A Technical Review of the Software Construction Knowledge Area in the SWEBOK Guide (Trial Version 1.0)

François Robert, Alain Abran, Pierre Bourque
École de Technologie Supérieure
(Revision: Sept. 6, 2002)

Abstract

The editorial team of the SWEBOK guide received feedback about its use at the National Technical University (NTU) confirming usefulness of the guide with the exception of chapter four, Software Construction, that did not map easily to industry practices nor to actual academic curriculum.

After analysis of this specific SWEBOK chapter, some issues were identified, such as inconstancies between the textual descriptions and the visual representation. Furthermore, the analysis of this chapter using the Vincenti classification of engineering knowledge types allowed to identify some further weaknesses and provided some guidance on how the structure of this chapter could be improved.

This paper proposes a revised breakdown of topics that is more aligned with an engineering perspective.

1 Introduction

The SWEBOK project was established with five objectives:

1. Characterise the contents of the software engineering discipline.
2. Provide a topical access to the Software Engineering Body of Knowledge.
3. Promote a consistent view of software engineering worldwide.
4. Clarify the place, and set the boundary, of software engineering with respect to other disciplines such as computer science, project management, computer engineering and mathematics.
5. Provide a foundation for curriculum development and individual certification material.

It must be emphasised that the product of the SWEBOK project is *not* the Body of Knowledge itself, but rather a guide to this knowledge. The knowledge already exists and the purpose of the project is to gain consensus on a characterisation of that knowledge which illuminates the nature of the software engineering discipline and explains what knowledge is generally accepted.

In May 2001, trial version 0.95 of the Guide to the Software Engineering Body of Knowledge (SWEBOK) was released in a web format, and, in December 2001, it was published in a book format [ABR01]. The guide is the result of more than three years of review by over five hundred members of the software engineering community. Two important principles guided the review process: *transparency* and *consensus*.

- *Transparency:* the development process is itself documented, published and publicised so that important decisions and status are visible to all concerned parties;
- *Consensus:* the only practical method for legitimising a statement of this kind is through broad participation and agreement by all significant sectors of the relevant community.

The guide is now in a trial period of two years by its targeted audiences:

- Private and public organisations desiring a consistent view of software engineering for the purpose of defining education and training requirements, classifying jobs and developing performance evaluation policies;
- Practising software engineers;
- Makers of public policy regarding licensing and professional guidelines;
- Professional societies defining accreditation and certification policies for university curricula and guidelines for professional practice;
- Educators and trainers defining curricula and course content;
- Students of software engineering.

The feedback collected during this trial period will be analysed and will serve as the basis for further improvements to the Guide. The SWEBOK editorial team has already received feedback on the use of the Guide by the National Technological University [FRAI02] which is using it as the neutral basis for the evaluation of software engineering courses offered by its member universities in the USA. Feedback received confirmed the usefulness of the Guide for all documented Knowledge Areas, with the exception of the

Software Construction chapter because its content did not map easily to industry practices or to actual academic curriculum.

The current breakdown of topics in this Knowledge Area is reviewed in section 2, followed in section 3 by an analysis using the Vincenti classification of engineering knowledge types [VIN90]. Based on the insights gained from this analysis, Section 4 proposes an improved breakdown of topics which is more aligned and consistent with breakdowns and taxonomies of the other Knowledge Areas of the SWEBOK Guide. Observations for further work are summarised in section 5.

2 Breakdown of topics

2.1 Current breakdown representation

While reviewing the SWEBOK Construction Knowledge Area to understand the feedback from NTU [FRAI02], we observed that the breakdown of topics, as presented on pages 4 to 10 of the SWEBOK Guide and reproduced here in Figure 1, did not correspond to the actual structure of the chapter itself. More specifically, the content of Figure 1 currently corresponds only to the text of a single section of chapter 4, that is 3.3. Figure 1 is therefore is not an accurate representation of the full chapter, as it deals with only a subset of the content of the whole chapter, and some elements of knowledge, or topics, are not included in the breakdown representation. In addition, there are significant duplications in the figure itself. By comparison, in the other chapters of the SWEBOK Guide, all topics have been included in the taxonomies (or breakdowns) of the respective Knowledge Areas.

