Journal of Software Engineering and Applications, 2016, 9, 52-56
Published Online February 2016 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2016.92003

Milestones for Computing Future

Teodor Rus

Department of Computer Science, The University of Iowa, Iowa City, IA, USA
Email: rus@uiowa.edu

Received 19 November 2015; accepted 15 February 2016; published 18 February 2016

Abstract

Computer applications expand continuously as problem solvers in all areas of human life. Yet, computer scientists have difficulties in calling computer science the "science of computer based problem solving methodology". This is probably caused by the development of software technology as a collection of tools dedicated to computer experts. The paper proposes a new methodology for computer based problem solving based on problem domain dedicated software tools. This prevents the exponential growth of software tool complexity and does not require computer user to be a computer expert.

Keywords

Computer, Problem, Problem-Solving, Problem-Domain, Software-Tool

1. Introduction

Vinton Cerf in Communications of the ACM 07/2015, 7 calls for setting anniversary milestones for Computing Future similar to the Hilbert program. At the end of 19 century, mathematicians started doubting whether or not some of the problems they were working on for long time did or did not have solutions. At the 1900 International Conference on Mathematics held in Paris, Hilbert proposed 23 unsolved problems as challenge for mathematicians of the 20 century. Contrasting this situation with today computer science (computing?), computer scientists are challenged to find milestones for future computing. Since computer science has no unanimous accepted definition as a science, one cannot easily identify unsolved computing problems as such challenges. On the other hand, the concept of computing describes an activity that evolves from the very early stages of human society as a mechanism for "problem solving". Of course, originally the problems were very simple, such as counting the number of sheep one had. But with time, computing evolved to the branch of mathematics called the theory of algorithms, computing tools evolved from abacus to today computers, and problem solving methodology evolved to computer programming. In a nutshell, this methodology can be seen as a generalization of Polya [1] four steps methodology for solving mathematical problems. But the concept of a problem in Polya four steps

methodology is not restricted to the mathematical problems. Therefore, when updating it to the computer based problem solving, Polya five steps methodology becomes:

1) Formulate the problem;
2) Develop a solution algorithm;
3) Map the solution algorithm into a program in the language of the computer;
4) Run the program obtained at (3) above on the instance of the problem;
5) Validate the result.

Consequently today computer becomes a tool used in the process of problem solving where problems are generated by all aspects of human endeavor and algorithms are performed by computers. Hence, perhaps the very first challenge for computing anniversary in 2017 would be to settle for an unanimous accepted definition of computing (or computer) science. Generalizing the concept of computing as it evolved to 21 century, the obvious definition would be:

Computer Science Is the Science of Computer-Based Problem Solving of Problems Arising from Any Domain of Human Endeavor

The concepts of a problem and of problem solving involved in this definition refers to usual human-live problems and problem solving, which are larger than the usual understanding of problem and problem solving in mathematics and computer science. Therefore, the problem with this definition is the *contradiction between the universal methodology used by computer-based problem solving process* (*programming*) *and the specific methodology used for problem solving in any individual domain of human endeavor.*

2. Limitations of Current Computer Technology

The contradiction generated by the definition of computer science as the science of computer based problem solving process is currently resolved by developing tools (software tools) that allow computer user to formulate the problem using a problem domain specific methodology while computer computes the solution using computer methodology. These tools rely on mappings of domain specific problem expressions into computer language expressions (programs). Since domain specific problem expressions are natural language expressions and programs are computer based expressions this methodology does not actually resolve the fundamental contradiction of today computer based problem solving process: *the programming*.

Programming (or coding) asks computer users to be computer educated in order to use the computer to solve their problems. The deepening of this contradiction is best illustrated by the growing complexity of the software tools which threatens to "kill computer" [2] as we know it. Due to knowledge spiral, the number and complexity of new computer applications increase exponentially with the successes of computer applications and thus lead to exponential increase in complexity of software tools required by computers-based problem solving process, this increases professional expertise requirements for computer usage putting strong burdens on computer education [3]. Computer research seeks "revolutionary approaches" to overcome this situation but does not challenge the programming as the computer-based problems solving methodology. By the contrary, by "computer democratization" programming is advocated to be moved lower on the scale of student education. This does not simplify software complexity and does not make easier computer usage for non-computer educated people. In other words, computer democratization is a vacuous expression as long as computer usage as a problem solving tool is based on programming. The anxiety created by computer democratization exploded recently under the AI panic of creating "killer machines more dangerous than nukes". Unfortunately this panic is not only promoted by media but it is embraced by computer scientists that have left a mark on computer technology and by Nobel price awarded scientists. And it all started with Turing's idea of creating intelligent machines. But Turing created his machine to help him solve problems. So, by creation Turing machine needed to be better than Turing in the problem solving task he created it to help with. Answering the question whether human can create intelligent machines Turing said "if only scientists could discover a mechanical explanation of how analogical thinking works in the human brain, they could program a computer to do the same". In 1970-s, Hubert Dreyfus identified six-levels of skills during problem solving process: beginner, advanced beginner, competent, proficient, expert, and master, where each represents a "higher level of embodiment" than previous. The embodiment means that skills are encapsulated within human body, not just in the brain, through immersive practice, until they become automatic actions performed with no aware of what they are doing [4]. Hence, it seems practically

impossible for people to describe their action rules they are not aware of.

