TOWARDS PROGRAM UNDERSTANDING SYSTEMS

Tuncer I. Ören
Department of Computer Science, University of Ottawa
Ottawa, On. Canada K1N 6N5

ABSTRACT

Some philosophic and artificial intelligence concepts on understanding are reviewed and fundamental definitions are given. Recent studies in the field are highlighted. Research issues in program understanding are presented in a systematic way.

ÖZET

Program anlama konusunda ilgili felsefe ve yapay zeka kavramları gözden geçirilip temel bazı tarifler verilmiş ve yakın geçmişte yapılmış bazı çalışmalardan bahsedilmiştir. Program anlama konusundaki araştırma konuları sistematik bir biçimde sunulmuştur.

1. INTRODUCTION

Tools to understand computer programs can be very useful in software maintenance and reengineering to satisfy practical goals such as:

1. Automated visualization and documentation, including automated documentation of programs in a natural language.
2. Answering questions about programs. This functionality requires ability to summarize and filter relevant knowledge. (Queries and/or answers can also be in natural language including speech).
3. Automated certification of certain characteristics (or lack of them).
4. Computer-aided verification and validation of programs.
5. Offering criticism about user programs and justifying them as well as providing recommendations to enhance a citicised program. (Including tutoring student programs).
6. Program explanation.

7. Extracting or building reusable components.

A program understanding tool can be a static or dynamic tool. The first type analyzes a program without executing it, while the second type monitors it during its execution.

Different types of knowledge can be discriminated by a program understanding system. These include, for example, discrimination of application domain entities, design decisions, high level abstractions, I/O behavior, rules, plans, policies, and inhereitence relations. In special purpose programs such as simulation programs, knowledge about the elements of static structure of a simulation program such as input, state, output, auxiliary, and interpolated variables, constants, parameters, auxiliary parameters, and tabular functions; the elements of the dynamic structure such as the state transition and output functions; and the experimental conditions can be discriminated.

Böcker et al. (1991, p. 274) reiterate the fundamental difference between the two points of view in computer science. Point of view 1 states that computer science is a formal mathematical discipline, while the second view states that it is an experimental discipline. The consequences of this distinction is very important. View 1 requires that main tools for the development are formal specification techniques, the basic challenge is to ask humans to think clearly and without logical errors, and that the programming methodology would insist on formal verification of specifications before they are converted into programs. The second view, acknowledging the knowledge processing abilities of humans, relies on the development of better tools (Newell and Simon, 1976; Fisher and Böcker, 1983). Program understanding aims to provide such advanced tools and environments.

The aim in this article is to review some philosophic and artificial intelligence concepts on "understanding," provide some definitions in program understanding, and present some studies that the author is involved in the development of some tools for program understanding.

2. PHILOSOPHICAL BACKGROUND

The concept of "understanding" is a fundamental issue in philosophy. See for example, Locke (1984 - original publication: 1690), Leibniz (1985 - original publication: 1765), and Russell (1948). Here, to provide a relevant background, only a few references are cited.

Whitehead distinguishes different types of understanding in chapter 3 of his treatise on Modes of Thought (Whitehead (1968 - original publication: 1938). Paraphrasing him we have the following three definitions.

Internal understanding of a system involves the notion of composition and refers to the system's elements and to their relationships. (Whitehead 1968, pp. 45-46)

External understanding treats the system as a unity and refers to its relationships with its environment. (Whitehead 1968, pp. 45-46).

Logical understanding starts with the details and passes to the construction achieved. (Whitehead 1968, pp. 61).

Some additional notes about understanding are af follows: Understanding is not primarily based on inference. Understanding is self-evidence. Thus inference is used as a means for the attainment of an understanding. Proofs are the tools for the extension of imperfect self-evidence. (Whitehead 1968, pp. 50). "Human understanding requires the adherence to some judicious abstraction, and the development of thought within that abstraction." (Whitehead 1968, pp. 55).