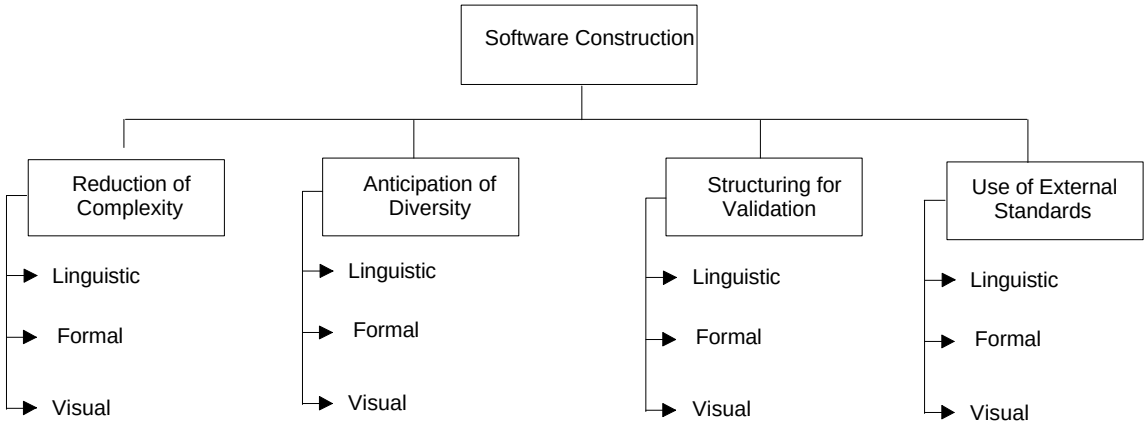

Figure 1: Breakdown of topics as represented in the SWEBOK Guide (Trial version 1.0)

2.2 Initial revised breakdown representation

To facilitate analysis of this Knowledge Area, we redrafted the representation of the breakdown of topics on the basis of the full set of concepts as actually described textually in the chapter. The corrected breakdown representation is presented in Figure 2. The comparison of both figures provides easily indications that the content of chapter four is much more comprehensive and rich in concepts that what was proposed by Figure 1.

In later steps, on the basis of further analysis from an engineering viewpoint, we propose further improvements to this initial revision.

