Experimental Support Analysis of the Software Construction Knowledge Area in the SWEBOK Guide (Trial Version 1.0)

Witold Suryn, François Robert, Alain Abran, Pierre Bourque, Roger Champagne
École de Technologie Supérieure
February 20, 2003

Abstract

In May 2001, trial version 0.95 of the Guide to the Software Engineering Body of Knowledge (SWEBOK) [ABR01] was released in a web format and, in December 2001, in book format with the intent to collect comments and possible improvements. Up to now, feedback received confirmed the usefulness of the Guide for all documented knowledge areas, with the exception of the software construction knowledge area, for which the content does not map easily to industry practices or to actual academic curricula.

An initial analysis of this specific SWEBOK chapter enabled us to propose an initial revision of the structure of topics in this knowledge area.

In addition, we conducted a review, presented here, of the chapter to identify the level of experimental support for each topic mentioned in this chapter. In order to classify the level of support, the classification in twelve experimental methods for validating technology by Zelkowitz and Wallace is used. It permits the identification of some of its weaknesses and provides further guidance on content improvements of the chapter.

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; 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) [ABR01] was released in a web format and, in December 2001, it was published in book format. 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 ready for a trial period of two years by its target 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 the trial period will be analysed and will serve as the basis for further improvements to the Guide. 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 actual academic curricula.

The purpose of this study is to analyse, from an engineering perspective, the content of one Knowledge Area of this Guide: Software Construction - chapter 4, with the classification of experimental validation methods by Zelkowitz and Wallace [ZEL01].

The current breakdown of topics in this Knowledge Area is reviewed in section 2, followed in section 3 by a description of the twelve experimental validation methods of Zelkowitz and Wallace. Section 4 presents the analysis of chapter four – Software Construction. A summary concludes the paper.

2 Breakdown of topics

2.1 Current breakdown representation

At the beginning of our analysis, 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. In fact, the content of Figure 1 corresponds only to the text in section 3.3 of chapter 4. It 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. 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.

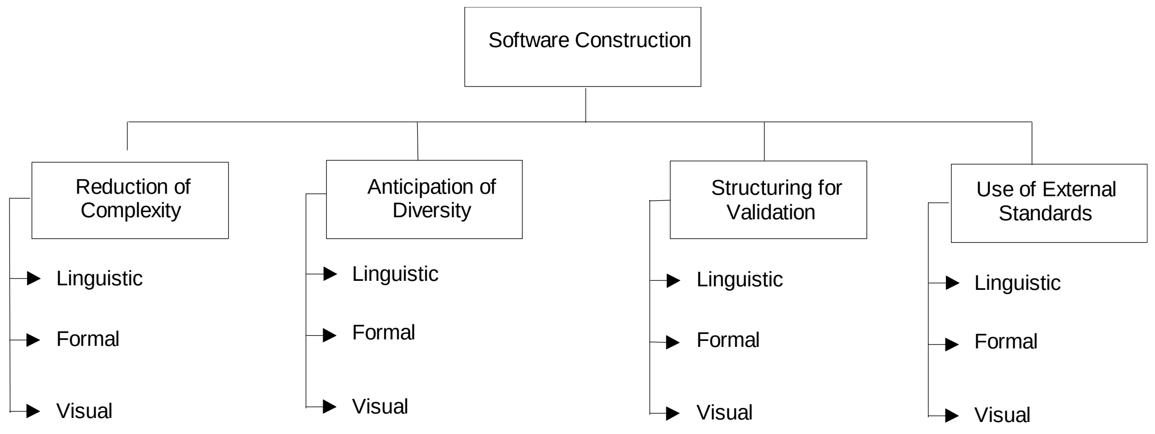

Figure 1: Breakdown of topics as represented in the SWEBOK Guide (trial version 1.0)

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. In later steps, on the basis of our analysis from the two selected viewpoints, we propose further improvements to this initial revision. In "A Technical Review of the Software Construction Knowledge Area in the SWEBOK Guide" [FRO01] we proposed a further set of improvements to this structure, on the basis of Vincenti's classification of engineering knowledge types.

