2010 2nd International Conference on Education Technology and Computer (ICETC)

Improving an Interactive Simulator for Computer Systems with Learning Objects

S.T. Fung
Dept. of E.E.E.
The University of Hong Kong
Hong Kong
e-mail: stfung@eee.hku.hk

Vincent Tam
Dept. of E.E.E.
The University of Hong Kong
Hong Kong
e-mail: vtam@eee.hku.hk

Edmund Y. Lam
Dept. of E.E.E.
The University of Hong Kong
Hong Kong
e-mail: elam@eee.hku.hk

Abstract— **In the 21st century, learning is a crucial activity through which people can assimilate or acquire new knowledge. However, many existing e-learning systems contain complicated knowledge structure that hinders the reuse or sharing of knowledge. In a previous project awarded by the Microsoft Research Asia, we successfully developed an interactive simulator to facilitate the learning of essential concepts related to computer systems through live animations. Here, we propose to integrate learning objects and relevant technologies into our interactive simulator to illustrate the underlying knowledge structure and, more importantly, facilitate the sharing and reuse of relevant concepts. Through adopting the IEEE learning object metadata (LOM) standard, our simulator can easily exchange relevant learning objects with other e-learning systems. The system design and prototype implementation of our LOM-based simulator is considered in this paper to evaluate how general and experienced users can benefit from our LOM-based simulator in various ways.**

Keywords- learning objects; simulators; knowledge structure; sharing of knowledge.

I. INTRODUCTION

With our careful observation obtained on certain introductory and intermediate-level Computer Systems courses over the past few years (Year 1 and 2 in our three-year undergraduate curriculum), we found that most students encountered difficulty to a certain extent in understanding some essential concepts in computer systems, such as the program execution and the underlying data transfer among the various registers. Intrinsically, these concepts are abstract and often involve a complex knowledge structure, and therefore are difficult to understand. Furthermore, most available simulators for computer systems are text-based and mainly focused on showing the final results after program execution without clearly showing the underlying "operations", and particularly the essential components/concepts involved during such operations. In many cases, students are simply presented with the final result(s) without knowing *how* such result(s) are produced. Undoubtedly, several existing simulators provide a limited set of debugging functions such as monitoring the values of selected registers at a certain step during the program execution. However, without knowing which components, or specifically internal registers, are actually involved in the process, it is totally impossible and meaningless to use such debugging functions for monitoring the changes of values on all the registers so as to better understand the behavior of program execution in the corresponding computer system. In a previous e-learning project awarded by the Microsoft Research Asia (MSRA), we successfully built an interactive simulator, namely the COMPAD [1] as a "learning PAD for COMputer systems", that greatly facilitates the learning of concepts related to computer systems through live animation of program execution on a specific computer architecture.

In this paper, we further enhance our COMPAD simulator by providing users with greater flexibility to interact with the simulator through a repository of learning objects [4, 8, 10, 11]. Accordingly, we have carefully revamped the original design of COMPAD to make it more flexible and scalable to a large repository of learning objects. In addition, we developed an efficient animation engine for general program execution and animation. More importantly, the procedure of modeling new computer architectures is drastically simplified through quick modifications of pre-defined schemas for various computer systems.

As an e-learning system, it is important for our COMPAD simulator system to have access to extensive learning resources of relevant topics in computer systems for learners to assimilate or acquire new knowledge. Therefore, to facilitate the searching and sharing of relevant learning resource, we propose in this paper to integrate the concept of learning objects and relevant technologies into our original COMPAD simulator. By adopting the IEEE Learning Object Metadata (LOM) standard [2], we have built a simple yet efficient learning object management subsystem to construct a local repository of learning objects. With the available learning object management subsystem, course content developers can easily maintain and update the knowledge structure of the underlying subjects by breaking down their original content into learning objects. Clearly, this will help students appreciate the underlying knowledge structure through the arrangement of learning objects in the content materials, and also make the content easily available for sharing between our COMPAD simulator and any other learning content management system (LCMS) [3, 12, 13] that strictly follows the IEEE LOM standard.

This paper is organized as follows. Section 2 reviews the system design of our improved COMPAD simulator while Section 3 details the modifications of the simulation engine. The LOM management system is clearly explained in Section 4. Section 5 provides an empirical evaluation of our implemented prototype. Lastly, concluding remarks are given in Section 6.

