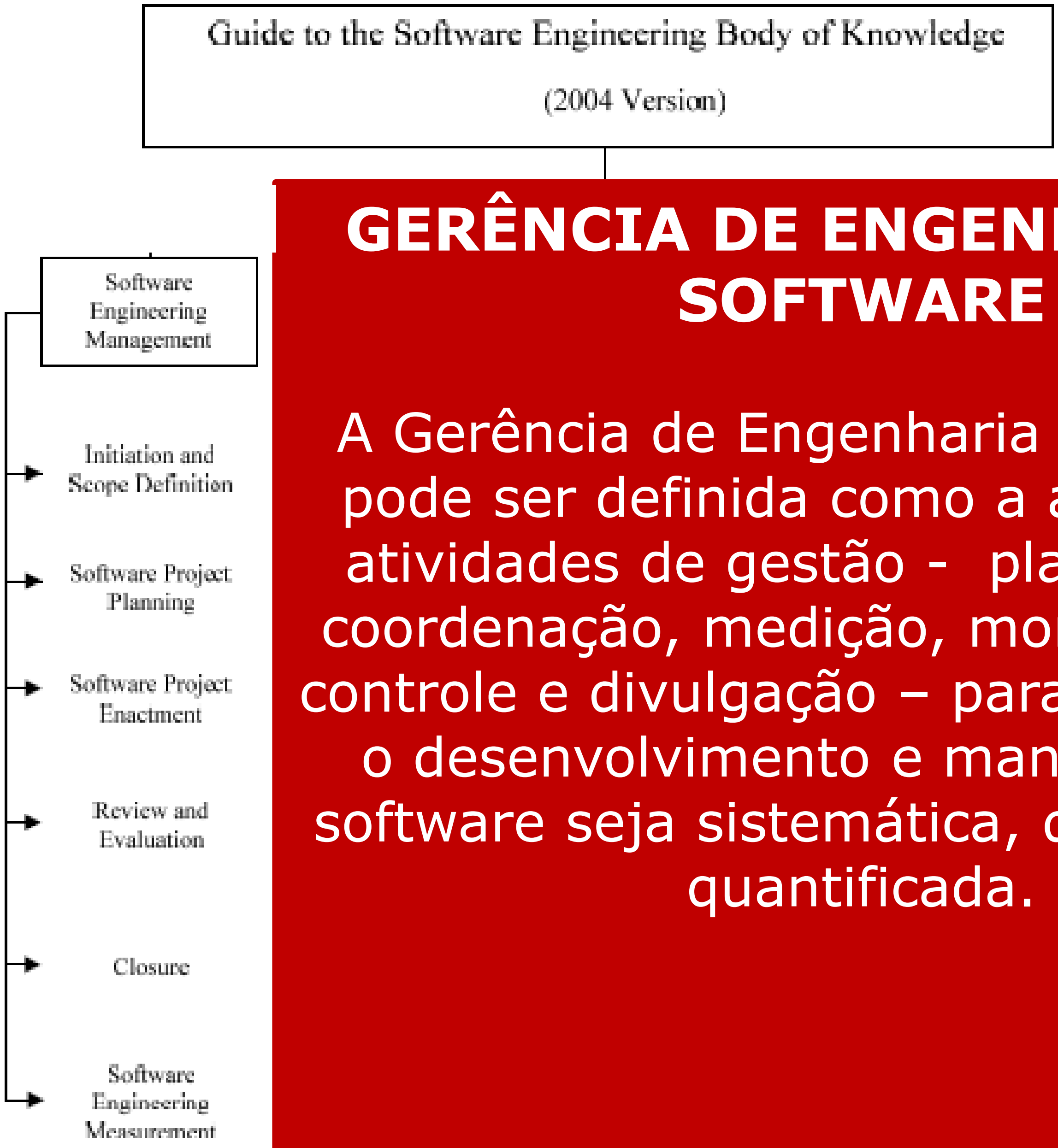
Fonte: Traduzido de SWEBOK, 2004


Guide to the Software Engineering Body of Knowledge
(2004 Version)
Software Configuration Management
Management of the SCM Process
Software Configuration Identification
Software Configuration Control
Software Configuration Status Accounting
Software Configuration Auditing
Software Release Management and Delivery

GERÊNCIA DE CONFIGURAÇÃO DE SOFTWARE

Gerência de Configuração de Software é um processo de suporte ao ciclo de vida do software que beneficia a gestão de projetos, as atividades de desenvolvimento e manutenção, atividades de garantia e consumidores e usuários do produto final.

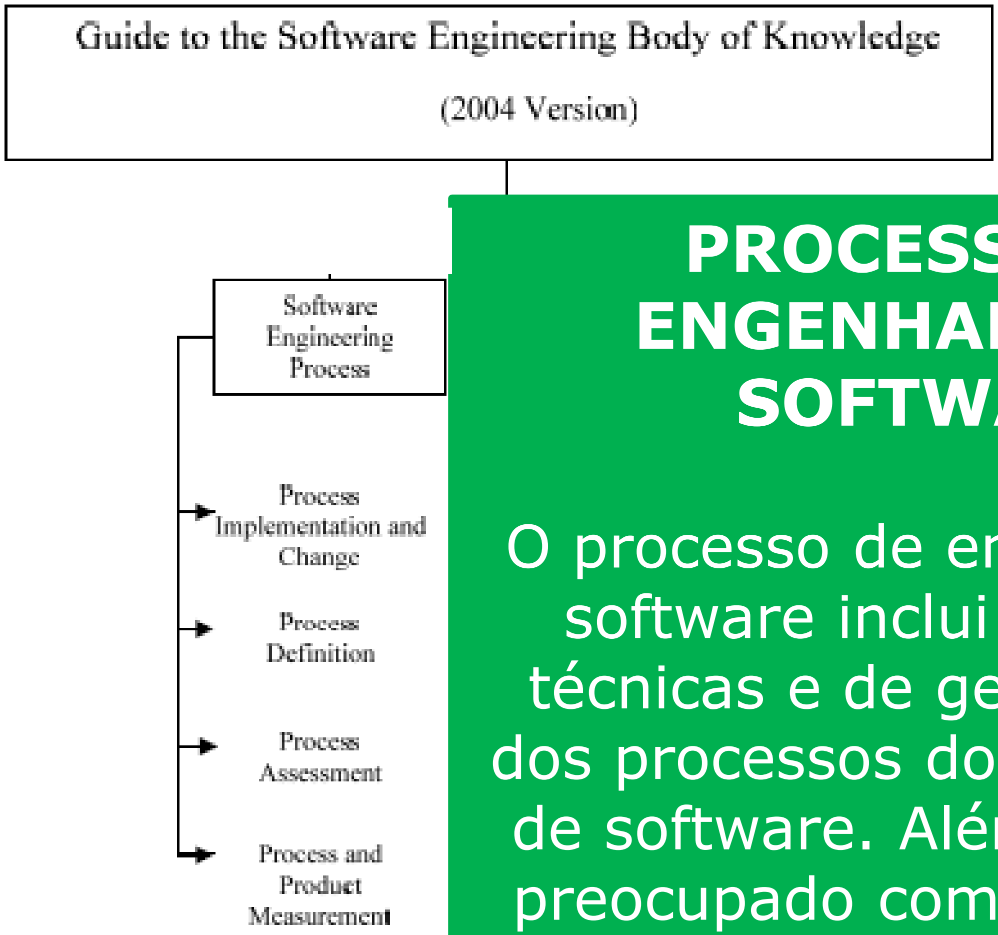
Fonte: Traduzido de SWEBOK, 2004


Guide to the Software Engineering Body of Knowledge
(2004 Version)
Software Engineering Management
Initiation and Scope Definition
Software Project Planning
Software Project Enactment
Review and Evaluation
Closure
Software Engineering Measurement

GERÊNCIA DE ENGENHARIA DE SOFTWARE

A Gerência de Engenharia de Software pode ser definida como a aplicação de atividades de gestão - planejamento, coordenação, medição, monitoramento, controle e divulgação – para garantir que o desenvolvimento e manutenção de software seja sistemática, disciplinada e quantificada.

