REAL-TIME SYSTEMS DEVELOPMENT USING OBJECT-ORIENTED REAL-TIME TECHNIQUES (OORT)

BIENVENIDO H. GALANG JR.
Advanced Science and Technology Institute
Quezon City, Philippines
Email: bien@asti.dost.gov.ph

JANICE M. BALLESTEROS, MABETH M. BORRES,
LUCELLE C. BOTARDO, MARGRETTE Q. CACCAM,
AND BILLY S. PUCYUTAN
Advanced Science and Technology Institute
Quezon City, Philippines
Email: bluetooth@asti.dost.gov.ph

ABSTRACT

The Department of Science and Technology-Advanced Science and Technology Institute (DOST-ASTI) is trying to support the Philippine software industry by addressing the lack of original software development. An iterative approach using Object-Oriented Real-time Techniques (OORT) was used entirely during the whole software development process. This paper goes through the Software Engineering Process (SEP), step by step, describing how the Bluetooth™ system was modeled using this approach. The Bluetooth™ system was an aggressive pioneering venture in original systems software development; a first and crucial step in proving that the Philippines is not only capable in software customization but also in advanced software development.

Key Words—Real-time Systems, Object-Oriented Real-time Techniques (OORT), Software Engineering Process (SEP), Bluetooth™

I. INTRODUCTION

The Philippines is one of the top three Asian countries with the greatest software potential. In fact, we are ranked 8th worldwide[1] when it comes to *Knowledge Jobs*. Our skills in IT and in software combined with our capability to read and speak good English make up our strong points, giving us a high ranking in this area. Why then is our overall ranking only 32nd out of the 47 countries surveyed? One of the answers is our lack of *Technology Innovation Capacity*, namely our lack in original software development.

The Philippine Software Industry is currently involved in customizing database and web-based software for foreign clients. Although this provides significant income to our economy, the Intellectual Property (IP) is still owned by the foreign clients. If we want to maximize returns, we must develop original software so that the IP remains here. This is ASTI's[2] strategy; to push for original software development through its Bluetooth™ [3] research efforts. ASTI will pave the way for original applications development of local developers by lowering barriers to entry; i.e., the high cost of the Bluetooth™ protocol stack.

II. BLUETOOTH™

Bluetooth™ is an emerging wireless standard, which aims to interconnect various personal devices in close proximity with each other. It uses short-range radio links to replace the cable(s) connecting portable and/or fixed electronic devices [1]. It is considered to be one of the more profitable technologies in the near future with applications driving the technology to its full potential.

Fig. 1. Components of a Bluetooth™ System

[1] Global New Economy Index (GNEI), Rubin Systems Inc., 2000

[2] DOST-ASTI is the Research and Development Institute of the Philippine government mandated to pursue R&D in the advanced fields of Microelectronics and Information and Communication Technologies.

[3] This project was initially funded by DOST under the RF Microelectronics for Wireless Technologies Project. Currently, it is an independent project entitled "System Software Development for Wireless Mobile Devices" and is fully funded by the Philippine Council for Advanced Science and Technology Research and Development (PCASTRD).

A complete Bluetooth™ solution includes a complete set of protocol software, a set of profiles for various usage scenarios, a networking architecture, software architecture, and hardware architecture as shown in Fig. 1 [1]. This system, including the different components, was modeled using Object-Oriented Real-Time Techniques (OORT).

III. OBJECT-ORIENTED REAL-TIME TECHNIQUES (OORT)

"There is no longer any doubt that Object-Oriented Technology (OOT) is the development solution of the 21st century" [2]. An object-oriented approach helps system analysts and designers integrate the notion of "software durability" into the Software Engineering Process (SEP) and to take into account software reuse at all phases of development because of its powerful concepts and mechanisms. However, this approach is more difficult to design and cannot cope with the following constraints for real-time systems: management of physical resources, multi-tasking, synchronization, data and control management, fault tolerance, and temporal performance [3]. Rather than invent another software engineering technique, the pragmatic approach is to combine OOT with Formal Description Techniques (FDT), each of them fulfilling a clearly defined need [2]. The main focus of FDTs is to ensure the correctness of the final system by the provision of extensive validation and verification methods. OOT on the other hand focuses on an efficient development process with smooth transition between phases [4, 5]. "The overall objective of OORT is to support an iterative process that covers all aspects of system development without any paradigm shift or discontinuity" [2]. It uses an iterative process, which is better than the V-life cycle—the most common SEP approach used in the industry.