II. THE SYSTEM DESIGN OF OUR LOM-BASED SIMULATOR

Basically, the COMPAD simulator [1] consists of two major components. The first major component is the simulation engine which converts any assembly program to the corresponding animation script. The animation script will then be used to generate all the real-time animation events as according to the execution state of the program. The second key component is the LOM management system that allows the insertion, deletion and searching of learning object into our local repository of learning objects about computer systems. Figure 1 shows the system architecture of our enhanced COMPAD simulator.

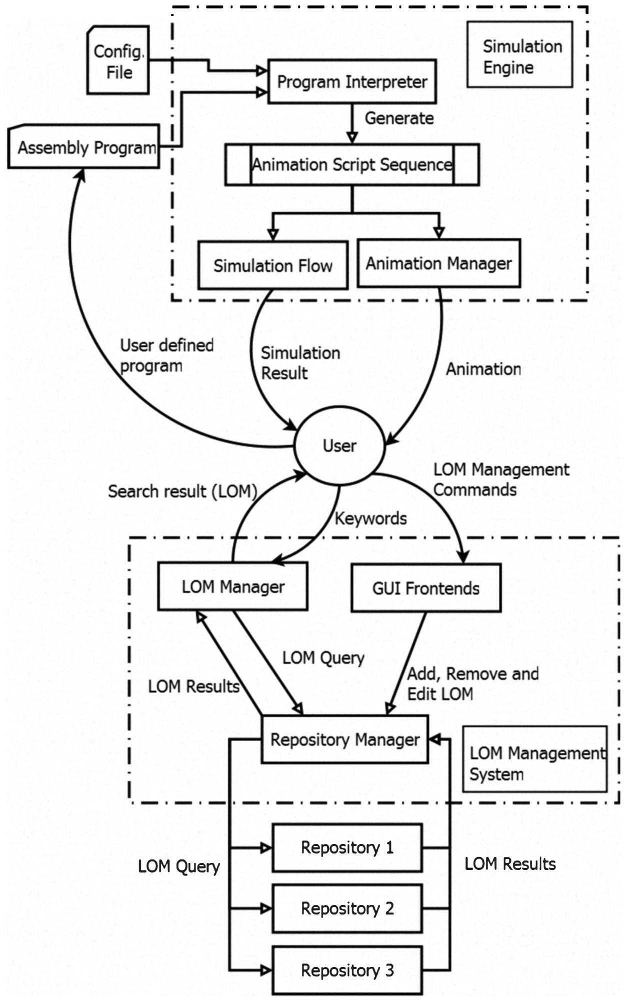


Figure 1. The system architecture of our improved COMPAD simulator.

The system flow of our COMPAD simulator can be explained as follows. The simulator firstly reads in the system architecture configuration file during its start-up process. Then, the user can define the components and data links for the concerned computer architecture. After the user enters an assembly program, the simulation engine will try to interpret the program and convert all the instructions into an animation script as according to the provided configuration files. Lastly, the simulator will use the animation script sequence to generate the live animation for the user to view at his/her own selected speed, ranging from slow, medium to fast. After viewing, the user is free to modify the original assembly program, and run the simulator again to view the latest changes in program behavior. Through these exercises of incremental modifications to an assembly program, students can better understand the operations of the underlying components/concepts involved in program execution in different computer architectures.

All in all, the design of our improved COMPAD simulator emphasizes on two major aspects: flexibility and scalability. First, our enhanced simulator is developed such that it can be easily configured to support simulations of various system architectures with different instruction sets and components. To define a new computer architecture, a user can simply edit some existing configuration files, including the behavior and logic of involved assembly instructions, and save them as a new set of configurations. Through providing different configuration set to the simulation engine, the COMPAD simulator can flexibly generate the live animation for the corresponding computer architecture. In this way, our COMPAD simulator is adaptive to the various computer architectures. Second, for scalability, our learning object metadata (LOM) management system is designed to tackle a potentially large repository of learning objects. It includes a backend database management system to bridge several connected LOM repositories and then performing database queries based on keyword searches. It also includes a graphical user interface for users to interactively add, remove, view or search any LOM as stored in the local repository or repositories of other e-learning systems that are both compatible with the IEEE LOM standard [2] and accessible via the Internet.

III. THE SIMULATION ENGINE

For clarity of presentation, we carefully distinguish the term ***architecture*** and ***model*** for our subsequent discussion. We consistently will use the term architecture to denote any existing computer system architecture such as the Motorola MC68HC11 micro-controller system whereas the term model refers to a user-defined set of hardware components and settings used to display the details for that specific computer architecture. In other words, for the same computer architecture, we may have different models, including a simple or complete model, defined to show the different levels of complexities for the concerned architecture. Therefore, after a user selects a specific configuration file such as that for the Motorola MC68HC11 architecture [5, 6, 9] in our improved COMPAD simulator, the user still needs to specify which model to be used to display the details of the animation. In case the user may want to focus on the core components such as the memory and arithmetic logic unit (ALU) for simple illustration, he/she can adopt the simple model to be used for animation.