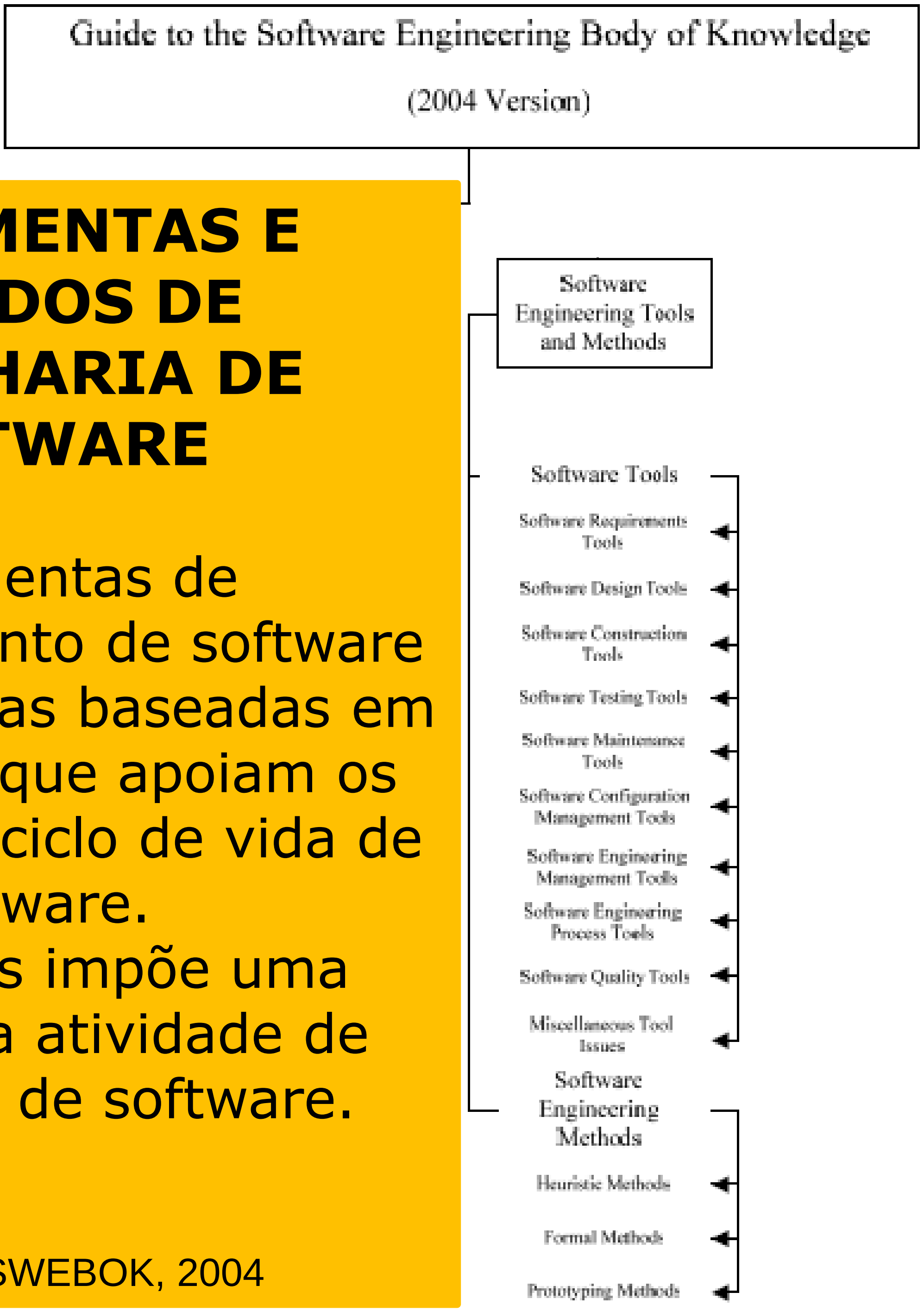
Fonte: Traduzido de SWEBOK, 2004


Guide to the Software Engineering Body of Knowledge
(2004 Version)
Software Engineering Process
Process Implementation and Change
Process Definition
Process Assessment
Process and Product Measurement

PROCESSO DE ENGENHARIA DE SOFTWARE

O processo de engenharia de software inclui atividades técnicas e de gestão dentro dos processos do ciclo de vida de software. Além disso está preocupado com a definição, implementação, avaliação, gerenciamento da mudança e melhorias nos próprios processos do ciclo de vida de software.