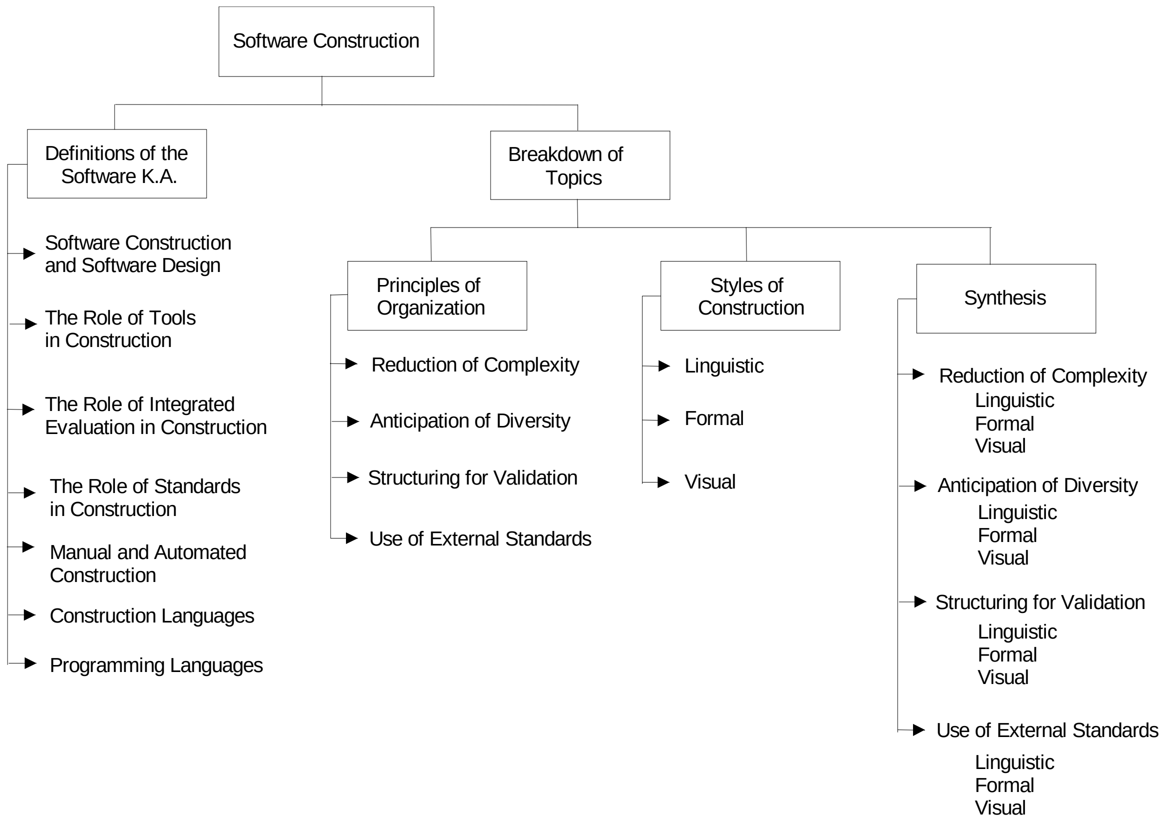
Software Construction
Definitions of the Software K.A.
Breakdown of Topics
Software Construction and Software Design
The Role of Tools in Construction
The Role of Integrated Evaluation in Construction
The Role of Standards in Construction
Manual and Automated Construction
Construction Languages
Programming Languages
Principles of Organization
Reduction of Complexity
Anticipation of Diversity
Structuring for Validation
Use of External Standards
Styles of Construction
Linguistic
Formal
Visual
Synthesis
Reduction of Complexity
Linguistic
Formal
Visual
Anticipation of Diversity
Linguistic
Formal
Visual
Structuring for Validation
Linguistic
Formal
Visual
Use of External Standards
Linguistic
Formal
Visual

Figure 2: Revised breakdown of topics on the basis of the actual text in this SWEBOK chapter

3 Experimental validation method classification

The content of the SWEBOK Guide was written initially by individual authors with expertise in the respective Knowledge Areas, and then widely reviewed by independent experts. While the content of each Knowledge Area is derived from the consensual view of the authors and of the extensive number of reviewers, with the support of references where they could be identified and agreed upon, the whole review process was still based on expert opinion. From an engineering perspective, it is important to know the level of experimental support within each of the respective Knowledge Areas.

Of course, there are multiple types of experimental validation methods, with corresponding strengths and weaknesses. Zelkowitz and Wallace [ZEL01] have identified and classified twelve experimental methods for validating technology *in software engineering.* We will use their classification of experimental methods to characterise the experimental foundation and the validity of software engineering statements made in this chapter, from an engineering perspective. This classification and the twelve validation methods within are:

Observational methods

1. Project monitoring
2. Case study
3. Assertion
4. Field study

Historical Methods

5. Literature search
6. Legacy data
7. Lessons learned
8. Static analysis

Controlled Methods

9. Replicated (e.g. replication of results)
10. Synthetic
11. Dynamic analysis
12. Simulation

Observational methods refer to methods that collect data *during* the development of the project. It includes Project monitoring, Case studies, Assertion and Field studies. *Project monitoring* refers to data collected during development, but with no specific goals. The *case study* is an in-depth monitoring of the project. By contrast to project monitoring, data are collected with specific goals and serve some type of predefined analysis. The *field study* extends the case study by comparing different projects simultaneously. Assertion, on the other hand, refers to the absence of a “true experimentation” in light of “good scientific principles” or the absence of experimentation altogether. An assertion is an affirmation by an expert that only relies on his own experience or with potentially biased experiments done with the goal of proving that his technology is superior, rather then comparing different approaches.