Understanding has two modes of advance: the gathering of detail within assigned pattern, and the discovery of novel pattern with its emphasis on novel detail." (Whitehead 1968, pp. 58).

In chapter 9 of "How we Think," Dewey (1991 - original publication : 1910) covers "Meaning: or Conceptions and Understanding." He clarifies that to say that you do not understand something and that it has no meaning are equivalent. (Dewey, 1991, p. 117).

He distinguishes two types of understanding which are direct and indirect understandings. *Direct understanding* is apprehension. *Indirect understanding*, called comprehension, is mediated understanding. (Dewey, 1991, p. 120).

3. ARTIFICIAL INTELLIGENCE BACKGROUND

Biermann (1990, pp.377-394) gives an example of how a system could "understand" that an object is a chair. Such a system would have a knowledge base about a chair where the knowledge is expressed in terms of a semantic network. Based on the explanation of the system's characteristics, he gives the following definition of understanding:

The understanding of a perception "...with respect to a body of knowledge involves finding a set of self-consistent links between the parts of the knowledge structure and the parts of the perceived data. After such a linkage is made, the intelligent being can follow arcs in its knowledge base to obtain innumerable useful facts, the name of the perceived objects, the names of its many parts,

their relationships to each other, the uses of the object, and all other information available in its knowledge base." (Biermann, 1990, p. 386).

"Reasoning is the process of finding or building a linkage from one entity in memory to another. There must be an initial entity, a target entity, and a way of choosing paths from the initial entity toward the target." (Biermann, 1990, p. 402).

4. SOME PROGRAM UNDERSTANDING SYSTEMS

Early references in program understanding are Basili and Mills (1982), Brooks (1983), and Johnson and Soloway (1983). Some empirical studies in program understanding are due to Letovski (1986) and Wiedenbeck (1986).

Böcker et al. (1986, 1991) stress program visualization in program understanding systems. They have developed tools for the visualization of data structures, control structures, object-oriented formalisms, and directed graphs. One of their systems, KAESTLE is a graphic editor for list strucures in LISP. It can be used to generate and edit the list structures of a LISP program . Another tool, FOOSCAPE, displays the static calling structure as well as dynamic behavior of a programs. ZOO is a knowledge acquisition tool for Smalltalk environment. TRACK is an extension of FOOSCAPE in object-oriented programming. TRISTAN is a tool for the visualization of directed graphs.

Ören et al. concentrated on the application of program understanding to simulation programs and to object-oriented programs written in C++. E/Slam (Elucidation of Slam programs) accepts as input a Slam II program and documents it by a series of statement and program oriented templates. The latter provides summaries of the program according to certain criteria (Ören, 1989; Ören et al. 1990a,b,c, 1992). A recent development is to modularize the natural language documentation process. An analyzer generates knowledge stored in a family of tables. A program generator based on the specification of the types of tables and the desired report generates a customized documentation program which can generate natural language documentations (King et al. 1992). Hamilton (1990) working with the author, implemented a tool, NLC++ (Natural language documentation of C++ programs) which generates english documentation of C++ programs on a PC.

Van Sickle (1992) announced a forthcoming workshop on AI and automated program understanding.

4. SOME CONCEPTS AND DEFINITIONS ON PROGRAM UNDERSTANDING

Program understanding is a model-based and knowledge-intensive activity. It requires an a priori model of the entity under scrutiny. The granularity of the model will limit the granularity of the understanding.

A system **A** can understand a system **B**, if (1) **A** has a model **C**, of **B**; (2) **A** can analyze **B** to find its elements and their relations, and (3) **A** can establish links between the elements as well as relationships of **B** and **C**.

Once the mapping between **B** and **C** is achieved, i.e., once **A** understands **B**, the knowledge of **A** can be used by other knowledge processing modules to perform several knowledge processing activities such as the ones enumerated in the introduction. For example, the following can be achieved:

1. *Visualization* and *documentation* of **B** can be done by providing knowledge about the elements and relationships of **B**. For this purpose, the documentation module can use knowledge collected as the result of the analysis of **B**, knowledge stored in the model **C**, as well as the discrepancies between the knowledge stored in **C** and generated from **B**.