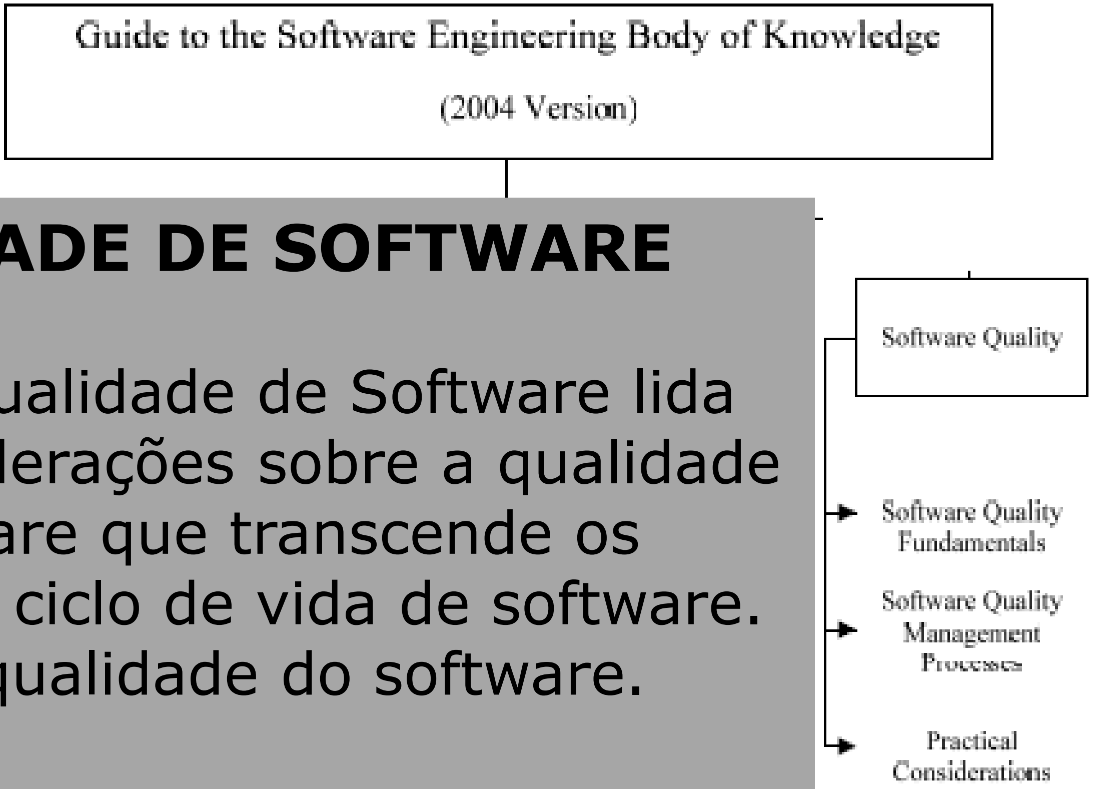
Fonte: Traduzido de SWEBOK, 2004


Guide to the Software Engineering Body of Knowledge
(2004 Version)
Software Engineering Tools and Methods
Software Tools
Software Requirements Tools
Software Design Tools
Software Construction Tools
Software Testing Tools
Software Maintenance Tools
Software Configuration Management Tools
Software Engineering Management Tools
Software Engineering Process Tools
Software Quality Tools
Miscellaneous Tool Issues
Software Engineering Methods
Heuristic Methods
Formal Methods
Prototyping Methods

FERRAMENTAS E MÉTODOS DE ENGENHARIA DE SOFTWARE

Ferramentas de desenvolvimento de software são ferramentas baseadas em computador que apoiam os processos de ciclo de vida de software.
Os métodos impõe uma estrutura na atividade de engenharia de software.

Fonte: Traduzido de SWEBOK, 2004


Guide to the Software Engineering Body of Knowledge
(2004 Version)
Software Quality
Software Quality Fundamentals
Software Quality Management Processes
Practical Considerations

QUALIDADE DE SOFTWARE

A área de Qualidade de Software lida com as considerações sobre a qualidade de software que transcende os processos do ciclo de vida de software. Foca na qualidade do software.

Fonte: Traduzido de SWEBOK, 2004

RESUMO DAS ÁREAS DE CONHECIMENTO

#	Áreas de Conhecimento	Nº de Tópicos	Nº de Subtópicos
1	Requisitos de Software	7	28
2	Projeto de Software	6	25
3	Construção de Software	3	14
4	Teste de Software	5	16
5	Manutenção de Software	4	15
6	Gerenciamento de Configuração de Software	6	17
7	Gerenciamento de Engenharia de Software	6	24
8	Processo de Engenharia de Software	4	16
9	Ferramentas e Métodos de Engenharia de Software	2	12
10	Qualidade de Software	4	11
	Total	**47**	**178**

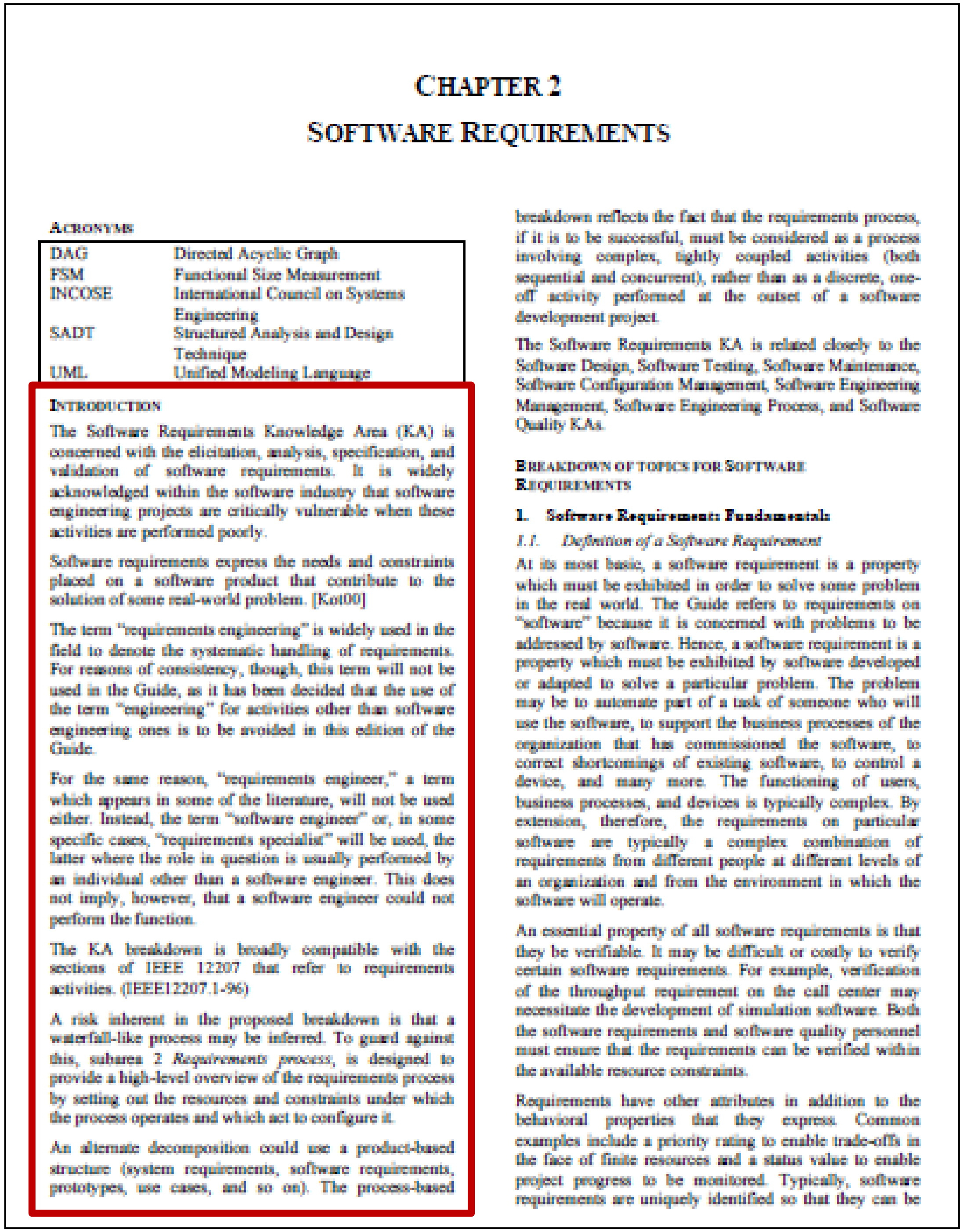
Fonte: SWEBOK, 2004

ESTRUTURA DAS ÁREAS DE CONHECIMENTO (KNOWLEDGE AREAS)

- **PARTE 1:** Definição da área, uma visão geral do seu escopo e de seu relacionamento com as outras áreas do conhecimento;
- **PARTE 2:** Divisão da Área em tópicos, descrevendo a Área do conhecimento em subáreas, tópicos e subtópicos
- **PARTE 3:** Matriz de Tópicos X Material de Referência. O material foi escolhido por ser a melhor apresentação do conhecimento relativo ao tópico.
- **PARTE 4:** Lista de referências recomendadas
- **PARTE 5:** Lista de Leitura Complementar

Fonte: Traduzido de SWEBOK, 2004

- **PARTE 1**: Definição da área, uma visão geral do seu escopo e de seu relacionamento com outras áreas do conhecimento;



CHAPTER 2
SOFTWARE REQUIREMENTS

ACRONYMS

DAG	Directed Acyclic Graph
FSM	Functional Size Measurement
INCOSE	International Council on Systems Engineering
SADT	Structured Analysis and Design Technique
UML	Unified Modeling Language

INTRODUCTION

The Software Requirements Knowledge Area (KA) is concerned with the elicitation, analysis, specification, and validation of software requirements. It is widely acknowledged within the software industry that software engineering projects are critically vulnerable when these activities are performed poorly.

