HASoC for Developing a Software System

Salah Eldin Abdelrahman[1], and Mohammed Badawy[2]

[1] Computer Science and Engineering Dept., Faculty of Electronic Engineering, Menoufia University Menouf, 32952, Egypt

[2] Computer Science and Engineering Dept., Faculty of Electronic Engineering, Menoufia University Menouf, 32952, Egypt

Abstract

We present an object-oriented development of a software system. The development is based on the customization of the lifecycle of a novel developing method called HASoC (Hardware and Software Objects on Chip). The development is presented in order to evaluate HASoC and establish a complete A-to-Z object-oriented teaching example for developing software systems. The evaluation is performed through an object-oriented development process of a software system and is aimed to prove the HASoC practicability and reveal its limitations. The HASoC method was originally aimed for developing embedded systems that are targeted at system-on-a-chip (SoC) implementations.

Keywords: Object-Oriented Approaches, UML, Object-Oriented Database, *Software* Development, Lifecycle Modeling.

1. Introduction

Object-oriented technology offers several specific benefits to software development. Good software engineering stresses modularity, especially in system architecture, detailed design, and implementation phases [1]. Therefore, the benefits of object-orientation are realized throughout the entire development lifecycle, which is typically supported by a set of system models. The system models, which are necessary to communicate the required information that users understand, can be translated into implementation code. object-oriented technology can be consistently applied in the analysis, design, implementation, and test phases of software systems. This means that an object-oriented approach is model-based and covers the development lifecycle in an integrated way.

The basic concept of object-oriented development approaches is that systems should be built from a collection of reusable interacting objects. Various object-oriented methods have been used for systems analysis and design. Therefore, the necessity to standardize these methods has become important. The contemporary modeling language called the Unified Modeling Language (UML) [2] has emerged from the various object-oriented analysis and design notations of these methods in order to meet this necessity.

The developers of UML have attempted to unify past experience of software modeling techniques and incorporate a number of the current software practices into a standard language. UML combines the commonly accepted concepts, notation, and terminology from many object-oriented development methods and selects a clear definition of them. The UML is process, domain, and media independent. It represents the conceptual and physical elements of systems as a set of views, which permit systems to be understood for different purposes. UML views are divided into Structural Classifications, Dynamic Behaviors, and Model Management as well as Extension Mechanisms [3, 4].

According to [5, 6], object-oriented development approaches provide architects, designers, and developers with the conceptual framework, i.e. object model, to attack the rapid increase of systems complexity, and enhance system flexibility. The object model encompasses the principles of abstraction, encapsulation, modularity, hierarchy, typing, concurrency, and persistence, and it provides a conceptual foundation of object-oriented development approaches. Therefore, such approaches offer several potential benefits for system developers and users. Since almost nothing is perfect and without cost, the benefits of object-oriented development are associated with potential limitations, problems, and drawbacks. As [7] has pointed out, most object-oriented development costs are concerned with performance and the difficulties of moving from non-objected-oriented to object-oriented development techniques.

A set of case studies are driven to illustrate and demonstrate the developing stages of object-oriented approaches. As mentioned in [5, 12, 15-18], not only one use case study is driven but also different case studies are driven. Each of such case studies illustrates and

demonstrates one stage of the development process. This leads to miss modeling mapping among the development stages. Therefore, the relation between different system models is not clear. Driving only one case study for all stages of the development process makes the relationship between system's models more clear and the transfer from one model to another straightforward.

This paper firstly aims to illustrate and demonstrate the design approach of the HASoC (Hardware and Software Objects on Chip) method presented in [19] for a software system to evaluate its performance against these systems. The evaluation is performed through the customization of HASoC lifecycle and object-oriented development process. In addition to the evaluation of the novel developing method HASoC for software systems development, a complete A-to-Z object-oriented teaching example for developing such systems is established. We chose a database system as an example of a software system to apply the HASoC method. This software system will be explained later in this paper.

This paper is organized as five sections. In the following section, an overview of the HASoC method as used in this paper is provided. The third section considers and indicates the customization of the HASoC lifecycle to target it specifically at the development of pure software systems. The development process of a software system is demonstrated and illustrated in section four. In the fifth section, the paper is concluded with final remarks.

