Future Generation Computer Systems 37 (2014) 417–428

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Hybrid intelligence aspects of programming in *AIDA algorithmic pictures

Yutaka Watanobe [a,*], Nikolay Mirenkov [b]

[a] Graduate Department of Information Systems, University of Aizu, Aizu-Wakamatsu, Fukushima 965-8580, Japan
[b] Kinjito Co., Ltd, Chigasaki, Kanagawa, Japan

HIGHLIGHTS

- Programming in algorithmic pictures is presented.
- Programs in pictures are special information resources.
- Techniques enhancing comprehension of users apply super-characters with annotations.
- Data/knowledge acquisition is based on clusters of different views.
- Confidence of users in making decision is based on the previous decisions of others.

ARTICLE INFO

Article history:
Received 29 December 2012
Received in revised form
27 September 2013
Accepted 13 December 2013
Available online 3 January 2014

Keywords:
Programming in pictures
*AIDA
Algorithmic CyberFilm

ABSTRACT

Programming in algorithmic pictures (***a***-pictures) is an approach where pictures and moving pictures are used as super-characters for representing features of computational algorithms and data structures. Within this approach some "data space structures" are traversed by "fronts of computation" and/or some "units of activity" are traversed by flows of data. There are compound ***a***-pictures to define algorithmic steps (called Algorithmic CyberFrames) and generic ***a***-pictures to define the contents of compound pictures. Compound ***a***-pictures are assembled into special series to represent some algorithmic features. The series are assembled into an Algorithmic CyberFilm. The generic/compound ***a***-pictures and their series are developed and acquired in special galleries of an open type where supportive pictures of embedded clarity annotations are also included. In this paper, *AIDA (Star-AIDA) modeling/programming language (AIDA stands for Animation and Images to Develop Algorithms) and its Filmification modeling (F-modeling) environment are briefly considered and examples of programs in ***a***-pictures are provided. A special attention is paid to *AIDA programs as special information resources which perception, comprehension and cognition depend on interaction with, at least, a few different but mutually supplementing features of ***a***-pictures. A scheme of data/knowledge acquisition based on clusters of different views and how this acquisition is oriented to enhancing user's ability within works on developing application models, corresponding algorithms and programs, are presented.

1. Introduction

The natural intelligence is related, in a great part, with people abilities to perform mental simulation, including thinking/reasoning, problem solving, and decision making [1–3]. However, this simulation depends not only on internal information processing mechanisms of the brain, but also on levels of abstraction connecting the physical world things and corresponding abstract models. Perception and cognition of people are usually much higher if they perform mental manipulations with real objects, and essentially decreased if abstract information objects are involved. Syntax–semantics gap is a fundamental basis for many problems related to the perception and cognition. This gap is predefined by existing systems of symbols, notations, languages, as well as by ambiguous terminologies, trans-disciplinary and cross-cultural contexts. It also depends on the representation of dynamical processes of modeling problems and the adaptive mechanisms of working systems. To bridge the syntax–semantics gap and decrease burden of mental simulation, modern information systems are supported by special data/knowledge acquisitions, ontologies and high-level interfaces based on different mental strategies related to different decision functions (see, for example, [4,5]). These supportive approaches facilitate knowledge sharing and re-use, and are based on a machine-processable semantics of information resources which can be provided to different agents. Though

* Corresponding author. Tel.: +81 242 37 2715; fax: +81 242 37 2753.
E-mail address: yutaka@u-aizu.ac.jp (Y. Watanobe).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.12.031

such agents are promoted to be both software systems and humans, in fact, the latter have serious problems with understanding and use of "formal specifications of a shared conceptualizations". In general, to enhance people ability in understanding meaning of objects and processes and in performing mental simulation, there is a necessity not only to increase the levels of symbols, notations and languages, but also to update our current concept of abstraction. Now abstraction is evaluated as a number of attributes/parameters which can be ignored (covered) for simulating some objects or processes. Instead, it should be evaluated by a number of attributes/parameters which should be preserved in order people can still understand the corresponding model meaning. The current technologies, including processor performance and memory sizes, allow us to take steps for such updating.

The intelligence of software for computational engineering, as for other computational fields, is usually based on a collection of decision support technologies for application researchers and practitioners in creating reliable models and corresponding programs, and in performing relevant experiments and simulations. The aims of the technologies are to enable people for making faster and better decisions, as well as for producing robust solutions. Such technologies are embedded in languages of specification and modeling, in special supportive tools and unified environments, as well as in some systems of data/knowledge acquisition. Roughly, these languages, tools, and systems can be divided into two subsets: one with orientation to make decisions instead of users and another with orientation to help users in making their own decisions (see, for example, [6–10]). In fact, a majority of the technologies include features of both types with some prevalence of one of them.