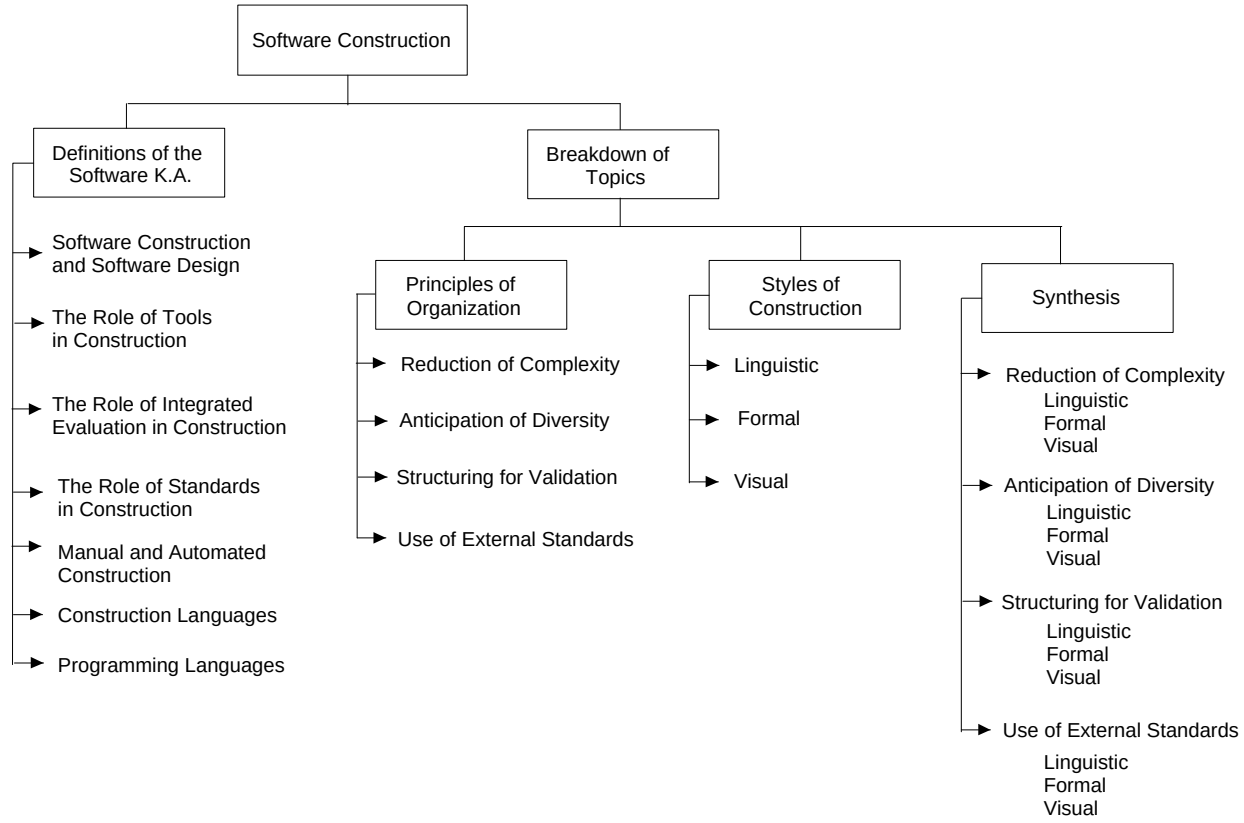

Figure 2: Initial revised breakdown of topics (basis: actual text in the SWEBOK Guide)

3 Analysis using the Vincenti classification

To gain further insights into the content of this chapter from an engineering viewpoint, we selected the classification of engineering knowledge types by Vincenti [VIN90] as our analytical tool for this study. Vincenti, on the basis of his analysis of the evolution of aerospace engineering knowledge, identified different types of engineering knowledge, and classified them into six categories:

1. Fundamental design concepts
2. Criteria and specifications
3. Theoretical tools
4. Quantitative data
5. Practical considerations
6. Design *instrumentalities*

Vincenti postulated that this classification was not specific to aerospace engineering, but more generic and applicable to engineering in the broad sense. However, to our knowledge, his classification has not yet been applied to the traditional disciplines of engineering. Maibaum [MAI01], in contrast, has used Vincenti's classification to investigate the mathematical foundations of software engineering, but did not look at specific Knowledge Areas within software engineering.

We used Vincenti's classification to recognise and identify, within this chapter of the SWEBOK Guide, the types of engineering knowledge included with its current status and description. By extension, this analysis will also provide insights into the elements of engineering knowledge that could be missing, either because they do not exist or because they have been missed in the information gathering and successive review processes.

In this report, we have not, of course, used the Vincenti classification on the generic domain of software engineering, but on one of its Knowledge Areas in the SWEBOK Guide, that is, software construction. In the specific context of software construction engineering, the six categories can be described by the following tailored definitions:

- *Fundamental design concepts:* can be viewed as the "operational principles of the *device"*, the device being the software construction technology itself, including not only tools but also procedural knowledge (engineering 'know-how').

- *Criteria and specifications:* the specific technical specifications applied to the construction of software.

- *Theoretical tools:* the mathematical or formal methods and theories for the design calculations involved in software construction.

- *Quantitative data:* the data obtained empirically or calculated based on a theoretical model.

- *Practical considerations:* the activities, which are not formally codified but are represented by rules of thumb.

- *Design instrumentalities:* in this case, the procedural knowledge.