2. HASoC Overview

HASoC is a method for developing 'complete' embedded systems that are targeted at system-on-chip (SoC) implementations. HASoC aims to develop the entire system not its components that meets functional and non-functional requirements. It supports concurrent development of computer systems software and hardware components. Since the use of contemporary object-oriented analysis and design techniques is helpful for developing embedded systems, HASoC is an object-oriented method. HASoC utilizes object-oriented principles, in general, and the generalized UML-RT profile in particular for system modeling. The generalized UML-RT profile is an extension of the standard UML. This profile includes alternative forms of behavioral description and a new communications mechanism. So, HASoC has a richer behavioral modeling capability than other embedded systems development methods such as POLIS [8], MOOSE [9, 10], and COSY [11].

A set of positive characteristics and development techniques from a number of existing well-defined development approaches have been merged within HASoC. The existence of such techniques is helpful to meet or reduce the effect of many of the challenges that face developers of embedded systems. These challenges include the ever increasing time-to-market pressure, the productivity gap, and the complexity of embedded systems coupled with the multi-disciplinary nature of the development task. Iterative, incremental, and use-case-driven techniques have been merged within the HASoC lifecycle (see Figure 1) to make it responsive to changing system requirements. The system functionality is gathered and specified in a set of use cases that are depicted in a use case model of the system. If one or more requirements of a system are changed, the HASoC lifecycle provides developers with the ability to update the system functionality by editing, adding, or removing one or more use cases. As well as the ability to accommodate changes in system requirements, the adoption of these techniques leads to a simplification of the development task, as it is achieved in a set of iterations rather than a single pass through the lifecycle. The adoption of the use-case driven technique also provides designers with the ability to manage the system development process, and trace requirements.

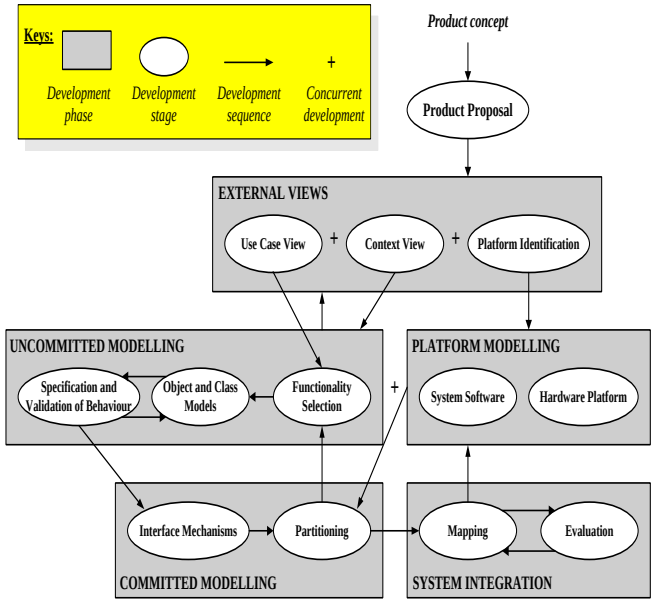

Fig. 1 The HASoC lifecycle.

The platform-based design approach has been integrated into the HASoC lifecycle. This integration, coupled with the above techniques provides a concurrent development facility within HASoC. The application and platform

models of a system can be concurrently constructed and evaluated at the early stages of development. This enables different specialists within the development groups to work in parallel but always in the context of a homogenous model of the complete system. Therefore, the development time of a system may be reduced, which is helpful in reducing the time-to-market of the product. The concurrent development capability of the HASoC method supports the reuse of pre-existing platforms, and the early exploration of system trade-offs. It provides consistency between the development of the application and platform models of a system.

3. HASoC Lifecycle for Software Systems

Although the basic HASoC lifecycle has been identified as shown in Figure 1, it is possible to operate different lifecycles in the context of the method. This could allow developers to adopt UML or HASoC-based modeling whilst still using a variant of their original lifecycle. It also offers the possibility of customizing the lifecycle to suit a particular project or type of project, or even specific application domains. However, we investigate this kind of use of the HASoC method to develop software systems.