Software requirements express the needs and constraints placed on a software product that contribute to the solution of some real-world problem. [Kot00]

The term "requirements engineering" is widely used in the field to denote the systematic handling of requirements. For reasons of consistency, though, this term will not be used in the Guide, as it has been decided that the use of the term "engineering" for activities other than software engineering ones is to be avoided in this edition of the Guide.

For the same reason, "requirements engineer," a term which appears in some of the literature, will not be used either. Instead, the term "software engineer" or, in some specific cases, "requirements specialist" will be used, the latter where the role in question is usually performed by an individual other than a software engineer. This does not imply, however, that a software engineer could not perform the function.

The KA breakdown is broadly compatible with the sections of IEEE 12207 that refer to requirements activities. (IEEE12207.1-96)

A risk inherent in the proposed breakdown is that a waterfall-like process may be inferred. To guard against this, subarea 2 *Requirements process*, is designed to provide a high-level overview of the requirements process by setting out the resources and constraints under which the process operates and which act to configure it.

An alternate decomposition could use a product-based structure (system requirements, software requirements, prototypes, use cases, and so on). The process-based breakdown reflects the fact that the requirements process, if it is to be successful, must be considered as a process involving complex, tightly coupled activities (both sequential and concurrent), rather than as a discrete, one-off activity performed at the outset of a software development project.

The Software Requirements KA is related closely to the Software Design, Software Testing, Software Maintenance, Software Configuration Management, Software Engineering Management, Software Engineering Process, and Software Quality KAs.

BREAKDOWN OF TOPICS FOR SOFTWARE REQUIREMENTS

1. Software Requirements Fundamentals

1.1. Definition of a Software Requirement

At its most basic, a software requirement is a property which must be exhibited in order to solve some problem in the real world. The Guide refers to requirements on "software" because it is concerned with problems to be addressed by software. Hence, a software requirement is a property which must be exhibited by software developed or adapted to solve a particular problem. The problem may be to automate part of a task of someone who will use the software, to support the business processes of the organization that has commissioned the software, to correct shortcomings of existing software, to control a device, and many more. The functioning of users, business processes, and devices is typically complex. By extension, therefore, the requirements on particular software are typically a complex combination of requirements from different people at different levels of an organization and from the environment in which the software will operate.

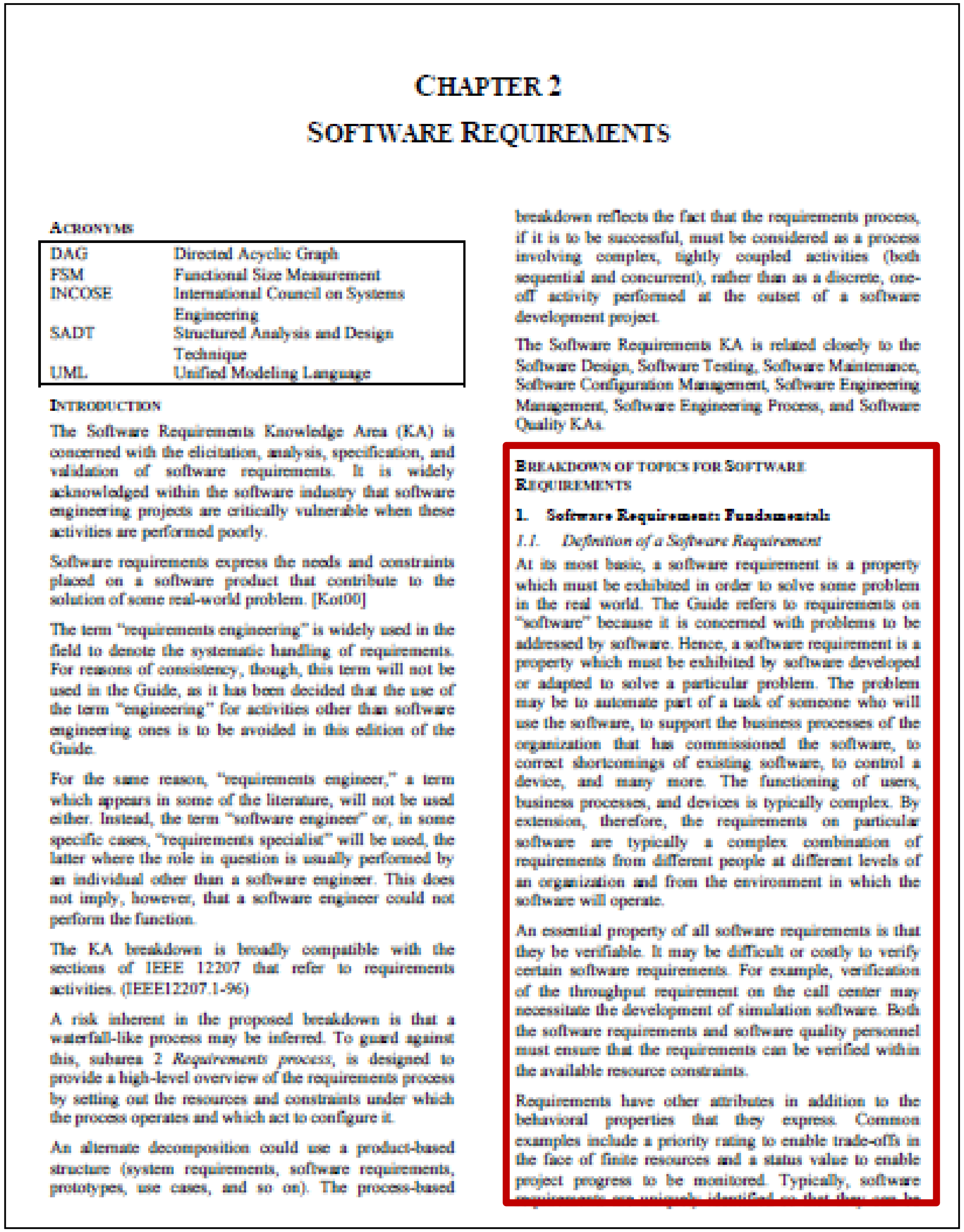
An essential property of all software requirements is that they be verifiable. It may be difficult or costly to verify certain software requirements. For example, verification of the throughput requirement on the call center may necessitate the development of simulation software. Both the software requirements and software quality personnel must ensure that the requirements can be verified within the available resource constraints.

