Exploring the Cognitive Foundations of Software Engineering

Yingxu Wang, University of Calgary, Canada
Shushma Patel, London South Bank University, UK

ABSTRACT

It is recognized that software is a unique abstract artifact that does not obey any known physical laws. For software engineering to become a matured engineering discipline like others, it must establish its own theoretical framework and laws, which are perceived to be mainly relied on cognitive informatics and denotational mathematics, supplementing to computing science, information science, and formal linguistics. This paper analyzes the basic properties of software and seeks the cognitive informatics foundations of software engineering. The nature of software is characterized by its informatics, behavioral, mathematical, and cognitive properties. The cognitive informatics foundations of software engineering are explored on the basis of the informatics laws of software and software engineering psychology. A set of fundamental cognitive constraints of software engineering, such as intangibility, complexity, indeterminacy, diversity, polymorphism, inexpressiveness, inexplicit embodiment, and unquantifiable quality measures, is identified. The conservative productivity of software is revealed based on the constraints of human cognitive capacity. [Article copies are available for purchase from InfoSci-on-Demand.com]

Keywords: Cognitive Models; Cognitive Informatics; Denotational Mathematics; Foundations; Informatics Laws; Nature of Software; Programming Psychology; Properties; Software Engineering; Software Science; Software Science

INTRODUCTION

Software engineering is an applied discipline of software science that adopts engineering approaches, such as established methodologies, processes, architectures, measurement, tools, standards, organisation methods, management methods, quality assurance systems and the like, in the development of large-scale software seeking to result in high productivity, low cost, controllable quality, and measurable development schedule (Bauer, 1972; Dijkstra, 1976; Brooks, 1987; McDermid, 1991; Perters and Pedrycz, 2000; Wang, 2007a; Wang and King, 2000). Software Science is a discipline that studies the theoretical framework of software as instructive and behavioral information, which can be embodied and executed by generic computers in
order to create expected system behaviors and
dependent on computational intelligence (Wang, 2007a, 2009a).
The relationship between software science and
software engineering can be described as that
software science is theoretical software engi-
neering; while software engineering is applied
software science.

The object under study in software en-
geineering and software science are software
and program systems, which are a set of
behavioral instructions for implementing a
certain architectural layout of data objects and
for embodying a set of expected behaviors on
a universal computer platform for a required
application. Large-scale software systems are
highly complicated systems that have never
been handled by mankind in engineering disci-
plines. It is recognized that software is a unique
abstract artifact that does not obey any known
physical laws (McDermid, 1991; Hartmanis. 
1994; Wang, 2007a). For software engineering
to become a matured engineering discipline
like others, it must establish its own theoreti-
cal framework and laws, which are perceived
to be mainly relied on cognitive informatics
(Wang, 2002a, 2003a, 2007b) and denotational
mathematics (Wang, 2008a), supplementing
to computing science (Gersting, 1982; Lewis
and Papadimitriou, 1998), information science
(Shannon. 1948; Bell, 1953; Goldman, 1953;
Wang, 2002a, 2003a), and formal linguistics

This paper explores basic properties
of software and cognitive foundations of software
engineering. The nature of software and soft-
ware engineering is explored in the facets of
the informatics, behavioral, and mathematical
properties. The cognitive informatics founda-
tions of software engineering are sought on the
basis of a set of informatics laws of software.
The fundamental cognitive constraints of
software engineering on intangibility, complex-
ity, indeterminacy, diversity, polymorphism,
inexpressiveness, inexplicit embodiment, and
unquantifiable quality measures are elaborated.
Based on the basic research, a set of cognitive
informatics principles for software engineer-
ing is established, such as the conservative
productivity of software constrained by human
cognitive capacity, the cognitive characteristics
of software engineering, software engineering
psychology, the cognitive mechanism of skill
transformation in software engineering, the
cognitive foundations of software quality theories,
and the cognitive complexity of software.

**BASIC PROPERTIES OF
SOFTWARE AND SOFTWARE
ENGINEERING**

The nature of software has been perceived quite
differently in research and practice of computing
and software engineering. Although in the IT
and software industries, software is perceived
broadly as a concrete product, there are three
types of metaphors in perceiving the nature of
software, known as the informatics, mathemat-
ics, and intelligent behavior metaphors. With the
product metaphor, a number of manufacturing
technologies and quality assurance principles
were introduced into software engineering. 
However, the phenomenon, which we are
facing almost the same problems in software
engineering as we dealt with 40 years ago,
indicates a deficiency of the manufacture and
mass production based metaphors on software
and its development. Therefore, the nature of
software and software engineering need to be
systematically investigated.

**The Informatics Properties of
Software**

Information is the third essence in modeling
the natural world supplementing to matter and
energy. According to cognitive informatics
theory (Wang, 2002a, 2003a, 2007b), informa-
tion is any property or attribute of entities in the
natural world that can be abstracted, digitally
represented, and mentally processed. Software
is both behavioral information to designers and
instructive information to computers. With
the informatics metaphor, software may be
perceived as follows.

---

Copyright © 2009, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
Related Content

The Effect of Multi-Parent Recombination on Evolution Strategies for Noisy Objective Functions
Yoshiyuki Matsumura, Kazuhiro Ohkura and Kanji Ueda (2003). *Computational Intelligence in Control* (pp. 264-280).
[www.irma-international.org/chapter/effect-multi-parent-recombination-evolution/6842/](www.irma-international.org/chapter/effect-multi-parent-recombination-evolution/6842/)

Numerical Version of the Non-Uniform Method for Finding Point Estimates of Uncertain Scaling Constants
[www.irma-international.org/chapter/numerical-version-non-uniform-method/76479/](www.irma-international.org/chapter/numerical-version-non-uniform-method/76479/)

Higher Order Neural Network Architectures for Agent-Based Computational Economics and Finance
[www.irma-international.org/chapter/higher-order-neural-network-architectures/56142/](www.irma-international.org/chapter/higher-order-neural-network-architectures/56142/)

An Innovative Model for Detecting Brain Tumors and Glioblastoma Multiforme Disease Patterns

Neural Network Based Classifier Ensembles: A Comparative Analysis
[www.irma-international.org/chapter/neural-network-based-classifier-ensembles/67706/](www.irma-international.org/chapter/neural-network-based-classifier-ensembles/67706/)