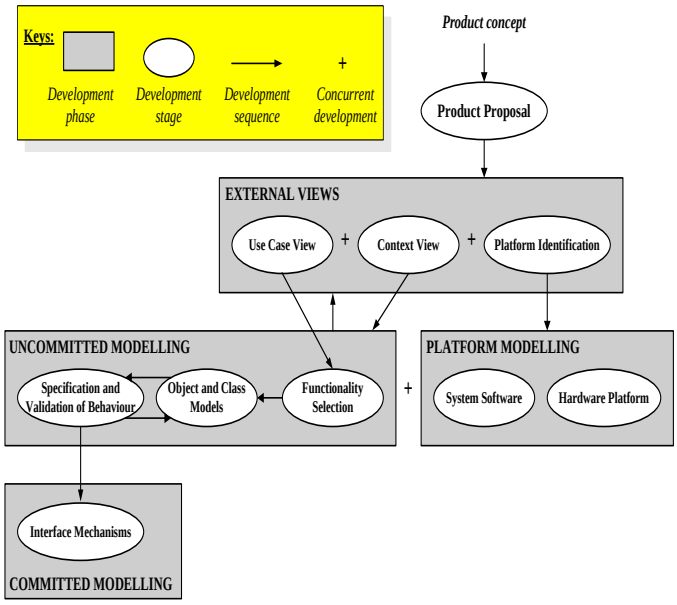

Fig. 2 The customized HASoC lifecycle for software systems.

Some parts of the HASoC lifecycle may be cancelled, in particular, the partition-map-evaluate cycle. Figure 2 illustrate the customization of the HASoC lifecycle for software systems development. This development starts with the basic concept definition of the application and progress through external views and uncommitted stages. We drive the University Registration System (URS) as a case study application.

4. The Software Development Process

The software development process partially follows the HASoC development phases (see Figure 2). These phases include product proposal, external views of the product, and the uncommitted modeling. In the next sections, we explain the steps of each phase in details.

4.1 Product Proposal

The application as a case study in this paper is a University Registration System (URS). The URS has enough features to partially illustrate and demonstrate the design approach adopted in HASoC to develop a software system. It is acknowledged that the development discussed in this paper may not be optimal for a URS. However, it was chosen to illustrate and evaluate as many aspects of the HASoC method.

Students enroll in courses which are offered by instructors. The registrar's office is in charge of maintaining a schedule of courses in a course catalog. They have the authority to add and delete courses and to add schedule changes. They also set enrolment limits on courses (such as, number of students in a course and studying the prerequisite of that course). The financial aid office is in charge of processing student's aid applications for which the students have to apply. The instructors are in charge of setting course offerings, receiving their courses' rosters, and entering the grades of courses they are teaching. Whereas, the students are in charge of being able to register for courses and apply for the financial aid. Assume that we have to design a database that maintains the data about students, Instructors, courses, aid, etc. We also want to design the application that enables us to do the course registration, financial-aid application processing, and maintaining of the university-wide course catalog by the registrar's office.

4.2 External Views of the URS

In the previous phase, the URS requirements have been recorded as a textual report that documents, classifies and clarifies (in an ad hoc way) what is initially known about the system. In this phase, the URS requirements are analyzed and categorized. The external views are developed to express the externally observable behavior that is required of the URS, and its interface with the surrounding environment, see Figure 3.

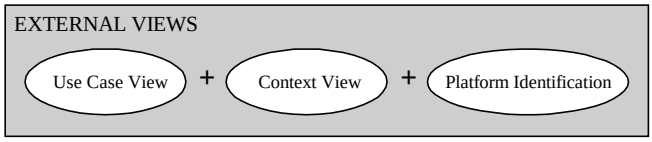

Fig. 3 External views of the HASoC lifecycle

4.2.1 The Context View

The URS Context View aims to state precisely but at a high-level of abstraction, the communications between the URS and its environment. The design and construction of this view are based on the analysis of the above statement that reports the URS requirements. The URS Context View, which is illustrated in Figure 4, depicts the complete URS under development as a single object (the system object) surrounded by external objects (actors) with which it has to communicate or interface. We describe the URS actors as in Table 1 based on an analysis of the written statement of the URS. The actors send a set of messages to the URS, which issues a set of responses. The set of messages passed between the URS and its actors defines the interactions between the URS and its surroundings. Table 2 names and describes the messages and the URS responses.