The analysis of this SWEBOK chapter from the above interpretation of Vincenti viewpoint can be summarised as follows:

- Fundamental design concepts (for the construction of software): three in particular are found under the three topic headings of reduction in complexity, structuring for validation and linguistic style of construction. More specifically:
  a) Reduction in complexity is recognised as being a crucial element of software construction, and even beyond construction. Several studies referenced in this chapter have addressed the topic and demonstrated its importance.
  b) Structuring for validation means constructing software with the intention of validating that it functions well by various means. This topic is well covered in a number of studies referenced in the chapter and can be considered fundamental knowledge.
  c) Linguistic style of construction is, by definition, fundamental knowledge in the computer sciences, and, by extension, in software engineering.

- Criteria and specifications: some of them are inherent in both the role of standards and the use of external standards. More specifically:
  d) Standards in software construction are mainly "prescriptive" and as such represent criteria and specifications.

- Theoretical tools: some of which can be located only in the formal style of construction. More specifically:
  e) Formal type of construction: the only theoretical tools available are described in this chapter. They are considered theoretical since they follow strict formal rules.

- Quantitative data: none of them are documented or referenced in the text.

- Practical considerations: they have be identified in four topic areas, that is, software construction and software design, role of integrated evaluation, manual and automated construction and anticipation of diversity. More specifically:
  f) The difference between what is in the software design area and what is in the software construction area is determined by practical considerations, since there are no existing rules, theories or procedures for doing so.
  g) Integrated evaluation is described as including all activities such as peer reviews and code testing. Surprisingly, there is no reference to procedural knowledge for such evaluation, and have therefore elected to classify these activities as practical considerations.
  h) Choosing between manual and automated construction of software is not based on structured or formal knowledge. According to Andrew Hunt and David Thomas [HU001], the programmer must refer to common sense for choosing between

manual and automated construction. Common sense must be classified as a practical consideration.

i) Similarly, there is no reference in the SWEBOK Guide that anticipation of diversity is based on formal knowledge. There are studies that demonstrate that requirements change, but little work has been done on the impact of such changes on software construction. For the time being, this should still be considered as practical knowledge.

- Design instrumentalities, some of which were identified as belonging to the role of integrated evaluation, construction languages, programming languages and the visual style of construction. More specifically:
  j) Choosing the right tool (language, compiler, etc.) for a given problem is a structured and documented skill.
  k) Construction languages that include programming languages constitute, almost by definition, procedural knowledge. Language design and usage are skills, and both are inherited from the computer sciences.
  l) Visual type of construction is not based on formal rules or specific theory, but procedures are followed. In this sense, the visual type can be classified under design instrumentalities.

Table 1 presents, in summary, the classification of each topic of the current breakdown according to Vincenti definition of the types of knowledge in engineering. Each letter found in columns refers to a label from the previous paragraphs. It can be readily observed from Table 1 that the content of this SWEBOK chapter, when classified by engineering types of knowledge, does not map easily to the current taxonomy (e.g. structure) of this Software Construction chapter.

	F.D.	C.S.	T.T.	Q.D.	P.C.	D.I.
2.0 Definition						
2.1 Software construction and software design					f	
2.2 The role of tools in construction						j
2.3The role of integrated evaluation in construction.					g	
2.4 The role of standards in construction		d				
2.5 Manual and automated construction					h	
2.6 Construction Languages						k
2.7 Programming Languages						k
3.0 Breakdown						
3.1 Principle of organisation						
3.1.1 Reduction in complexity	a					
3.1.2 Anticipation of diversity					i	
3.1.3 Structuring for validation	b					
3.1.4 Use of external standards		d				
3.2 Style of construction						
3.2.1 Linguistic	c					
3.2.2 Formal			e			
3.2.3 Visual						l

F.P: Fundamental Design Concepts C.S: Criteria and Specifications
T.T: Theoretical Tools Q.D: Quantitative Data
P.C: Practical Considerations D.I: Design Instrumentalities

Table 1: Analysis of Construction KA using the Vincenti classification

4 Proposed new breakdown

From section 2, and a comparison of Figure 2 with the breakdowns of topics in the other Knowledge Areas, it can be observed that the Software Construction breakdown of topics is not as well organised as taxonomy: for instance, there are clearly duplications. It can also be observed that, on the left-hand side of Figure 2, no distinction is being made between basic concepts and other concepts, all of them being grouped in a single category of first level of decomposition of '*Definitions'*. In the breakdown proposed next in Figure 3, duplication has been eliminated and the breakdown has been restructured on the basis of similarity of concepts. This is referred to as an intermediate revised breakdown.