2. A *critique* of **B** can be provided based on the discrepancies between the knowledge stored in **C** and generated from **B**.

3. *Enhancement* of **B** can be achieved by modifying **B** to remedy the perceived deficiencies.

5. RESEARCH ISSUES IN PROGRAM UNDERSTANDING SYSTEMS

Major research issues in program understanding are systematized in Figure 1. They are as follows:

1. Extend the scope of program understanding by identifying new goals for program understanding.
2. For each type of programming (e.g., functional, declarative, object-oriented, structured, or special purpose such as simulation), identify:
   - 2.1 What must be understood (i.e., what are the elements and which of their relations must be understood?)
   - 2.2 What should be the levels of abstractions and granularity of program understanding?

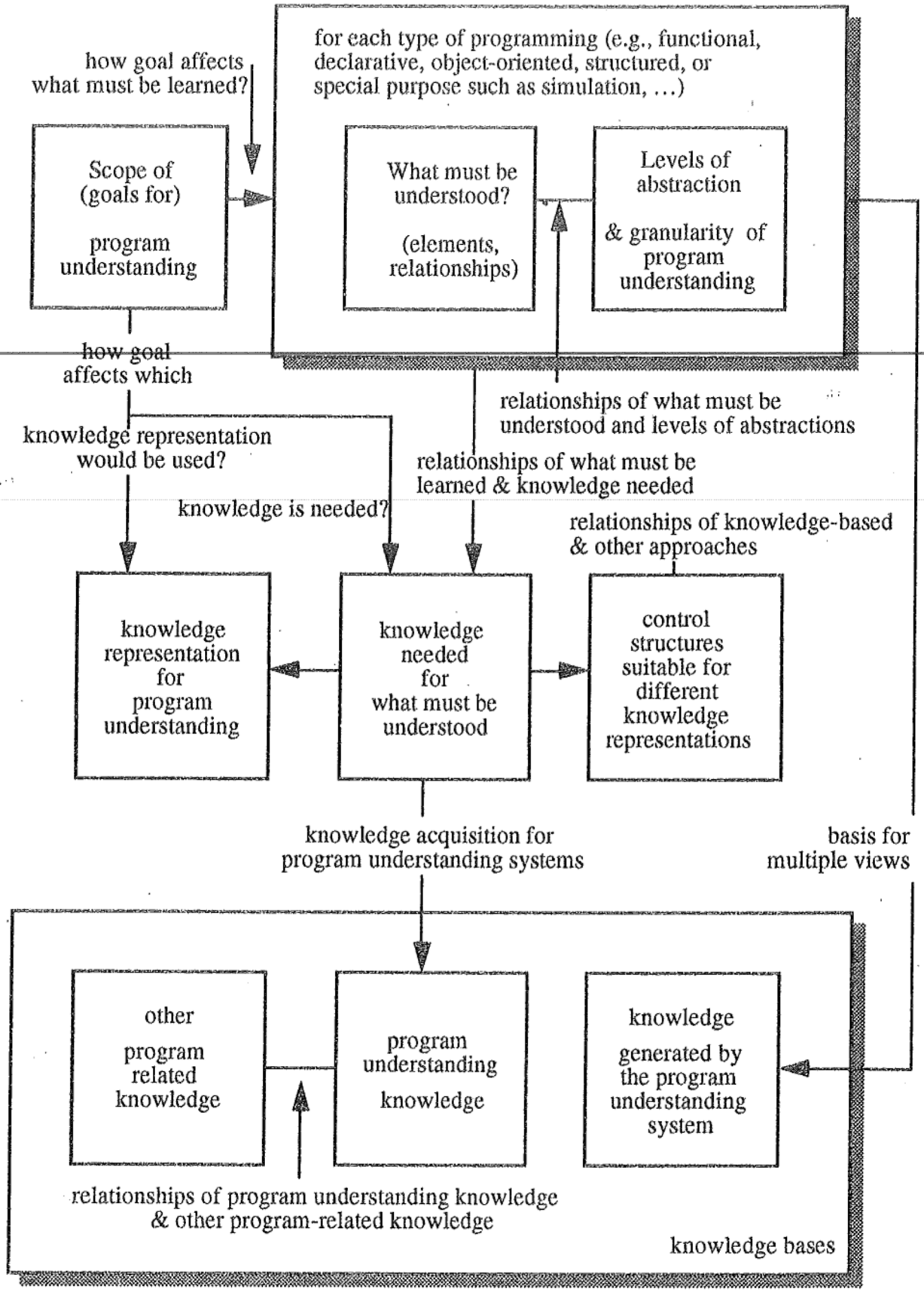

Figure 1. Major research issues in program understanding

3. What knowledge is needed for what must be understood? (i.e., what is the knowledge understanding knowledge?) (Ören 1990).
4. What are the knowledge representation schemes best suited (with respect to which criteria) of the program understanding knowledge?
5. Which control structures are suitable for different knowledge representations?
6. What are the elements of knowledge base(s) for program understanding knowledge?
7. What are the elements of knowledge base(s) for other program-related knowledge?
8. What are the elements of knowledge base(s) to store knowledge generated by program understanding systems? (For increasing the level of abstraction from program level, to user, user group, programming type levels, for example).
9. Determine how goal affects what must be learned?
10. Determine the relationships of what must be learned and levels of abstractions.
11. Determine how goal affects which knowledge is needed for what must be done?
12. Determine how goal affects which knowledge representation would be used?
13. Determine the relationships of what must be understood and levels of abstraction. (Are there any transformation rules between them?)
14. Determine the relationships of knowledge-based and other approaches for the control structures suitable for different knowledge representations.
