

PREFACE

The *Handbook of Software Engineering and Knowledge Engineering* is the first comprehensive handbook covering these two important areas that have become interwoven in recent years. Many international experts contribute to this Handbook. Each article is written in a way that a practitioner of software engineering and knowledge engineering can easily understand and obtain useful information. Each article covers one topic and can be read independently of other articles, providing both a general survey of the topic and an in-depth exposition of the state of the art. Practitioners will find this Handbook useful when looking for solutions to practical problems in software engineering and knowledge engineering. Researchers in turn can use the Handbook to quickly obtain background information, current trends and the most important references on a certain topic.

The Handbook consists of two volumes. Volume One covers the basic principles and applications of software engineering and knowledge engineering. Volume Two expands the coverage of basic principles and also contains many articles that specifically addresses visual and multimedia software engineering, and emerging topics in software engineering and knowledge engineering such as software patterns, data mining for software knowledge, etc. The two volumes form a complete set, but can be used separately for different purposes.

Turning Knowledge into Software

There is a growing awareness that the central issue in software engineering and knowledge engineering is how to turn knowledge into software. Traditionally software engineering is concerned with the specification, design, coding, testing and maintenance of software. It also implicitly deals with the issues of transforming knowledge into software in the sense that the gathering of knowledge about the problem domain is incorporated into the requirements analysis phase of the software life cycle. Often, informal techniques of knowledge acquisition are used. Thus in the past, the role of knowledge engineering in the software process is an implicit one.

However it has long been recognized by many people that knowledge engineering plays an increasingly important role in software design. Indeed it is because of this conviction that the international conference series on Software Engineering and Knowledge Engineering (SEKE) was founded in 1988, followed by the publication of the *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)* three years later. For both the SEKE conference series and the IJSEKE

journal, the basic viewpoint is that the interdisciplinary area of software engineering and knowledge engineering is concerned with the interplay between software engineering and knowledge engineering — how software engineering can be applied to knowledge engineering, and how knowledge engineering can be applied to software engineering.

This viewpoint should now be modified and expanded because, both in theory and in practice, more and more software engineers and knowledge engineers are explicitly incorporating knowledge into the software process. In editing this two-volume handbook, this expanded viewpoint — that software engineering is concerned with the transformation of knowledge into software — has been carefully taken into consideration to conceptually organize the recent progresses in software engineering and knowledge engineering.

Software Patterns

Let us start with two distinct, yet complementary, viewpoints on software engineering. The two viewpoints may seem completely different, but they are but different ways of viewing the “elephant” that is software engineering.

The first viewpoint, as stated above, is that software engineering is concerned with the transformation of knowledge into software. The second viewpoint is somewhat more technical. It says that software engineering is concerned with the specification, design, transformation, verification and validation of patterns.

Software is nothing but patterns. A program is constructed from some basic patterns, and the construction rules can in turn be expressed as other types of patterns. With grammars, formal languages and automata, there are many approaches to describe the basic patterns and how they are composed into programs.

Specifications are composed of patterns that are the basic building blocks of formal, informal or visual specifications. The specification, in the ideal case, can then be automatically transformed into programs, and verified and validated in the transformational process.

As mentioned above, knowledge used to be described informally, but now there are formal techniques and more precise ways of dealing with knowledge. With advances in object oriented methods, one comes to the inevitable conclusion that knowledge is also composed of patterns. Knowledge is first acquired, then transformed into formal/informal/visual specification, design and finally program.

Therefore, software engineering can now be viewed as the transformation of knowledge into software through the transformation of patterns. The central issue of software engineering is how to turn knowledge into software by means of the creation, composition and transformation of various types of patterns. A key question that can be asked repeatedly for any topic or sub-topic is the following: how to turn what-kind-of knowledge patterns into what-kind-of software patterns?

Overview of Volume One

As mentioned above, the *Handbook of Software Engineering and Knowledge Engineering* is a comprehensive handbook providing readers with both useful overviews and detailed explanations of the methodologies, techniques and current research issues in software engineering and knowledge engineering. Volume One deals with the fundamentals of software engineering and knowledge engineering. Topics relevant to the traditional software life cycle are covered. Current research in software engineering and knowledge engineering are also surveyed.

The first group of articles deal with the basics in software engineering, including such topics as requirements engineering, domain engineering, object technology, software architecture, computer languages, program slicing techniques, incremental software development, technical reviews, formal verification, software maintenance, software reliability engineering, management of inconsistencies in software engineering, software configuration management, reengineering, software measurement, software metrics for identifying critical components in software projects, software engineering standards, engineering access control, usability issues in the software life cycle, software processes in software engineering and knowledge engineering, and message sequence charts in the software engineering process.

The second group of articles deal with the basics in knowledge engineering, including such topics as: conceptual modeling, case based reasoning, logical abduction in software engineering, knowledge-level models of knowledge systems, knowledge discovery and data mining, knowledge based Information access, machine learning for software engineering, and neural networks.

The third group of articles deal with the interplay of software engineering and knowledge engineering, including such topics as: pattern based software reengineering, agent-oriented software engineering, ontologies in software design, rationale management in software engineering, learning software organization, software engineering and learning theory, task models for interactive software systems, software engineering and knowledge engineering issues for web based education systems, and software engineering and knowledge engineering issues in bio-informatics.

Overview of Volume Two

Volume Two expands the coverage of the basic principles of software engineering. However, this volume also addresses many current topics in software engineering and knowledge engineering such as visual patterns, multimedia software engineering, etc.

A central issue is how to turn knowledge patterns into visual software patterns and visual specifications such as UML. Techniques for knowledge acquisition, software visualization and knowledge visualization, are also of great interest.

The first group of articles deal with software engineering, including such topics as: formal description techniques, software specification and design, software inspections, component-based software engineering, versions of program integration, software reuse, assessing different testing strategies for software engineering and

knowledge engineering, automated knowledge-based selection of suitable unit testing techniques, verification and validation, software cost estimation, data model metrics, uncertainty management, software project management, and reverse engineering,

The second group of articles deal with visual and multimedia software engineering, including such topics as: multimedia software engineering, web engineering, object-oriented modeling of multimedia applications, visual languages in software engineering, software engineering for visual programming languages, assessing visual programming languages, visual parallel programming, software visualization, and visualization of knowledge structures.

The third group of articles deal with emerging topics in software engineering and knowledge engineering, including such topics as: software patterns, supporting software processes using knowledge management, methods for knowledge elicitation, knowledge elicitation from software code, nonmonotonic reasoning and consistency management in software engineering, agent-oriented software construction with UML, improving UML designs using automatic design pattern detection, application of knowledge-based systems for supervision and control of machining processes, system-level design notations for embedded systems, situated computing for mobile users, and the synchronization of interactive web documents.

In a rapidly expanding area such as software engineering and knowledge engineering, no handbook can claim to cover all the subjects of interest. However it is hoped that this Handbook is comprehensive enough to serve as a useful and handy guide to both practitioners and researchers for a number of years to come.

Shi-Kuo Chang
*University of Pittsburgh and
Knowledge Systems Institute*