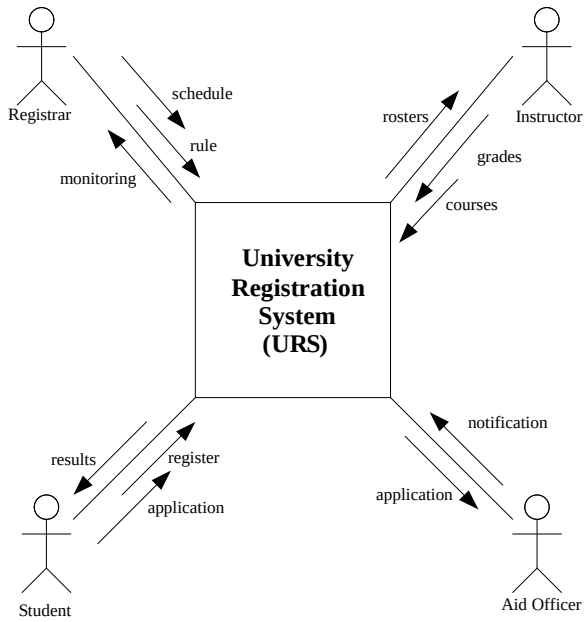

Fig. 4 A URS context diagram

Table 1 The description of the URS actors

Actor	*Description*
Registrar	An employee who maintains the schedule, sets the enrollment limits, and monitors the registration process.
Instructor	A person who offers the courses, receives the rosters, and enters the grades of the students.
Student	A person, who registers for a course, applies for a financial aid.
Aid Officer	An employee who receives the application for an aid and return a notification of acceptance.

Table 2:The description of the external interacting messages of the URS with its actors

Message	*Description*	*The URS response*
schedule	Courses are added or deleted and the schedule is changed by the registrar.	It accepts the commands and updates the catalog.
rules	The registrar sets the enrolment limits on courses.	It accepts or refuses the registrations based on those limits.
monitoring	The registrar monitors the use of the system.	It sends messages about the use of the system to the registrar.
register	The student requests a registration for courses.	It accepts all or part of his/her request based on the rules.
application	The student applies for a financial aid.	It accepts or refuses the aid. It sends the request to the aid officer.
results	Messages returned to the student about his/her registration or aid requests.	It sends a message to the student about accepting or refusing his/her requests.
notification	A notification of acceptance the aid is sent by the aid officer.	It sends the notification to the student.
courses	The instructor offers some courses.	It adds these courses to the catalog.
grades	The instructor enters the grades of students	It saves the grades in the system.
rosters	The instructor receives the rosters of his/her courses.	It sends rosters to the instructors for their courses.

4.2.2 The Use Case View

The Use Case View of the URS establishes the forces that will drive subsequent development stages. It can be developed concurrently with the URS Context View as a UML use case diagram (Figure 5) with the associated use case descriptions (examples are shown in Table 3). It

contains a set of use cases that define the functionality of the URS.

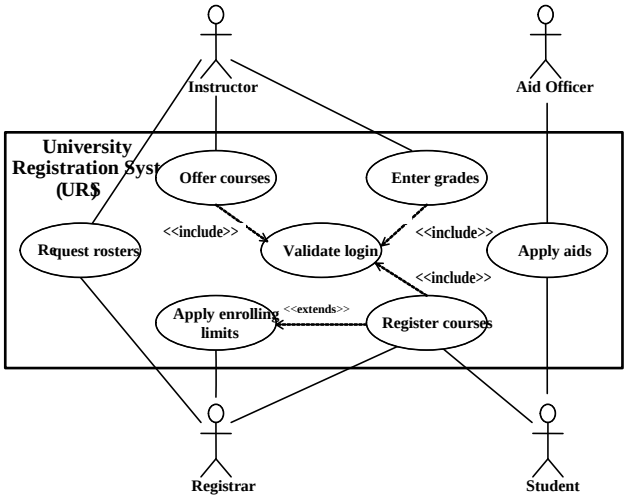

Fig. 5 A use case diagram for a URS

Table 3: Use cases descriptions

Name	Register courses
Summary	The URS receives requests for registration from students.
Actors	Student, Registrar
Description	The students send a registration requests for courses to the system. Registrar monitors the registration process.
Name	Validate login
Summary	The URS has to validate the using of the system.
Actors	Instructor, Student
Description	The instructor and the student must have the authority to use the URS for offering the courses and receiving the rosters in the case of the instructor and enrolling for courses and applying for an aid in the case of the student.
Name	Apply enrolling limits
Summary	The URS ensures the limits of enrolling for courses.
Actors	Registrar, Student
Description	The registrar has to ensure that the enrolling limits are satisfied when students registering for courses.
Name	Request rosters
Summary	The URS receives a request for a roster from the instructor.
Actors	Instructor, Registrar
Description	The instructor requests the names of students who are registered for his/her courses.
Name	Apply aids
Summary	The URS receives the requests for aids from the student.
Actors	Aid Officer, Student
Description	The aid officer receives the requests from the student and sends notifications by an acceptance to him/her.
Name	Offer courses
Summary	The URS receives course offerings from the instructor.
Actors	Instructor
Description	The instructor offers the courses he/she wants to add to the catalog for student registration.
Name	Enter grades
Summary	The URS receives the grades of courses from the instructor.
Actors	Instructor
Description	The instructor enters the grades of courses he/she teaches.