Requirements have other attributes in addition to the behavioral properties that they express. Common examples include a priority rating to enable trade-offs in the face of finite resources and a status value to enable project progress to be monitored. Typically, software requirements are uniquely identified so that they can be

Fonte: SWEBOK, 2004

- **PARTE 2**: Divisão da Área em tópicos, descrevendo a Área do conhecimento em subáreas, tópicos e subtópicos

Fonte: SWEBOK, 2004



CHAPTER 2
SOFTWARE REQUIREMENTS

ACRONYMS

DAG	Directed Acyclic Graph
FSM	Functional Size Measurement
INCOSE	International Council on Systems Engineering
SADT	Structured Analysis and Design Technique
UML	Unified Modeling Language

INTRODUCTION

The Software Requirements Knowledge Area (KA) is concerned with the elicitation, analysis, specification, and validation of software requirements. It is widely acknowledged within the software industry that software engineering projects are critically vulnerable when these activities are performed poorly.

Software requirements express the needs and constraints placed on a software product that contribute to the solution of some real-world problem. [Kot00]

The term "requirements engineering" is widely used in the field to denote the systematic handling of requirements. For reasons of consistency, though, this term will not be used in the Guide, as it has been decided that the use of the term "engineering" for activities other than software engineering ones is to be avoided in this edition of the Guide.

For the same reason, "requirements engineer," a term which appears in some of the literature, will not be used either. Instead, the term "software engineer" or, in some specific cases, "requirements specialist" will be used, the latter where the role in question is usually performed by an individual other than a software engineer. This does not imply, however, that a software engineer could not perform the function.

The KA breakdown is broadly compatible with the sections of IEEE 12207 that refer to requirements activities. (IEEE12207.1-96)

A risk inherent in the proposed breakdown is that a waterfall-like process may be inferred. To guard against this, subarea 2 *Requirements process*, is designed to provide a high-level overview of the requirements process by setting out the resources and constraints under which the process operates and which act to configure it.

An alternate decomposition could use a product-based structure (system requirements, software requirements, prototypes, use cases, and so on). The process-based breakdown reflects the fact that the requirements process, if it is to be successful, must be considered as a process involving complex, tightly coupled activities (both sequential and concurrent), rather than as a discrete, one-off activity performed at the outset of a software development project.

The Software Requirements KA is related closely to the Software Design, Software Testing, Software Maintenance, Software Configuration Management, Software Engineering Management, Software Engineering Process, and Software Quality KAs.

BREAKDOWN OF TOPICS FOR SOFTWARE REQUIREMENTS

1. Software Requirements Fundamentals

1.1. Definition of a Software Requirement

At its most basic, a software requirement is a property which must be exhibited in order to solve some problem in the real world. The Guide refers to requirements on "software" because it is concerned with problems to be addressed by software. Hence, a software requirement is a property which must be exhibited by software developed or adapted to solve a particular problem. The problem may be to automate part of a task of someone who will use the software, to support the business processes of the organization that has commissioned the software, to correct shortcomings of existing software, to control a device, and many more. The functioning of users, business processes, and devices is typically complex. By extension, therefore, the requirements on particular software are typically a complex combination of requirements from different people at different levels of an organization and from the environment in which the software will operate.

An essential property of all software requirements is that they be verifiable. It may be difficult or costly to verify certain software requirements. For example, verification of the throughput requirement on the call center may necessitate the development of simulation software. Both the software requirements and software quality personnel must ensure that the requirements can be verified within the available resource constraints.

Requirements have other attributes in addition to the behavioral properties that they express. Common examples include a priority rating to enable trade-offs in the face of finite resources and a status value to enable project progress to be monitored. Typically, software requirements are uniquely identified so that they can be