Historical methods refer to all methods based on data collection of *completed* projects. This group includes Literature search, Legacy data, Lessons-learned and Static analysis. *Literature search* refers to the review of results of published papers and other public documents. Legacy data refers to the analysis of data left by a completed project such as source program, specification document, design document, test plan, etc. It is a form of *software archaeology*. *Lessons-learned* refers to lessons-learned documents generally produce after a large industrial project is completed. *Static analysis* refers to structural analysis performed on a completed product. Such analysis includes, for instance, software complexity and data flow analysis.

Controlled methods refer to methods that use multiple instances of an observation in order to provide for statistical validity of the results. This group includes Replicated experiments, Synthetic environment, Dynamic analysis and Simulation. *Replicated experiments* refer to a controlled environment where the same task is performed by different teams or a different task is performed by the same team, or ideally both, in order to provide a statistically valid basis of comparison. *Synthetic environment* refers to experiments done in an artificial setting that mimics a larger system for economical or other reasons. Dynamic analysis refers to methods that use a controlled environment to execute a given product under specific conditions. Rather than reproducing the product at a smaller scale, the complete product is put under specific experimentation by applying externally given conditions. This includes applying scenarios and benchmarking products. Simulation refers to the usage of a model representing the real environment. Contrary to the dynamic analysis, it does not use the real product and contrary to Synthetic environment, it does not use the real environment.

4 Analysis of experimental support

For the classification of the experimental methods supporting each topic of the Software Construction Knowledge Area, each of the references listed in the SWEBOK 'Matrix of Topics vs. Reference Material' (SWEBOK, Chapter 4, section 4) was reviewed and analysed. The experimental methods used for each topic are listed in Table 2. For each specific sub-topic, the assignment of a specific experimental method type was based on the text within the SWEBOK Guide, texts of referred documents (seminal references) and the assessment of the empirical methods used by authors referenced.

4.1 Method used

Knowledge topic	Method used
2.0 Definition	
2.1 Software construction and software design	Assertion
2.2 The role of tools in construction	Field Studies
2.3 The role of integrated evaluation in construction	Assertion
2.4 The role of standards in construction	not applicable
2.5 Manual and automated construction	Assertion
2.6 Construction Langages	Assertion
2.7 Programming Languages	not applicable
3.0 Breakdown	
3.1 Principle of organisation	
3.1.1 Reduction in complexity	Field Studies
3.1.2 Anticipation of diversity	Assertion
3.1.3 Structuring for validation	Case Studies
3.1.4 Use of external standards	not applicable
3.2 Style of construction	
3.2.1 Linguistic	not applicable
3.2.2 Formal	not applicable
3.2.3 Visual	not applicable

Table 2: Types of experimental method support for each knowledge sub-topic

It can be observed that almost every subtopic in Software Construction is based on assertions. This clearly points to a possible lack of validated scientific knowledge in the domain of software construction. The literature review has revealed only three topics that were based on some form of experimental studies.

For the topic *role of tools*, Steve McConnell, in his book “Code Complete” [MCC01], refers to two studies:

- A field study by Barry Boehm, concluding that only twenty (20) percent of tools account for eighty (80) percent of tools usage;
- A survey, reported by Don Reifer, on the effectiveness of CASE tools.

The topic *reduction of complexity* contains many references to studies ranging from the effect of the level of code cohesion on error rate to the ease of modification of modular programs.

In the third topic, *structuring for validation*, many field studies also support the topic. They are mostly, but not exclusively, reported by Steve McConnell in “Code Complete” [MCC01].

5 Summary

The Software Construction Knowledge Area of the SWEBOK Guide was analysed with the classification of experimental methods described by Zelkowitz and Wallace [ZE001].

From the insights gained from the identification of the experimental validation methods supporting each sub-topic discussed, we find that most of the knowledge comes from assertions by experts, and not from structured studies. This clearly points to the need for much stronger and unambiguous empirical evidence to ensure that this Knowledge Area develops progressively into a mature engineering discipline.

This analysis has revealed significant areas for improvement, from an engineering perspective, in this Knowledge Area. Sound and robust experimental methods must replace 'assertions' for most, if not all, of the topics in this Knowledge Area.

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, 2001.

[FRO01] F. Robert, A. Abran, P Bourque, "A Technical Review of the Software Construction Knowledge Area in the SWEBOK Guide ", Software Technology and Engineering Practice Conference (STEP), Montreal, Quebec, October 2002.

[MCC01] McConnell, Steve. “Code Complete – A Practical Handbook of Software Construction.” Microsoft Press, 1993.

[ZEL01] Zelkowitz, Marvin V., Wallace, Dolores R. “Experimental Models for Validating Technology.” IEEE Computer, May 1998, pp. 23-31.