15. Determine the knowledge acquisition needs of program understanding systems.
16. Determine the relationship of program understanding knowledge and other program-related knowledge.
17. Determine the knowledge base requirements of the knowledge generated by the program understanding system. (Determine the types of knowledge and how they should be processed, for example for associative knowledge processing and for learning as applied to program understanding).
18. Extend the concepts of program understanding to large programs and multi programs.

6. CONCLUSION

Software systems are becoming larger and more and more complex. Their documentation, maintenance, reengineering, and reverse engineering activities necessitate advanced tools. Program understanding tools and environments are very promising new set of software engineering tools and environments for these types of problems. In this article are discussed some basic concepts of understanding and program understanding, some existing systems and a systematic discussion of the needed research.

REFERENCES

Basili, V.R., Mills, H.D. (1982). Understanding and Documenting Programs. IEEE Transactions on Software Engineering, SE-8: 270-283.

Biermann, A.W. (1990). Great Ideas in Computer Science. MIT Press, Cambridge, MA.

Böcker, H.-D., Fischer, G., Nieper-Lemke, H. (1986). The Enhancement of Understanding Through Visual Representations. In: Human Factors in Computing Systems, CHI'86 Conference Proceedings. ACM, New York, 44-50.

Böcker, H.-D., Fischer, G., Nieper-Lemke, H. (1991). The Role of Visual Representations in Understanding Software. In: Artificial Intelligence and Software Engineering, D. Partridge (ed.). Ablex, Norwood, NJ, 273-290.

Brooks, R. (1983). Towards a Theory of the Comprehension of Computer Programs. International Journal of Man-Machine Studies, 18: 543-554.

Dewey, J. (1991). How We Think. Prometheus Books, Buffalo, NY. (Originally published: D.C. Heath, Lexington, MA, 1910).

Fisher, G. (1987). A Critic for LISP. In: Proc. of the 10th IJCAI (International Joint Conference on Artificial Interlligence), J. McDermott (ed.), Morgan Kaufman, Los Altos, CA., 177-184.