Basically, in our enhanced COMPAD simulator, each architecture is defined with a configuration file containing the general information about the specific architecture and all its available instruction sets. Each user can flexibly customize the instruction sets to simulate the behavior of new architectures. The configuration file essentially keeps track of all the static information about the current computer architecture used for simulation. On the other hand, the dynamic information generated to describe the logic and behavior of each inputted assembly program with respect to the selected architecture and model is contained in the animation script. The animation script is generated efficiently in a real-time manner by the animation engine that will be considered in the following subsection.

A. *The Animation Engine*

Intrinsically, the animation script is not simply used for generating live animation. In fact, it is also used to represent all the arithmetic and logical operations involved during the simulation of program behavior for the specific computer architecture. From this perspective, the animation script can be viewed as a set of low-level arithmetic operations involving a number of key components, for which each instruction in the original assembly program will be translated into a sequence of such low-level operations inside the ultimately generated animation script.

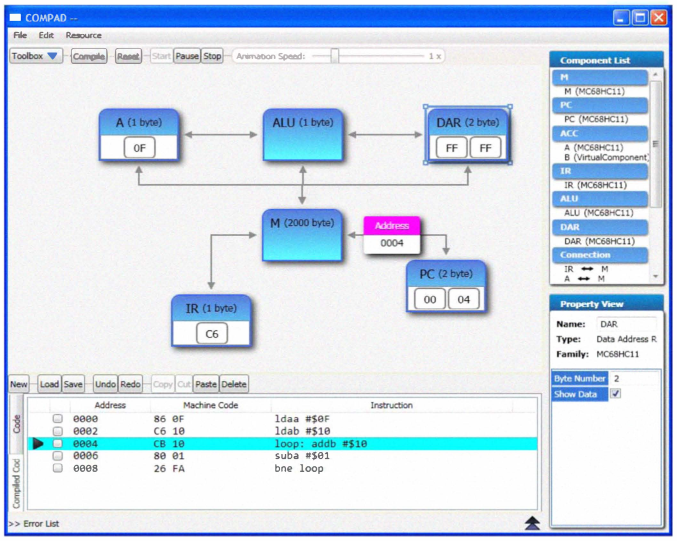
Typically, an animation script can be represented as a sequence of low-level operation specified in the following format:

$$\textit{(Source 1) (Operator) (Source 2)} \rightarrow \textit{(Target)}$$

Both Source *1* and *Source 2* refer to the data operands required for the binary and low-level operation. The *Target* is the destination for the current operation. The *Operator* simply describes the required operation to be performed. Our current implementation of the animation script engine supports most of the common operations such as SUM, SUBTRACT, NOT, AND, OR, XOR, ROTATE, etc. The animation script enables any user to flexibly modify the behavior and logic of the inputted assembly program without the need to modify the original settings of the whole application. Furthermore, the animation script greatly simplifies the process of animation generation since live animation can be dynamically and effectively generated by mapping or adapting various low-level animation operations to a specific animation model of the selected computer architecture.

B. *Defining New Models for Animation*

In our design, a user can define their own model by inserting new components or linking them up by specifying the data links. Our improved COMPAD simulator uses LOM as a container of component data for which each component is a LOM. Accordingly, the user can easily search or link up learning objects related to the corresponding components by completing the "Relation" and/or "Keyword" field of the LOM management system. As the COMPAD simulator also allows the user with the flexibility of defining their own model, conflicts between the selected model and architecture like missing components and connections may arise. However, our simulator will try to resolve this conflict by constructing virtual components and data links which are invisible to user.