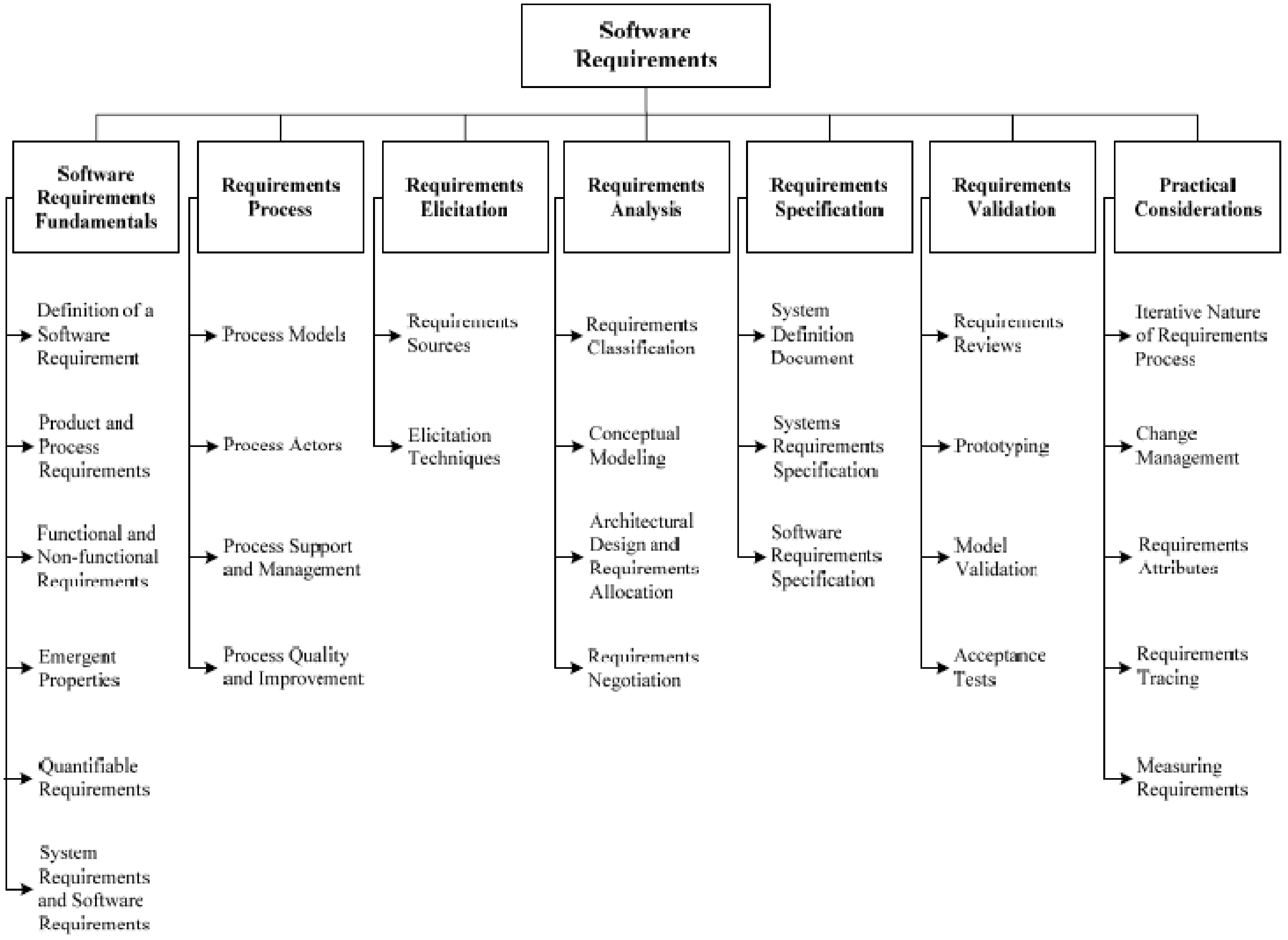
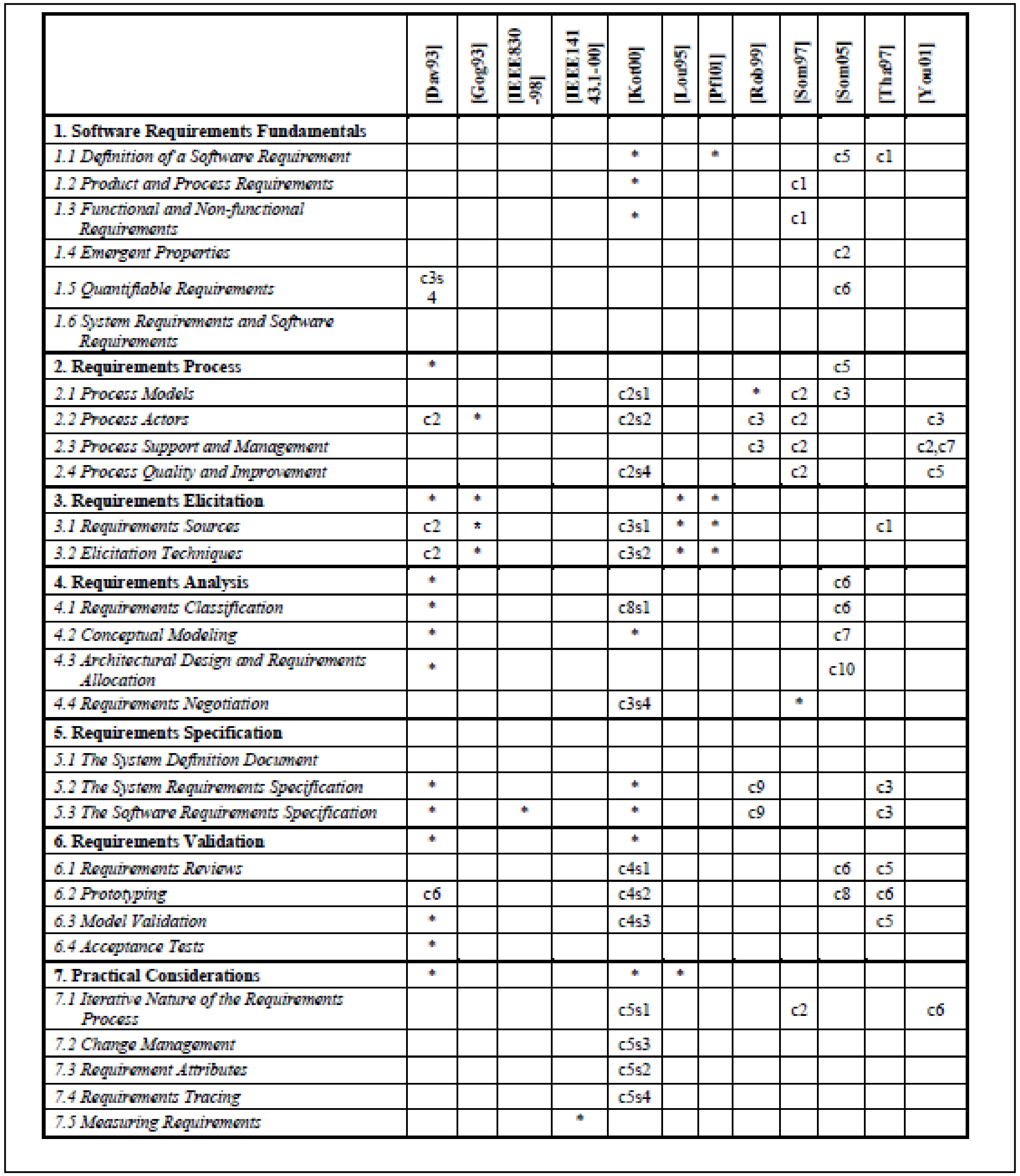

Software Requirements
Software Requirements Fundamentals
Definition of a Software Requirement
Product and Process Requirements
Functional and Non-functional Requirements
Emergent Properties
Quantifiable Requirements
System Requirements and Software Requirements
Requirements Process
Process Models
Process Actors
Process Support and Management
Process Quality and Improvement
Requirements Elicitation
Requirements Sources
Elicitation Techniques
Requirements Analysis
Requirements Classification
Conceptual Modeling
Architectural Design and Requirements Allocation
Requirements Negotiation
Requirements Specification
System Definition Document
Systems Requirements Specification
Software Requirements Specification
Requirements Validation
Requirements Reviews
Prototyping
Model Validation
Acceptance Tests
Practical Considerations
Iterative Nature of Requirements Process
Change Management
Requirements Attributes
Requirements Tracing
Measuring Requirements

Figure 1 Breakdown of topics for the Software Requirements KA

Fonte: SWEBOK, 2004

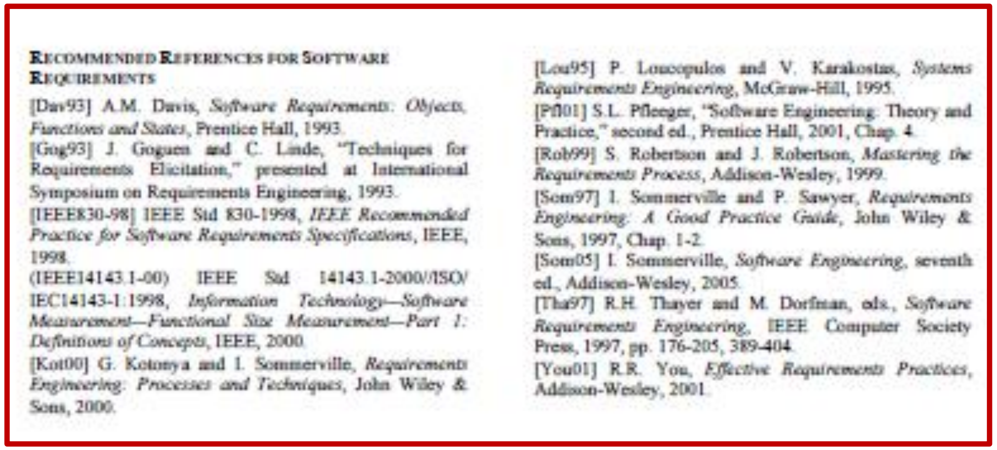
- **PARTE 3**: Matriz de Tópicos x Material de Referência