Fisher, G., Böcker, H.-D. (1983). The Nature of Design Processes and Computer Systems Can Support Them. In: Integrated Interactive Computing Systems, P. Degano and E. Sandewall (eds.), Proc. of the European Conference on Integrated Interactive Computing Systems (ECICS 82), North-Holland, Amsterdam, 73-88.

Hamilton, I.J. (1990). A Graphic Representation Scheme and a Tool for Natural Language Documentation of C++ Programs. M.Sc. Thesis, Computer Science Department, University of Ottawa, Ottawa, On., Canada.

Johnson, W.L., Soloway, E. (1983). PROUST: Knowledge-Based Program Understanding. In: Proceedings of the 7th International Conference on Software Engineering. IEEE, Orlando, FL.

King, D., Ören, T.I., Hitz, M. (1992 - In Press). Automatic Generation of Natural Language Documentation for SLAM II Programs. In: IMACS Transactions on Scientific Computing '91, E.N. Houstis and J.R. Rice (eds.), North-Holland

Leibniz, G.W. (1985 - original publication: 1765). New Essays on Human Understanding. University Press, Cambridge, England.

Letovsky, S. (1986). Cognitive Processes in Program Comprehension. In: Empirical Studies of Programmers, E. Soloway and S. Iyengar (eds.),Ablex Norwood, NJ, 58-79.

Locke, J. (1984 - original publication 1690 date). An Essay Concerning Human Understanding. Collins, Glasgow, UK.

Newell, A., Simon, H.A. (1976). Computer Science as an Empirical Inquiry: Symbols and Search. CACM 19:3, 113-136.

Ören, T.I. (1989). Program Understanding Systems in Simulation. In: J.W. Rozenblit et al. (eds.) Proc. of the 4th AI and Simulation Workshop, IJCAI-11, Detroit, MI, August 21, 1989, pp. 110-111.

Ören, T.I. (1990). A Paradigm for Artificial Intelligence in Software Engineering. In: Advances in Artificial Intelligence in Software Engineering - Vol. 1, T.I. Ören (Ed.), JAI Press, Greenwich, Connecticut, pp. 1-55.

Ören, T.I., Abou-Rabia, O., King, D.G., Birta, L.G., Wendt, R.N. (1990a) (Plenary Paper). Reverse Engineering in Simulation Program Understanding. In: Problem Solving by Simulation, A. Jávor (Ed.) Proceedings of IMACS European Simulation Meeting, Esztergom, Hungary, August 28-30, 1990, pp. 35-41.

Ören, T.I., Birta, L.G., Abou-Rabia, O., King, D.G., Wendt, R.N. (1990b). E/Slam: A Software Understanding Environment for SLAM II Programs. In: Proceedings of European Simulation Multiconference, Erlangen-Nuremberg, Germany, June 10-13, 1990, SCS International, San Diego, CA, pp. 235-240.

Ören, T.I., Wendt, N.R., Abou-Rabia, O., King, D.G., Birta, L.G. (1990c). A Reverse Engineering Application for Simulation Program Understanding. In: Proceedings of the

Bell Canada Quality Engineering Workshop II, Montréal, PQ, October 4-5, 1990. (Participation by invitation).

Ören, T.I., Wendt, R.N., King, D.G., Abou-Rabia, O., Birta, L.G. (1992 - In Press). A Template-Oriented Approach for Simulation Program Understanding. Mathematics and Computers in Simulation.

Russell, B. (1948). Human Knowledge — Its Scope and Limits. George Allen and Unwin, London, England.

Van Sickle, L. (ed.) (1992 - In Preparation). Proceedings of the AAAI-92 Workshop on AI & Automated Program Understanding, San Jose, CA.

Whitehead, A.N. (1968). Modes of Thought. The Free Press, New York. (Original publication: 1938).

Wiedenbeck, S. (1986). Processes in Computer Program Comprehension. In: Empirical Studies of Programmers, E. Soloway and S. Iyengar (eds.), Ablex Norwood, NJ, 48-57.