4.2.3 Platform Identification

We assume the initial platform of the URS exists. This at least consists of one microcomputer, and a set of software including a database management system such as Oracle [12, 20], an operating system e.g. Windows, etc. Therefore, based on a UML deployment diagram, the initial version of the URS platform is shown in Figure 6. The basic URS platform may be as that in the initial platform.

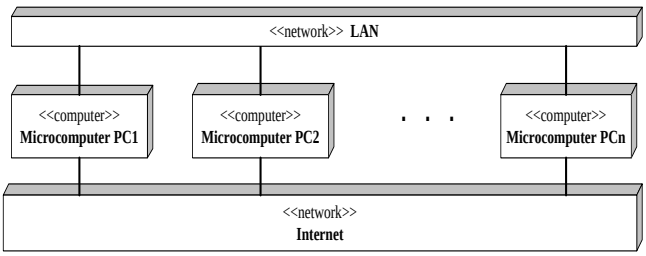

Fig. 6 A deployment diagram for the URS platform

4.3 Uncommitted Modeling: First Iteration

For the sake of simplicity, attention is restricted to the registration process. In other words, the intention is to consider the development of URS registration sub-system. The activities associated with uncommitted modeling are shown in Figure 7. Here, the concern is the development of

the core functionality of the URS, and assume that in the later stages additional functionality will be introduced.

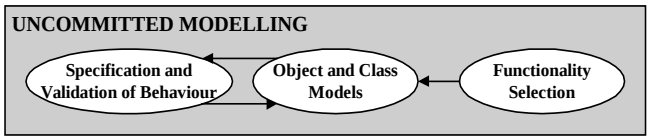

Fig. 7 Uncommitted modeling phase of the HASoC lifecycle

4.3.1 Functionality Selection

Since a typical URS has to register courses for students, the 'Register courses' use case is selected as a core functionality. We select the 'Register courses' use case from the URS use case model and then develop it within the first iteration.

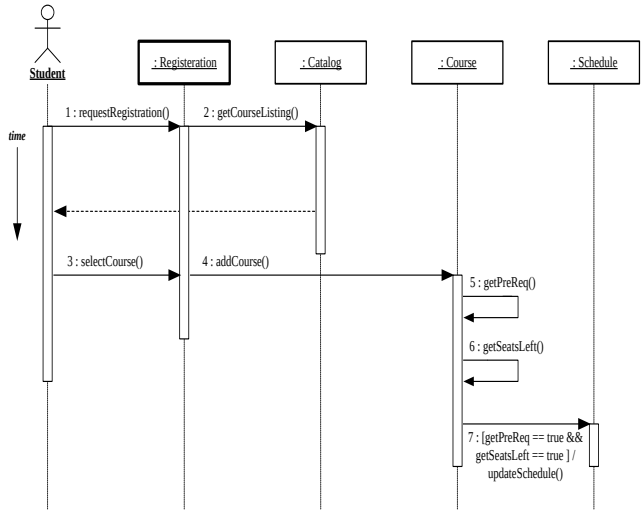

Fig. 8 A sequence diagram of the "Register courses" use case

4.3.2 Object and Class Models

A UML sequence diagram representing the objects and the interactions involved in a scenario that realizes the selected '**Register courses**' use case is depicted in Figure 8. This scenario describes a typical course of events that might occur when the URS registers courses for students. The sequence diagram indicates that the : Registration object is active because it is capable of autonomous activities. Once the student asks for a course, the : Registration object receives such request and performs a number of registration steps. The : Registration object calls the : Catalog object for bringing the course list. Once the student selects the requested course from the list, the : Registration object calls the : Course object for adding the course to the student's courses. Figure 9 illustrates the collaboration diagram.