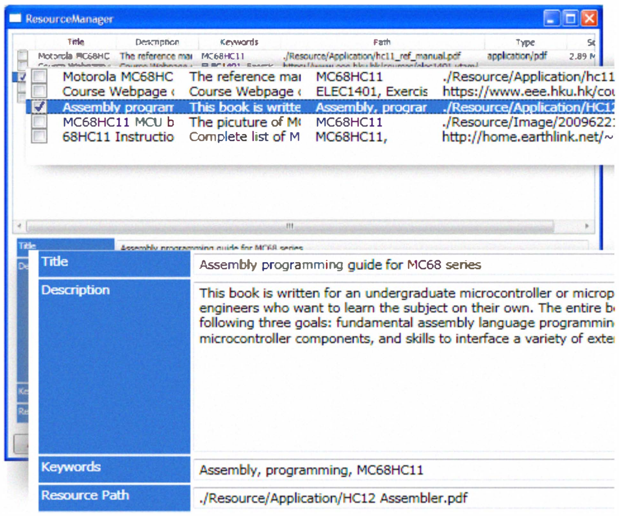
Property View

Name: DAR
Type: Data Address R
Family: MC68HC11

Byte Number: 2
Show Data: ✓

```
Address   Machine Code   Instruction
0000      86 0F          ldaa #$0F
0002      C6 10          ldab #$10
0004      CB 10          loop: addb #$10
0006      80 01          suba #$01
0008      26 FA          bne loop
```

(a) Simulation of an assembly program on the MC68HC11.



	Title	Description	Keywords	Path
	Motorola MC68HC	The reference mai	MC68HC11	./Resource/Application/hc11
	Course Webpage (	Course Webpage (	ELEC1401, Exercis	https://www.eee.hku.hk/cou
✓	Assembly prograr	This book is writte	Assembly, prograr	./Resource/Application/HC1
	MC68HC11 MCU b	The picuture of M	MC68HC11	./Resource/Image/2009622
	68HC11 Instructio	Complete list of M	MC68HC11,	http://home.earthlink.net/~

Title	Assembly programming guide for MC68 series
Description	This book is written for an undergraduate microcontroller or microp engineers who want to learn the subject on their own. The entire b following three goals: fundamental assembly language programmin microcontroller components, and skills to interface a variety of exte
Keywords	Assembly, programming, MC68HC11
Resource Path	./Resource/Application/HC12 Assembler.pdf

(b) Screenshot of the LOM management tool.

Figure 2. The screen captures of our improved COMPAD simulator

IV. THE LOM MANAGEMENT SYSTEM

By following the IEEE-1484 LOM standard [2], the LOM management system of our COMPAD simulator [1] can facilitate the searching of learning object using standardized metadata format. Basically, LOM is used in our simulator to capture explicit knowledge, context, perspectives, and opinions. Learning objects of the form of text, image or video can be imported and searched by keywords. Therefore, each user is free to access, discover and find information using the LOM. In this way, the process of learning and knowledge creation will be significantly enhanced and smoothed.

The LOM management system provides the basic functionality of managing the local LOM database and also an interface for bridging the COMPAD simulator with other online learning content management system (LCMS) [7, 13]. After a user uses any keyword(s) to query the LOM management system, the system will redistribute the specific query to all connected LOM repositories and then respond to the user by displaying the resulting LOM when the search is successful. For efficiency of the search, the implementation of LOM in the COMPAD simulator is simplified since the subjects are mainly restricted to computer system and assembly programming. As most of the LOM search is in the form of keyword(s), the LOM management system will try to match any LOM with the provided keyword(s) at the title, description or keyword field. Accordingly, we only need to implement the corresponding search function on a relatively small subset of fields as defined in the original IEEE-1484 LOM schema, including the title, description, keyword and technical related fields.

V. AN EMPIRICAL EVALUATION OF OUR PROTOTYPE

The prototype of our COMPAD simulator was written in C# using the Microsoft .NET Framework 3.5 and the Windows Presentation Foundation (WPF) graphical subsystem. To demonstrate the feasibility of our proposal on integrating the LOM and relevant technologies into the COMPAD simulator, our current prototype implementation simulates the Motorola MC68HC11 microcontroller [5] using the predefined architecture configuration file and a set of component files. The LOM management system in our COMPAD simulator uses a local LOM repository to store the relevant learning objects and a specific reference file for a clear demonstration.

A. Simulation

Figure 2(a) shows the main window of our improved COMPAD simulator to simulate the execution of a specific assembly program on the MC68HC11 computer architecture. As clearly shown in the concerned interface, a user can edit or create their model by dragging components from the pull-down menu on the left-hand side into the central working space. After formulating the model used for animation, the user can directly enter his/her own assembly program in the code box below the working space. The COMPAD simulator will then verify the assembly program by checking its syntax and any conflict(s) occurred among the involved components for a successful compilation of the program. Upon a successful program compilation, the user can start the simulation process. During simulation, the simulator will generate live animation to illustrate the data flow and essential operations issued by each instruction in the assembly program.

B. The LOM Management System

The LOM management system consists of a LOM manager used to manage the LOM inside its local LOM repository, and a LOM explorer to perform keyword search or browsing through LOM in the repository. Figure 2(b) shows the user interface of the LOM manager used in our COMPAD simulator.

