Chapter 20

A Theory of Program Comprehension:
Joining Vision Science and Program Comprehension

Yann-Gaël Guéhéneuc
École Polytechnique de Montréal and Université de Montréal, Canada

ABSTRACT

There exists an extensive literature on vision science, on the one hand, and on program comprehension, on the other hand. However, these two domains of research have been so far rather disjoint. Indeed, several cognitive theories have been proposed to explain program comprehension. These theories explain the processes taking place in the software engineers' minds when they understand programs. They explain how software engineers process available information to perform their tasks but not how software engineers acquire this information. Vision science provides explanations on the processes used by people to acquire visual information from their environment. Joining vision science and program comprehension provides a more comprehensive theoretical framework to explain facts on program comprehension, to predict new facts, and to frame experiments. We join theories in vision science and in program comprehension; the resulting theory is consistent with facts on program comprehension and helps in predicting new facts, in devising experiments, and in putting certain program comprehension concepts in perspective.

In theory, there is no difference between theory and practice. But, in practice, there is. —Jan L. A. van de Snepscheut

INTRODUCTION

Joining vision science and program comprehension provides a theoretical framework to explain

how software engineers understand programs and, thus, to explain known facts, to predict new facts, and to set up experiments on program comprehension.

In the recent year, the domain of cognitive informatics as emerged to study the internal processing mechanisms of the human brain and their applications in computing. Cognitive informatics is intrinsically multi-disciplinary and unites researchers in several domain of research such

as cognitive science, cybernetics, and software engineering. In particular in software engineering, it could bring interesting advances that could explain the mechanisms trough which software engineers understand, write, and debug programs. Therefore, it encompasses the domain of program comprehension.

Program comprehension is the domain of software engineering that seeks to explain how software engineers understand programs (Rugaber, S., 1995), how they obtain a mental representation of a program structure and function (Navarro-Prieto, R., 1998). Facts and laws of program comprehension lie at the heart of almost every software related activities, from development to maintenance, deployment, and use. Many studies on program comprehension have been published in the literature. However, understanding program comprehension requires more than just knowing facts; it requires a theory.

Our research concerns situations in which expert software engineers readily recognise well-known patterns (*any* kind of patterns) in a program model, while novice software engineers only perceive their constituents. Figure 1 shows a subset of a class diagram. (We cloud class names for the sake of explanation). The question is: What is the structure and function that an expert software engineer recognises instantly in that diagram and that a novice software engineer does not? (The curious reader may take few seconds to study the class diagram.) An expert software engineer could recognise that the classes follow the solution of the Composite design pattern (Gamma, E., Helm, R., Johnson, R., & Vlissides, J., 1994), thus assigning the function of representing part–whole hierarchies, while a novice software engineer only sees a hierarchy of four classes, some operations, and an aggregation relationship. (Compare with the solution of the Composite design pattern in Figure 4). We attempt to explain this difference in comprehension between expert and novice software engineers with a theory.

Existing cognitive theories[1], such as Brooks' (1978), von Mayrhauser's (1995) Pennington's (1987), Soloway, E., Pinto, J., Letovsky, S., Littman, D., & Lampert, R. (1998), provide explanations on the short-, long-, and working-memories used, on the cognitive internal processes at play, and on the internal and external knowledge incorporated and constructed during program comprehension. Other authors, such as Minsky (1974), Rich and Waters (1990), and Soloway (1986) theorise the cognitive internal representation of knowledge through the concepts of frames, plans, and chunks. However, no existing theory of program comprehension explains well the difference between expert and novice software engineers, in particular wrt. the use of patterns.

Our approach to explain the previous difference recognises the importance of sight during program comprehension and consists in describing the information flow from graphical program models (texts or diagrams) to cognitive internal processes and memories, using vision science. Vision science is an interdisciplinary domain of cognitive science, which provides a framework for understanding vision in terms of phenomena of visual perception, the nature of optical information, and the physiology of the visual nervous system (Palmer, S. E., 1999). It does not "analyse the sociocultural basis of the staggering amount of functional information people learn about familiar" items[2] but focuses "on the more perceptually relevant question of how sighted people manage to perceive an [item]'s functional significance by looking" (Palmer, S. E., 1999). Theories of vision science explain the capabilities of the human visual system to acquire visual information and to provide this information to cognitive internal processes and memories. Therefore, we develop a theory of program comprehension including the processes of acquiring and comprehending information through sight, by drawing inspiration from theories of vision science and of program comprehension. This vision–comprehension theory relates vision processes, cognitive internal

Related Content

Application of Statistical Modelling and Evolutionary Optimization Tools in Resin-Bonded Molding Sand System
Ganesh R. Chate, Manjunath Patel G. C., Mahesh B. Parappagoudarand Anand S. Deshpande (2018). *Handbook of Research on Investigations in Artificial Life Research and Development (pp. 123-152).*
www.irma-international.org/chapter/application-of-statistical-modelling-and-evolutionary-optimization-tools-in-resin-bonded-molding-sand-system/207202

Remote Conversation Support for People with Aphasia
Nilar Aye, Takuro Ito, Fumio Hattori, Kazuhiro Kuwabaraand Kiyoshi Yasuda (2010). *International Journal of Software Science and Computational Intelligence (pp. 72-85).*
www.irma-international.org/article/remote-conversation-support-people-aphasia/39106

Power Pell Sequences, Some Periodic Relations of These Sequences, and a Cryptographic Application With Power Pell Sequences
Çala Çelemolu, Selime Beyza Özçevikand enol Eren (2022). *Applications of Computational Science in Artificial Intelligence (pp. 21-43).*
www.irma-international.org/chapter/power-pell-sequences-some-periodic-relations-of-these-sequences-and-a-cryptographic-application-with-power-pell-sequences/302060

A Comparative Analysis on Economic Load Dispatch Problem Using Soft Computing Techniques
O.V. Singhand M. Singh (2020). *International Journal of Software Science and Computational Intelligence (pp. 50-73).*
www.irma-international.org/article/a-comparative-analysis-on-economic-load-dispatch-problem-using-soft-computing-techniques/252215

On the Cognitive Complexity of Software and its Quantification and Formal Measurement
Yingxu Wang (2009). *International Journal of Software Science and Computational Intelligence (pp. 31-53).*
www.irma-international.org/article/cognitive-complexity-software-its-quantification/2792