	[Dav93]	[Gog93]	[IEEE830-98]	[IEEE1413.1-00]	[Kot00]	[Lou95]	[Pfl01]	[Rob99]	[Som97]	[Som05]	[Tha97]	[You01]
1. Software Requirements Fundamentals												
1.1 Definition of a Software Requirement					*		*			c5	c1	
1.2 Product and Process Requirements					*				c1			
1.3 Functional and Non-functional Requirements					*				c1			
1.4 Emergent Properties										c2		
1.5 Quantifiable Requirements	c3s4									c6		
1.6 System Requirements and Software Requirements												
2. Requirements Process	*									c5		
2.1 Process Models					c2s1			*	c2	c3		
2.2 Process Actors	c2	*			c2s2			c3	c2			c3
2.3 Process Support and Management								c3	c2			c2,c7
2.4 Process Quality and Improvement					c2s4				c2			c5
3. Requirements Elicitation	*	*				*	*					
3.1 Requirements Sources	c2	*			c3s1	*	*				c1	
3.2 Elicitation Techniques	c2	*			c3s2	*	*					
4. Requirements Analysis	*									c6		
4.1 Requirements Classification	*				c8s1					c6		
4.2 Conceptual Modeling	*				*					c7		
4.3 Architectural Design and Requirements Allocation	*									c10		
4.4 Requirements Negotiation					c3s4				*			
5. Requirements Specification												
5.1 The System Definition Document												
5.2 The System Requirements Specification	*				*			c9			c3	
5.3 The Software Requirements Specification	*		*		*			c9			c3	
6. Requirements Validation	*				*							
6.1 Requirements Reviews					c4s1					c6	c5	
6.2 Prototyping	c6				c4s2					c8	c6	
6.3 Model Validation	*				c4s3						c5	
6.4 Acceptance Tests	*											
7. Practical Considerations	*				*	*						
7.1 Iterative Nature of the Requirements Process					c5s1				c2			c6
7.2 Change Management					c5s3							
7.3 Requirement Attributes					c5s2							
7.4 Requirements Tracing					c5s4							
7.5 Measuring Requirements				*								

Fonte: SWEBOK, 2004

- **PARTE 4**: Lista de referências recomendadas



Recommended References for Software Requirements

[Dav93] A.M. Davis, *Software Requirements: Objects, Functions and States*, Prentice Hall, 1993.

[Gog93] J. Goguen and C. Linde, "Techniques for Requirements Elicitation," presented at International Symposium on Requirements Engineering, 1993.

[IEEE830-98] IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*, IEEE, 1998.

(IEEE14143.1-00) IEEE Std 14143.1-2000//ISO/IEC14143-1:1998, *Information Technology—Software Measurement—Functional Size Measurement—Part 1: Definitions of Concepts*, IEEE, 2000.

[Kot00] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, 2000.

[Lou95] P. Loucopulos and V. Karakostas, *Systems Requirements Engineering*, McGraw-Hill, 1995.

[Pfl01] S.L. Pfleeger, "Software Engineering: Theory and Practice," second ed., Prentice Hall, 2001, Chap. 4.

[Rob99] S. Robertson and J. Robertson, *Mastering the Requirements Process*, Addison-Wesley, 1999.

[Som97] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons, 1997, Chap. 1-2.

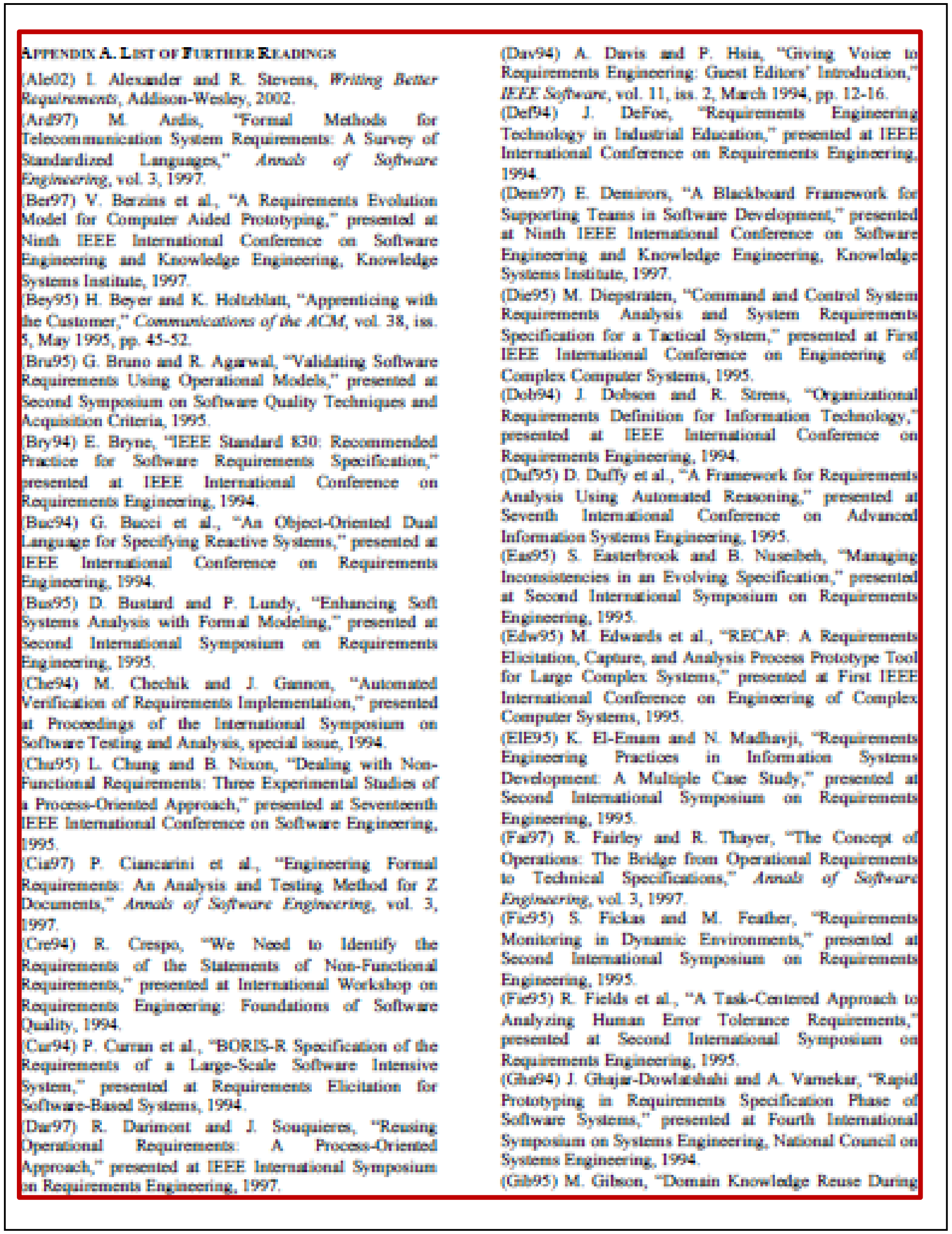
[Som05] I. Sommerville, *Software Engineering*, seventh ed., Addison-Wesley, 2005.

[Tha97] R.H. Thayer and M. Dorfman, eds., *Software Requirements Engineering*, IEEE Computer Society Press, 1997, pp. 176-205, 389-404.

[You01] R.R. You, *Effective Requirements Practices*, Addison-Wesley, 2001.

Fonte: SWEBOK, 2004

- **PARTE 5**: Lista de Leitura Complementar



APPENDIX A. LIST OF FURTHER READINGS

(Ale02) I. Alexander and R. Stevens, *Writing Better Requirements*, Addison-Wesley, 2002.