However, while Turing used his machine as a tool helping his brain solve some given problems, today computer-based problem solving methodology asks the human brain to work as such a machine (computational thinking), which is obviously a gross aberration. Therefore, another obvious milestone would be to *create a computer-based problem solving methodology where the computer is used as a human brain assistant instead of using the human brain as a computer assistant*. Tools created by other human technologies that extend the power of human cognition organs used in the process of skill-embodiment during problem solving process show that this would be a natural step on human evolution. The successes of computer applications such as iPhone and Internet, which do not require their users to be programmers, tell us that yes, such a technology is feasible. Then, why don't we try to find it instead of creating panic with threatens posed by AI? The answer is perhaps in the thinking-inertia created by the successes of human thinking process.

3. A New Methodology

Now, let us try to solve the fundamental contradiction of computer-based problem solving differently. For that let us assume that every problem domain is provided with a specific virtual machine, characteristic to the domain, which can interpret computationally the domain concepts. That means every concept of the problem domain is computationally characterized by its natural language term and the computational meaning of that term. For example, a cell in biology will be characterized by the word cell and a data-model of the biological cell expressed in terms of cell components, which in turn are characterized by the natural language terms denoting them and their computation models expressing their meanings. Then a domain phrase can be seen as a computational model resulted from the composition of the computational models of the terms composing that phrase.

3.1. Domain Algorithmic Language

Following the idea expressed above, a domain expert would use a computer to solve her problems by expressing her solution algorithm in terms of the domain concepts. That is, the domain expert communicates with her machine using the natural language of the domains, called here *Domain Algorithmic Language* (DAL). By normal education, domain experts integrate within their brains concepts and their meanings. Consequently DAL is the fragment of natural language spoken by domain experts. Cell in biology is not confused with the cell in telephony or in automata theory, unless the user is neither a biology expert nor a cell phone user. But in that case the term cell has the meaning of the domain the user belongs to. If she has no domain, then the term makes no sense. The assumption we are making here is that during the learning process provided by normal school education, the domain concepts are stored both in the human brain, as natural language terms (as done by today process of education) and although on a data-carrier specific to the domain, as computer representations of the meanings of natural language terms. The effect of this learning methodology is a process we call Computational Emancipation of Application Domain (CEAD) [5]. CEAD-ing a domain is performed by usual educators collaborating with computer scientists. Domain educators provide the tuple (*term*, *meaning*) and computer experts provide data-carriers and the tools to record domain concepts and the computer expressions of the computational meaning of the terms thus recorded. We refer further to the data-carrier holding domain concepts as the Domain Ontology (DO). Primitive concepts of DAL are manually recorded on the domain ontology. Concepts generated by the learning process in term of other concepts are automatically represented on DO by tools provided by computer scientists [6]. The tools (XML, RDF, URI, SPARLQ, WSDL, etc.) used today to advance Semantic Web can be effectively used for this purpose. That is, DAL of a problem domain grows dynamically during computer-based problem solving process. Computer user uses DAL to express problem solving algorithms while programming is performed by computer domain educated programmers. Therefore in any domain of expertise, during problem solving process, the computer is used as a brain assistant by natural communication, as any other humanly developed tool. The complexity of software tool development and use is factored out from the problem solving process as an activity performed by computer science domain experts. The process of education is no longer complicated by melanging domain concepts with computer concepts.

3.2. New Computer Based Problem Solving Methodology

To solve a problem of the application domain D we follow the Polya's methodology. However, now we assume

that the domain D is computationally emancipated and its domain ontology is DO(D). In addition, the domain D is provided with a virtual machine, whose instructions are the terms denoting concepts of D and the computation denoted by these instructions are stored as Web Services in the ontology DO(D). The Domain Dedicated Virtual Machine (DDVM) is provided with an abstract processor which has an abstract register called Concept Counter (CC) and performs as follows:

```
CC := WebService(DO(D));
while CC is not end
        {
          Execut the Web Service pointed to by CC;
          CC := Next(CC);
        }
        Report the result;
```

where Next(CC) searches DO(D) for the concept CC and performs the computation process thus identified. This mimics the algorithm of program execution performed by a concrete computer. However, the program is a DAL expression, which is a phrase of the natural language of the domain, and the memory holding the instructions and data is the domain ontology, which contains the problem solver knowledge. If the problem domain is "computer programming" then the DAL is a programming language of the computer and the DAL phrase, that is, the program representing the solution algorithm, is stored in the memory of that computer. For a problem domain different from "computer programming" there is no programming step as usual. Rather, to use the computer to solve a problem the solver develops the solution algorithm (which is a phrase in the DAL(D)), makes the CC of the DDVM point to the first concept of the solution algorithm, and the rest is automatically performed by the DDVM(D). We have illustrated this methodology with High School Algebra as the problem domain and DDVM implemented in the Cloud [7]. The new software tools required by this computer based problem solving methodology are:

- A Software Architecture Description Language (SADL) [8]. SADL is universal and was designed as an XML name space whose primitive constructs are tuples <*DALterm*> Web-Service </*DALterm*>.
- SADL interpreter which execute SADL algorithms (programs) on given computer architectures.
- A Translator mapping DAL algorithms into SADL programs.