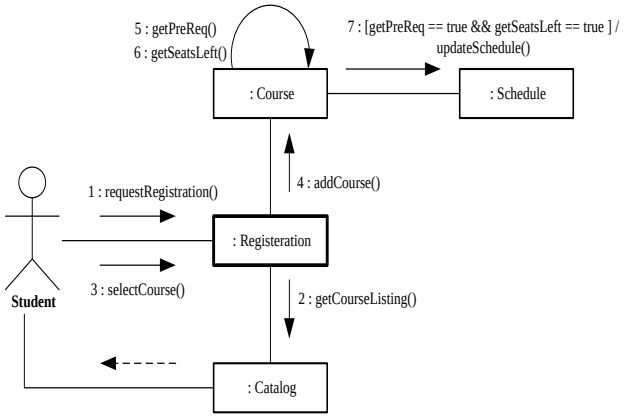

Fig. 9 A collaboration diagram of the "Register Courses" use case

From the above sequence/collaboration diagrams, the corresponding UML class diagram is illustrated in Figure 10. This diagram depicts the static structure of the application-specific part of the URS corresponding to this iteration.

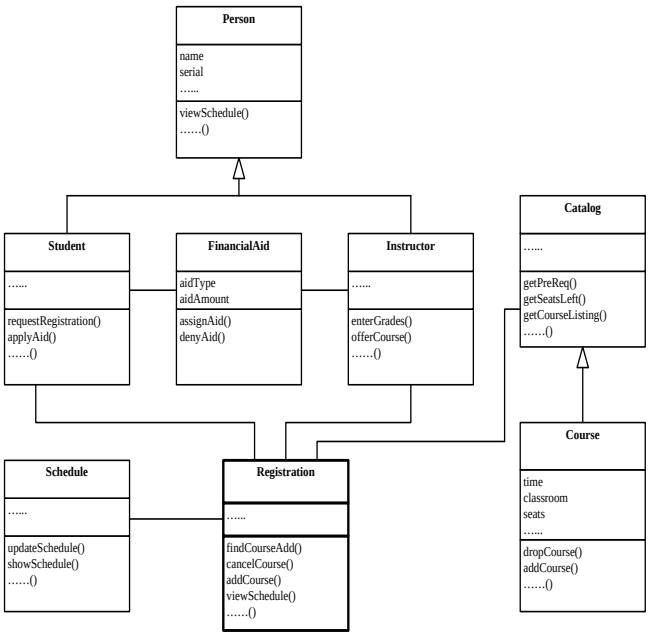

Fig. 10 A UML class diagram of the URS (first iteration)

4.3.3 The Specification and Validation of the URS Behavior

In this stage, the detailed implementation of the classes relevant in the current development iteration is specified. This can be involved in a convenient way through the synthesis of an executable model from the details of object interactions and internal behavior specified in this stage. In other words, model validation can be accomplished by considering the overall behavior of the current increment. The creation of an executable model is briefly considered below. Each development iteration extends the class and object models by introducing functionality into the system specified by the use cases under consideration in the current iteration. Hence, the use case view can be used to provide test specifications for the executable model.

Executable Modeling

The detailed model validation can be achieved through the execution of a full software model. The class model provides important information for synthesis of the executable model and supports the synthesis of interface code. Also, the sequence diagrams provide information about run-time inter-object communications. From these models a skeleton of the executable model can be synthesized. This basic model can be made executable with the addition of supporting generic objects that control the simulation process. Once the model is executable, then the validation of object-level communications can be performed. In order to provide a finer degree of behavioral validation, detailed code would be added to the skeleton executable model, via the addition of code to describe the behavior of passive objects.

4.4 Committed Modeling: First Iteration

The commitment of a system model aims to move the uncommitted model of the URS system towards an implementable specification. Once the executable and initial platform models of the system have been constructed and evaluated, uncommitted objects of the system can be committed. In case of a pure software system, the commitment is achieved as illustrated in Figure 11 through the only Interface Mechanisms development stage.

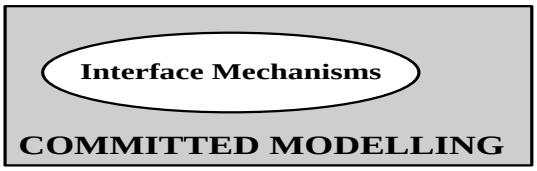

Fig. 11 The committed modeling phase of the modified HASoC lifecycle

Interface Mechanisms

The only stage in the commitment of a software system model considers the external interface of the system. The mechanisms by which the environment interacts with the software system will be fixed at the start of development, whereas in others they must be determined as part of the design process. Clearly, the decision that is taken leads to different sets of interface objects being added to the model. Once the structural and behavioral aspects of the selected 'Register courses' use case have been developed, we move towards the development of an implementable specification. With respect to this use case, there are no additional interface objects that need to be added to the URS model.