(Ard97) M. Ardis, "Formal Methods for Telecommunication System Requirements: A Survey of Standardized Languages," *Annals of Software Engineering*, vol. 3, 1997.

(Ber97) V. Berzins et al., "A Requirements Evolution Model for Computer Aided Prototyping," presented at Ninth IEEE International Conference on Software Engineering and Knowledge Engineering, Knowledge Systems Institute, 1997.

(Bey95) H. Beyer and K. Holtzblatt, "Apprenticing with the Customer," *Communications of the ACM*, vol. 38, iss. 5, May 1995, pp. 45-52.

(Bru95) G. Bruno and R. Agarwal, "Validating Software Requirements Using Operational Models," presented at Second Symposium on Software Quality Techniques and Acquisition Criteria, 1995.

(Bry94) E. Bryne, "IEEE Standard 830: Recommended Practice for Software Requirements Specification," presented at IEEE International Conference on Requirements Engineering, 1994.

(Buc94) G. Bucci et al., "An Object-Oriented Dual Language for Specifying Reactive Systems," presented at IEEE International Conference on Requirements Engineering, 1994.

(Bus95) D. Bustard and P. Lundy, "Enhancing Soft Systems Analysis with Formal Modeling," presented at Second International Symposium on Requirements Engineering, 1995.

(Che94) M. Chechik and J. Gannon, "Automated Verification of Requirements Implementation," presented at Proceedings of the International Symposium on Software Testing and Analysis, special issue, 1994.

(Chu95) L. Chung and B. Nixon, "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach," presented at Seventeenth IEEE International Conference on Software Engineering, 1995.

(Cia97) P. Ciancarini et al., "Engineering Formal Requirements: An Analysis and Testing Method for Z Documents," *Annals of Software Engineering*, vol. 3, 1997.

(Cre94) R. Crespo, "We Need to Identify the Requirements of the Statements of Non-Functional Requirements," presented at International Workshop on Requirements Engineering: Foundations of Software Quality, 1994.

(Cur94) P. Curran et al., "BORIS-R Specification of the Requirements of a Large-Scale Software Intensive System," presented at Requirements Elicitation for Software-Based Systems, 1994.

(Dar97) R. Darimont and J. Souquieres, "Reusing Operational Requirements: A Process-Oriented Approach," presented at IEEE International Symposium on Requirements Engineering, 1997.

(Dav94) A. Davis and P. Hsia, "Giving Voice to Requirements Engineering: Guest Editors' Introduction," *IEEE Software*, vol. 11, iss. 2, March 1994, pp. 12-16.

(Def94) J. DeFoe, "Requirements Engineering Technology in Industrial Education," presented at IEEE International Conference on Requirements Engineering, 1994.

(Dem97) E. Demirors, "A Blackboard Framework for Supporting Teams in Software Development," presented at Ninth IEEE International Conference on Software Engineering and Knowledge Engineering, Knowledge Systems Institute, 1997.

(Die95) M. Diepstraten, "Command and Control System Requirements Analysis and System Requirements Specification for a Tactical System," presented at First IEEE International Conference on Engineering of Complex Computer Systems, 1995.

(Dob94) J. Dobson and R. Strens, "Organizational Requirements Definition for Information Technology," presented at IEEE International Conference on Requirements Engineering, 1994.

(Duf95) D. Duffy et al., "A Framework for Requirements Analysis Using Automated Reasoning," presented at Seventh International Conference on Advanced Information Systems Engineering, 1995.

(Eas95) S. Easterbrook and B. Nuseibeh, "Managing Inconsistencies in an Evolving Specification," presented at Second International Symposium on Requirements Engineering, 1995.

(Edw95) M. Edwards et al., "RECAP: A Requirements Elicitation, Capture, and Analysis Process Prototype Tool for Large Complex Systems," presented at First IEEE International Conference on Engineering of Complex Computer Systems, 1995.

(ElE95) K. El-Emam and N. Madhavji, "Requirements Engineering Practices in Information Systems Development: A Multiple Case Study," presented at Second International Symposium on Requirements Engineering, 1995.

(Fai97) R. Fairley and R. Thayer, "The Concept of Operations: The Bridge from Operational Requirements to Technical Specifications," *Annals of Software Engineering*, vol. 3, 1997.

(Fic95) S. Fickas and M. Feather, "Requirements Monitoring in Dynamic Environments," presented at Second International Symposium on Requirements Engineering, 1995.

(Fie95) R. Fields et al., "A Task-Centered Approach to Analyzing Human Error Tolerance Requirements," presented at Second International Symposium on Requirements Engineering, 1995.

(Gha94) J. Ghajar-Dowlatshahi and A. Varnekar, "Rapid Prototyping in Requirements Specification Phase of Software Systems," presented at Fourth International Symposium on Systems Engineering, National Council on Systems Engineering, 1994.

(Gib95) M. Gibson, "Domain Knowledge Reuse During

Fonte: SWEBOK, 2004

DISCIPLINAS RELACIONADAS

DISCIPLINAS RELACIONADAS A ENGENHARIA DE SOFTWARE

- Disciplinas relacionadas à Engenharia de Software:

Engenharia da Computação	Gestão de Projetos
Ciência da Computação	Gestão da Qualidade
Administração	Ergonomia de Software
Matemática	Engenharia de Sistemas

Fonte: SWEBOK, 2004

EVOLUÇÃO DO GUIA SWEBOK (V. 3) – 21/08/2012

#	Áreas de Conhecimento	
1	Requisitos de Software	Em finalização para revisão
2	Projeto de Software	Disponível para revisão
3	Construção de Software	Finalização de Versão Beta
4	Teste de Software	Em finalização para revisão
5	Manutenção de Software	Finalização de Versão Beta
6	Gerência de Configuração de Software	Finalização de Versão Beta
7	Gerência da Engenharia de Software	Disponível para revisão
8	Processo de Engenharia de Software	Em finalização para revisão
9	Modelos e Métodos de Engenharia de Software	Finalização de Versão Beta
10	Qualidade de Software	Em finalização para revisão
11	Prática Profissional de Engenharia de Software	Disponível para revisão
12	Economia da Engenharia de Software	Em finalização para revisão
13	Fundamentos de Computação	Versão Beta aprovada
14	Fundamentos de Matemática	Finalização de Versão Beta
15	Fundamentos de Engenharia	Em finalização para revisão

CONSIDERAÇÕES FINAIS

- Os tópicos listados como “geralmente aceitos” no Guia foram cuidadosamente selecionados, no entanto, inevitavelmente, esta seleção precisa evoluir.
- O volume de literatura publicado sobre engenharia de software é considerável, por essa razão as referências bibliográficas indicadas neste Guia não devem ser vistas como uma seleção definitiva, mas sim como uma seleção razoável.

Fonte: Traduzido de SWEBOK, 2004

REFERÊNCIAS BIBLIOGRÁFICAS

- **IEEE**. **About IEEE**. Disponível em http://www.ieee.org/index.html?WT.mc_id=hpf_logo. Acesso em 13 de outubro de 2012.
- **IEEE.** Guide to the Software Engineering Body of Knowledge (SWEBOK). 2004 Version. Disponível em: http://www.swebok.org . Acesso em 13 de outubro de 2012.

OBRIGADO!

vagar@cin.ufpe.br

about.me /vitoragar