The computer user develops problem solving algorithms using her DAL and calls the DAL translator to translate her algorithms into SADL programs which are then interpreted by SADL interpreter on computer architectures available in the Cloud.

That is, for each problem domain *D*, software technology implements the DAL(D) in the cloud and allow domain user to subscribes to the cloud for DAL(D). Therefore further we assume that a cloud provider provides its users with CEAD-ed problem domains and virtual machines that interpret problem domain phrases by performing the computations associated with them in the domain ontology using appropriate computer architectures.

This approach of computer-based problem solving methodology shows that there should be no fear of loosing jobs by extensive developments in AI. These developments concern the CEAD-ing problem domains by an activity which effectively democratizes the computer thus showing the entire iceberg, not just its tip, where everybody (from childhood to death) will have the usual natural job of learning. The difference is that learning process now is complimented by recording the knowledge not only in the brain, which is perishable, but also on appropriate data carrier supports by the CEAD-ing of the domain, which is persistent. This activity can be carried out by software tools which create domain dedicated virtual machines that perform the brain work better, because that is why they are created for in the first place. And if during this process one can CEAD the process of human brain transformation of electrical signals into language concepts then, yes, AI will create robots that could behave like human brain. So, yet another computing milestone would be *the development of computer based problem solving methodology where computer is used as a brain tool* (*thinking with computer's help or thinking computationally*) *and the software tools supporting it*. This would allow the computer to become a thinking tool integrated within the cognition process [9].

4. Conclusions and Further Research

Research reported in this paper has been initiated years ago and software evolution during the last 10 years vali-

dates its karma: **to sustain computer evolution we need to develop application domain dedicated software**. Since computer evolves as a brain-tool, we need a computer based problem solving methodology dedicated to problem domain. So far, software technology has been dedicated to the computer itself as the problem domain. Hence, the major benefits of the problem solving methodology discussed in this paper are:

- The new problem solving methodology regards the computer as a brain-tool. In other words, exactly as other human technology tools developed to assist other human cognition organs, such as hands-tools, feet-tools, eye-tools and mouth-tools, it is meant as a human cognition tool as a brain assistant.
- The new methodology is inclusive. It allows computer users to use the computer naturally, as they use any other human tools.
- Since brain is actually the engine of human-cognition process, the evolution of this methodology is endless.
- This methodology provides the mechanism which integrates the computer within the human cognition process. Thus, it evolves the human with her domain of activity. Therefore, it shows that the entire iceberg of computer usage is human life.

However, this methodology needs new experiments based on new domains of application, and therefore it should be considered as an open-ended list of software development projects.

Acknowledgements

I thank JSEA Editor, who helped me fit this paper in the journal pattern, anonymous reviewers, who allowed me to make the paper more readable, and all my previous students, whose works validated the ideas discussed in this paper.

References

[1] Polya, G. (1957) How to Solve It. 2nd Edition, Princeton University Press, Cambridge.

[2] Markoff, J. (2012) Killing the Computer to Save It. *ACM TechNews*, 31 October 2012.

[3] Horn, P. (2001) Autonomic Computing: IBM's Perspective on the State of the Information Technology. http://www.research.ibm.com/autonomic/manifesto

[4] Denning, P. (2015) The Profession of It. *Communications of the ACM*, **58**, 34-36. http://dx.doi.org/10.1145/2804248

[5] Rus, T. (2015) Computer-Based Problem Solving Process. World Scientific, Singapore City. http://dx.doi.org/10.1142/9534

[6] Bui, C.K. (2013) An Evolutional Domain Oriented Approach to Problem Solving Based on Web Service Composition. PhD Thesis, The University of Iowa, Department of Computer Science, Iowa City.

[7] Rus, T. and Bui, C. (2010) Software Development for Non-Expert Computer Users. *Proceedings of the International Conference on Cloud Computing and Virtualization*, Singapore City, 3-5 May 2010, 200-207. http://dx.doi.org/10.5176/978-981-08-5837-7_165

[8] Rus, T. and Curtis, D. (2006) Application Driven Software Development. *Proceedings of International Conference on Software Engineering Advances*, Tahiti, October 2006, 32. http://dx.doi.org/10.1109/icsea.2006.261288

[9] Rus, T. (2013) Computer Integration within Problem Solving Process. *Proceedings, RoEduNet International Conference*, 11th Edition, *NETWORKING in Education and Research*, Sinaia, 17-19 January 2013, 7-12. http://homepage.cs.uiowa.edu/~rus