4.5 Platform Modeling: First Iteration

The aim of the platform modeling is to describe the overall execution environment in sufficient detail to facilitate the implementation of the complete system, application and platform. Therefore, the platform-modeling phase considers those components that play a supporting role in the operation of the URS. The hardware components including processor, buses, memories, etc. are described through the HAM (Hardware Platform in Figure 12), and the software components, including the operating system, DBMS, etc.

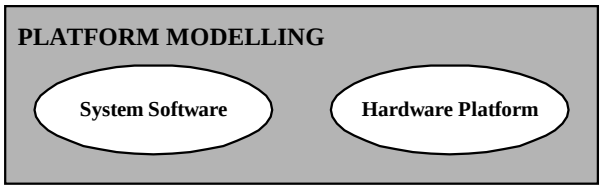

Fig. 12 The platform modeling phase of the modified HASoC lifecycle

At the completion of an iteration of the Platform Identification stage, a model of the implementation of one or more use cases exists in terms of a set of interacting objects. In order to construct a working system, however, further consideration must be given to those components that play a supporting role in the operation of system. These components including processors, buses, memories, DBMS, and an operating system provide the execution environment.

At one end of the spectrum, a pre-existing Hardware Platform may be used in its entirety for the implementation of a system. However, it is more likely that an existing platform would be customized or reconfigured for a particular system through the addition, removal, replacement, or modification of hardware and software IP objects. In each iteration, the capabilities of the platform

could be updated to support the additional functionality introduced into the system in that iteration. If a new platform were to be developed, the HASoC method may be used (in an iterative way) to develop those parts of the platform that are directly used by the application through the introduction of IP cores and supporting software.

4.6 Uncommitted Modeling: Second Iteration

By the same way, we can consider and complete the second iteration of the development of the URS. This development starts by selecting the '**Offer courses**' use case from the use case model and then develop it through the subsequent stages.

5 Conclusion

We have customized the lifecycle of the HASoC method for software systems development, although this method was aimed for developing embedded systems that are targeted at system-on-chip implementations. Not all stages of the HASoC lifecycle are applied for developing software systems. Such development is concerned with the evaluation of the HASoC method and established a complete A-to-Z object-oriented teaching example for developing software systems.

A set of specific benefits have been gained from object-oriented development of software systems using HASoC method. Iterative, incremental, and use-case-driven techniques that have been merged within the HASoC lifecycle make the development of software systems responsive to changing system requirements. The development process has reaped the benefits of using a widely understood notation of UML that has been successfully applied in the development of large-scale software systems. Indeed, the development process has a significant concurrency that makes the underlying platform model of the developed system can commence much earlier with its application model development. Moreover, object oriented development can lead to robust and extendible software systems.

As future work, the following topics deserve more attention as a result of using HASoC method. It is fundamental to investigate, at different aspects, what are the implications and consequences of software systems development. It is important to devise more rigorous supporting software tools for HASoC method, to bring out the benefits of object-oriented mechanisms like inheritance and polymorphism. Applying the HASoC method and demonstrating its perceived strengths in the development of other software systems would allow more solid assessments about the worth and usefulness of this method. We believe that the HASoC is a realistic and feasible method for software systems development. Moreover, the current application areas under consideration include the development of pure hardware systems using HASoC method.