The LOM management system is fully integrated into the COMPAD simulator such that a user can view or search extensive learning objects from its local repository by double-clicking on any component or assembly instruction in the simulator. The LOM management system will automatically generate the appropriate LOM search query according to the nature of the selected component or instruction.

After completing our prototype implementation, we plan to provide our enhanced COMPAD simulator to students enrolled in certain Year-1 and Year-2 compulsory courses on computer systems to try out in both semesters of 2010/2011. By the end of each semester, a set of carefully designed evaluation forms will be distributed to all students attending the courses so as to collect their feedbacks so as to further enhance our system. By then, we would conduct a detailed analysis to evaluate about the effectiveness of our improved COMPAD simulator to facilitate the students' understanding in computer systems through live animation and learning object technologies.

VI. CONCLUDING REMARKS

We successfully integrated the learning object metadata (LOM) [2] and relevant technologies into our interactive COMPAD simulator [1] for users to quickly create and work with related learning objects in the underlying subject area of computer systems. Our enhanced simulator is so generic that users may reuse or modify the information inside the existing learning objects so as to create learning objects in defining new models. This will help to shorten the development time of relevant course or simulation materials. All in all, this paper reports our on-going work that has initiated many interesting directions for future investigation including the uses of sophisticated visualization techniques to guide the systematic structure of learning objects in a specific field, or the integration of an interactive discussion forum to foster the exchange of ideas among students over a peer-to-peer network. More importantly, the pedagogical impacts brought by our LOM-based simulation tool should be thoroughly studied.

ACKNOWLEDGMENT

The authors are grateful to the generous supports from Department of Electrical and Electronic Engineering, the University of Hong Kong. Furthermore, the authors would like to express their gratitude to Professor Yi Shang and Dr. Daniel Churchill for their fruitful discussions on learning objects and e-learning systems.

REFERENCES

[1] J. Yeung, V. Tam, E. Lam, and C. Leung, “Developing an innovative and pen-based simulator to enhance education and research in computer systems,” in Proceedings of the 9th IEEE ICALT 2009. Riga, Latvia: The IEEE Computer Society Press, 2009, pp. 267–269.

[2] “Draft standard for learning object metadata. (ieee 1484.12.1-2002),” The IEEE WG12 Working Group. [Online]. Available at: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final Draft.pdf

[3] “The Moodle System.” [Online]. Available at: http://moodle.org/

[4] F. Porto, A. M. de C. Moura, and F. J. C. da Silva, “Rosa: A repository of objects with semantic access for e-learning,” Database Engineering and Applications Symposium, International, vol. 0, pp. 486–488, 2004.

[5] “The MC68HC11 Reference Manual Rev. 6.1”, Freescale Semiconductor Inc. [Online]. Available at: http://www.freescale.com/files/microcontrollers/doc/ref_manual/M68HC11RM.pdf

[6] R. J. Tocci and F. J. Ambrosio, Microprocessors And Microcomputers: Hardware And Software, 6th ed. Englewood Cliffs, N.J.: Prentice Hall PTR, 2002.

[7] T. Kleinberger and P. Mller, “Content management in web based education,” in Proceedings of the Webnet 2000 Conference on the WWW and Internet, San Antonio, Texas, USA, 2000, pp. 329–334.

[8] J. Vargo, J. Nesbit, K. Belfer, and A. Archambault, “Learning object evaluation: Computer-mediated collaboration and interrater reliability,” International Journal of Computers and Applications, vol. 25, no. 3, pp. 198–205, 2003.

[9] T. Dickens, “Dickens 68hc11 - documentations and references.” [Online]. Available at: http://home.earthlink.net/_tdickens/68hc11/docs_references.html

[10] M. Hatala, G. Richards, T. Eap, and J. Willms, “The interoperability of learning object repositories and services: standards, implementations and lessons learned,” in Proceedings of the 13th International World Wide Web conference, New York, NY, USA, 2004, pp. 19–27.

[11] G. Richards and M. Hatala, “Semantic cobblestones: An interoperability mechanism for learning object repositories,” in McGreal, R.(ed.) Online Education Using Learning Objects, Routledge-Falmer, London, 2004, pp. 301–313.

[12] X. Liu, A. ElSaddik, and N. D. Georganas, “An implementable architecture of an e-learning system,” in Proceedings of CCECE, Montreal, QC, Canada, 2003, pp. 717–720.

[13] S. Rapuano and F. Zoino, “A learning management system including laboratory experiments on measurement instrumentation,”Trans. on Instrumentation and Measurement, vol. 55, no. 5, pp. 1757–1766, Oct 2006.