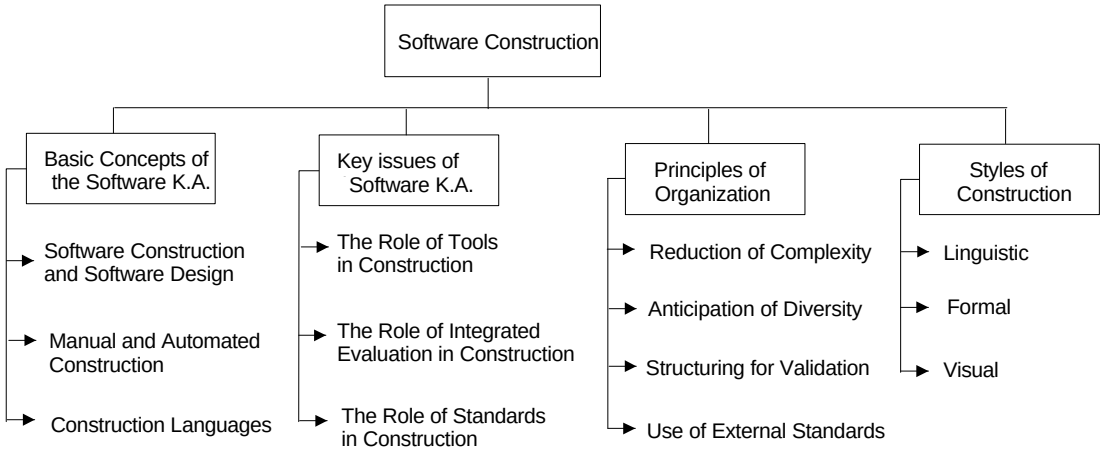

Figure 3: Intermediate revised breakdown

As seen in Figure 3, the previous *synthesis* section has been removed, to eliminate the duplication in the taxonomy. Also, the previous '*Definitions'* topic has been broken down into two sets of related but distinct topics, that is, basic concepts and key issues (as often found in the breakdowns in the other Knowledge Areas). Also, *Construction Languages* and *Programming Languages* have been grouped into a single sub-topic as *Construction Languages*.

On the basis of the insights gained from the analysis of this Knowledge Area from the engineering viewpoint selected, further improvements to the breakdown can now be recommended. The proposed additional changes for a new and improved breakdown of topics are presented in Figure 4. For instance, *Style of Construction,* which does not constitute fundamental knowledge, does not deserve an entire section and we recommend that it be moved to the first level of decomposition of 'basic concepts'. Also, *role of standards* and *use of external standards* should be grouped together, since there are few differences between them in the current SWEBOK text.

Furthermore, we recommend that both *Software Construction Tools* and *Integrated Evaluation* be positioned at the first level of decomposition; neither of these topics are optional when talking about software construction (i.e. coding and generated code) and both should be further defined as covering the three types of construction, linguistic, formal and visual:

- Linguistic tools are mostly compilers.
- Formal tools are mostly case tools.
- Visual tools are mostly graphical and integrated development environments.

Of course, Figure 4 should not be considered the definitive taxonomy on this Knowledge Area and will need further independent review by domain experts.