Our motivation is to develop a modeling/programming environment where users can feel themselves more comfortable in developing new models and corresponding algorithms, and in representing them in forms recognizable and understandable by other users. In this environment, the confidence of users in making decisions is based on access to previous decisions of others, on quick perception and understanding these decisions and on their transformations leading to new models and algorithms. To reach such a goal, special methods for data/knowledge acquisition are implemented. They are based on clusters of different views representing hybrid forms of information resource features. Additional aspects of our motivation are to create such hybrid forms which can make our modeling/programming environment more competitive in expressing details of application algorithms and which can use animation for presenting behavioral features of model constructs.

In this paper we briefly describe *AIDA modeling/programming language employing pictures and moving pictures as algorithmic super-characters (***a***-characters) for representing features of computational algorithms. Special attention is paid to intelligent aspects of *AIDA and its environment with a focus on data/knowledge acquisitions which have influence on user's perception, comprehension, and cognition abilities. We also present a number of techniques which are used with programming in ***a***-pictures for enhancing the abilities of people working on developing application models and corresponding algorithms. These techniques are based on acquiring new super-characters, their compositions within compound constructs and template programs involved, on applying background, foreground and auxiliary images, and on representing scalable forms of model spaces and attributes declarations, dynamical features (animation) of algorithmic steps, and multiple views of an algorithm as a whole.

2. Related works

There is a great variety of works which have some relation and influence on our research activity. However, here we have an opportunity to mention only a selected part of them. First of all, we would like to point new tools related to developing human–computer interfaces for the Internet applications [11], and environments where the concept of putting human needs first is applied and where a necessity for uniting scientific and artistic sides, to obtain a balance between usefulness and attractiveness of technologies, are considered [12].

We would also like to point a paper related to cognitive aspects of software design [13], where authors try to find solutions by considering answers to the following questions: is it possible to bring together theoretically a computer program and a natural language text, and what are differences between programming languages and natural languages. Other papers of our interests are [14,15], where cognitive and computational features of diagrams are considered through another question: why a diagram is (sometimes) worth ten thousand words. The authors point that the fundamental difference between diagrammatic and sentential representations is that the diagrammatic representation preserves explicitly the information about the topological and geometrical relations among the component of the problem, while the sentential representation does not. In addition, we would like to mention approaches related to analogical representations of programs (including a semiotic analysis of comics) [16], self-assembling tilings (for the creation of the desired shapes) [17], various types of information resources annotations [18,19], and the long-term activity of researchers in Certec (Sweden) [20] (where the use of pictures as language for people with disabilities is promoted).

Special points of our attention are ways supporting the application programmers in their decisions by enhancing their perception, cognition, and comprehension abilities. This is the ability to integrate, analyze, and act on large amounts of data from various sources at once [21]. Existing programming environments based on UML (including Executable UML [22–24]) and other diagrams are still not easy with abstract, necessity to possess specific skills for putting together different views, limited number of options and with understanding dynamical processes through static structures [25–27]. In addition, they are not good for detailed programming and textual action languages with appropriate semantics are usually involved. A great part of the programming work should be done in such languages (Alf or just C, Java, C++, etc.). Then what do we really abstracted away and what is difference between programming and modeling? [28]. In addition, they are not so good in displaying "large amounts of data" to be useful for enhancing the above mentioned perception, cognition, and comprehension abilities. Another important thing for us is the software intelligence and forms of data/knowledge acquisitions within the corresponding environments and tools [29]. In conventional programming environment such acquisitions are, in a great part, reduced to adding new items in libraries of components, classes, and/or procedures. These items of black-box types are useful in general, but not so supportive in enhancing "see, think and quickly understand" abilities of the users.

3. Programming in *a*-pictures: a brief overview

As we have mentioned in the introduction, people cognition abilities, in a great part, depend on their abilities to perform mental simulation. However, this simulation depends not only on internal information processing mechanisms of the brain, but also on levels of abstraction connecting the physical world things and corresponding abstract models. To make such simulation more efficient, it is necessary to decrease spending of people energy and time for the recognition of semantics behind unknown terminology, for understanding associations between objects which syntax–semantics forms are based on rote memorization,