6 References

[1] E. J. Braude, Software Engineering: an Object-Oriented Perspective, John Wiley & Sons Inc., 2001.
[2] http://www.uml.org
[3] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide (The Second Edition), Addison Wesley Object Technology Series, 2005. ISBN: 0321267974.
[4] J. Rumbaugh, I. Jacobson, and G. Booch, The Unified Modelling Language Reference Manual: Addison Wesley, 2004.
[5] G. Booch, Object-Oriented Analysis and Design with Applications, Third Edition: Redwood City, CA, Benjamin/Cummings Publishing Company, Inc., 2007 ISBN: 9780201895513.
[6] G. Booch, Object-Oriented Design with Applications: Redwood City, CA, Benjamin/Cummings Publishing Company, Inc., 1991.
[7] I. Graham, Object-Oriented Methods Principles and Practice, Third Edition: Addison Wesley, 2001.
[8] F. Balarin, E. Sentovich, M. Chiodo, P. Giusto, H. Hsieh, B. Tabbara, A. Jurecska, L. Laavangno, C. Passerone, K.Suzuki, and A. Sangiovanni-Vincentelli, Hardware-Software Co-Design of Embedded Systems: The POLIS Approach (The Springer International Series in Engineering and Computer Science) Kluwer Academic Publishers, 1997.
[9] Computer Systems Engineering Group, "MOOSE: User Manual Version 5.01," http://www.cl.co.umist.ac.uk/moose, February 1, 1999.
[10] Derrick Morris, David Evans, Peter Green, and Colin Theaker "Object Oriented Computer Systems Engineering (Applied Computing)", Springer, 1996 ISBN-13: 978-3540760207.
[11] J.-Y. Brunel, W. Kruijtzer, H. Kenter, F. Petrot, L. Pasquier, E. d. Kock, and W. Smits, "COSY Communication IP's," in Proceedings of Design Automation Conference (DAC), pp. 406-409, 2000.
[12] Ramez Elmasri, and Shamkant Navathe, "Fundamentals of Database Systems (6th Edition)", Addison Wesley, (April 9, 2010) ISBN-13: 978-0136086208
[13] http://www.phindia.com/gupta/
[14] http://www.webhostingsearch.com/articles/oracle-database-management-system.php
[15] Michael R. Blaha and William Premerlani, Object-oriented Modeling and Design for Database Application Using OMT and UML: Prentice Hall 1 edition, 1997.
[16] C.S.R. Prabhu, Object-oriented Database Systems Approaches and Architectures, 2nd edition: Prentice-Hall, 2005

[17] M. O'Docherty, Object-oriented Analysis and Design Understanding System Development Using UML 2.0: John Wiley and Sons, 2005
[18] J. Harrington, Object-oriented Database Design: Morgan-Kaufman, 2000
[19] A. Salah Eldin, "Object-Oriented Technology for System-Level Design," Ph.D. Thesis, University of Manchester Institute of Science and Technology, UMIST, Manchester, U. K, 2003
[20] Oracle Corporation, Oracle 10g Release 2 (10gR2), July 2005.

Salah Eldin Abd Elrahman. got his B. Sc. in Industrial Electronics Engineering May 1988, and M. Sc. in Computer Science and Engineering 9th of July 1994; both degrees from the Faculty of Electronic Engineering, Menoufia University, Egypt. He got his Ph. D. in Computations 11th of July 2003, from the Dept. of Computation, University of Manchester Institution of Science and Technology (UMIST), U. K. From 15th of January, 1989 to 4th of September 1994, he was a demonstrator, from 5th of September 1994 to 22nd of September 1998, as a lecturer, and from 21st of September 2003 to 2005 as an assistant professor at the department of Computer Sciences and Engineering, Faculty of Electronic Engineering, Menoufia University, Egypt. From 2005 to July 2010, he was an assistant professor at the department of Computer Engineering, Faculty of Computers and Information Systems, Taif University, Saudi Arabia and from 27st of September 2010 to August 2011, as an assistant professor at the Faculty of Computers and Information Technology, Tabuk University, Saudi Arabia. he is interested to do some researches in Using Object-Oriented Technology for Developing Embedded Systems, Real-Time Systems, and Software Systems such as Database Systems. The best publication is P. N. Green, M. D. Edwards, and S. E. S. Essa, "HASoC-Towards a New Method for System-on-a-Chip Development," Design Automation for Embedded Systems, vol. 6, No. 4, pp. 333-353, 2002.

Mohammed Badawy received his B.Sc. and M.S. in computer science and engineering at Menoufia University (Egypt) and received his Ph.D. in computer science and engineering at Czech Technical University in Prague (Czech Republic). He worked as assistant professor in the department of Computer Science and Engineering at Menoufia University (Egypt) from 2002 to 2005. He worked as assistant professor in the department of Information Technology at Taif University (Saudi Arabia) from 2005 to 2010 (3 years of them as chairman of the department). Currently, he worked as a consultant in the deanship of Information Technology at Islamic University (Saudi Arabia) from 2010. His research interests includes databases, data stream systems, and software development. He is a member of Association of Computer Science and Information Technology (IACSIT) and a reviewer of the International Journal of Engineering and Technology (IJET). He has published about 13 papers in various scientific journals and refereed conferences.