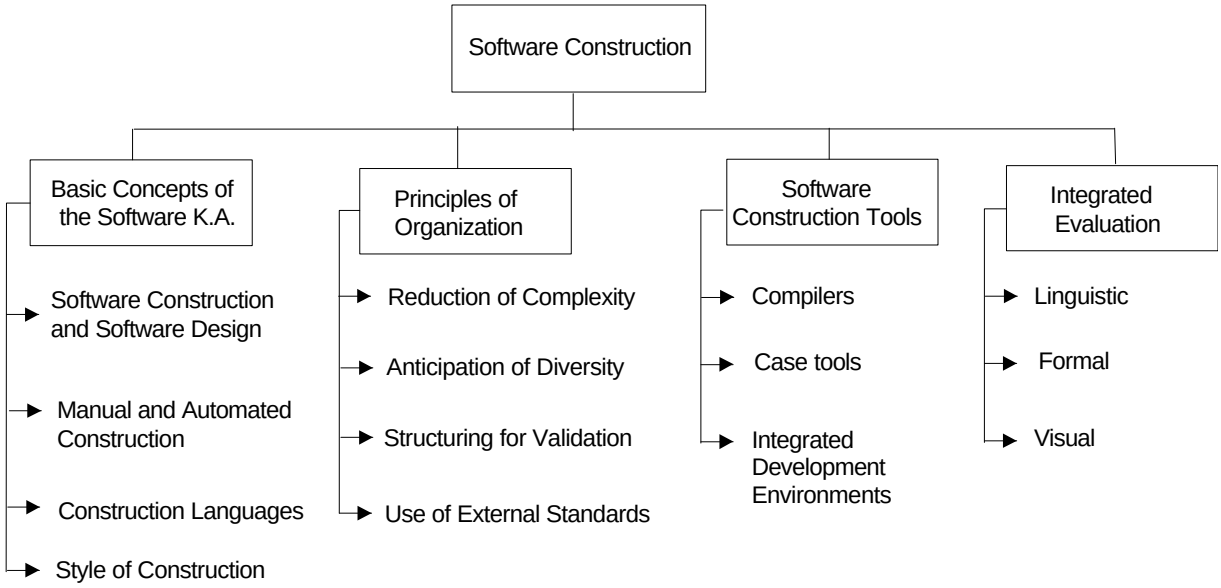

Figure 4: Final revised breakdown

5 Observations for further work

The Software Construction Knowledge Area of the SWEBOK Guide was analysed to address the feedback received from its use at NTU.

The results of this analysis have led to a proposal for a revised breakdown of topics for this Knowledge Area. Such a revised breakdown (Figure 4) is more consistent with the current content of the chapter, with the breakdown structure of the other chapters and also reflects more accurately both the current descriptive content of this Knowledge Area and the quality of their support from an engineering perspective.

Finally, from the insights gained from the use of Vincenti's classification, it can be observed that we have an absence of quantitative data and the quasi absence of theoretical tools. This is, of course, quite surprising for an engineering discipline and it provides a clear indication that much further research work is required for fostering the maturation of software construction as an engineering knowledge area. Similarly, theoretical tools must be developed and extensively validated and quantitative data must be progressively made accessible to the software engineering community.

6 References:

[ABR01] Abran, A., Moore, J.W., Bourque, P., Dupuis, R., Tripp, L. “Guide to the Software Engineering Body of Knowledge – Trial Version”. IEEE Computer Society, Los Alamos, Dec. 2001.

[FRAI02] Frailey, Dennis J. Mason, James. “Using SWEBOK for Education Programs in Industry and Academia”, Conference on Software Engineering Education and Training (CSEE&T), Covington, Feb 2002.

[HUN01] Hunt, Andrew., Thomas, David. “The Pragmatic Programmer – from Journeyman to Master,” Addison-Wesley,1999.

[MAI00] Maibaum, T. "Mathematical Foundations of Software Engineering: A Roadmap," in Proceedings of the Conference on the Future of Software Engineering, Limerick, Ireland, ACM 2000, pp. 161-172.

[VIN90] Vincenti, G. W. “What Engineers Know and How They Know It – Analytical Studies from Aeronautical History,” Johns Hopkins University Press, 1990.