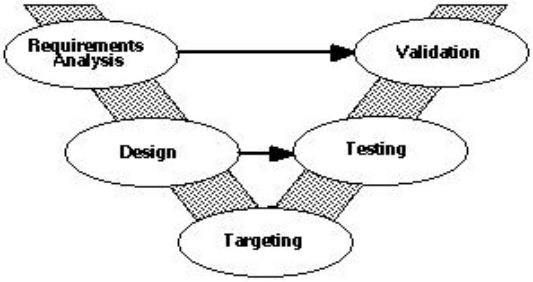

Fig. 2. V-life cycle [2]

Fig. 2 shows the V-life cycle [2]. As shown in the figure, the activities are carried out sequentially. It follows a highly top-down approach and it does not allow reuse of components that are further defined in the later phases of the development process. It can also provide difficulties in shifting from one phase to the next, and it does not support rapid prototyping. A better approach would be an iterative process, which supports software reuse and rapid prototyping as shown in Fig. 3 [2].

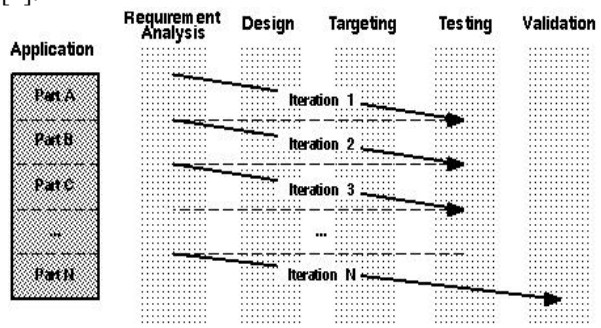

Fig. 3. Iterative Process [2]

Design errors are found and corrected during each iteration and software is reused at all levels. OORT uses the best-adapted notations at each phase of the SEP. It uses OOT in conjunction with FDTs to get the benefit of software reuse, early validation, rapid prototyping and automatic code generation [2, 3]. The transition from one SEP activity to the next is easy using a Computer-Aided Software Engineering (CASE) tool. For the Bluetooth™ project, we used ObjectGEODE as our CASE tool [6].

IV. OORT ENGINEERING PROCESS FOR THE BLUETOOTH™ SYSTEM

The OORT Engineering Process is composed of the different system engineering activities as shown in Fig. 4 [2].

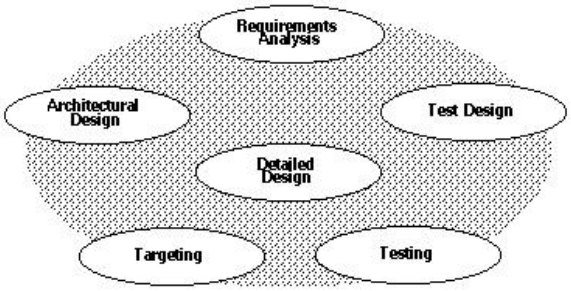

Fig. 4. System Engineering Activities [2]

Each activity makes use of a particular FDT suited for its purpose. It is the combination of these different

FDTs combined with OOT that makes OORT a powerful approach.

Requirement analysis is the initial study of the system that will be developed. It consists of modeling the user requirements and the real-world environment in which the system interacts. The output of *requirement analysis* is a model of the list of actors (objects) related to the system and a model of control flows exchanged between the system and these actors. OORT recommends a combined use of the Object Modeling Technique (OMT) [7] for the actors and the Message Sequence Charts (MSC) [8] for the control flows.

For the Bluetooth™ system, the actors are the different protocol layers, the profiles, and the hardware. OMT was not used because the elements in the system were explicitly known. MSCs were used to describe how the different layers communicated with each other as defined in the Bluetooth™ specifications. These MSCs will also serve as test cases for the Bluetooth™ system during the *testing* and *validation* phases [9].

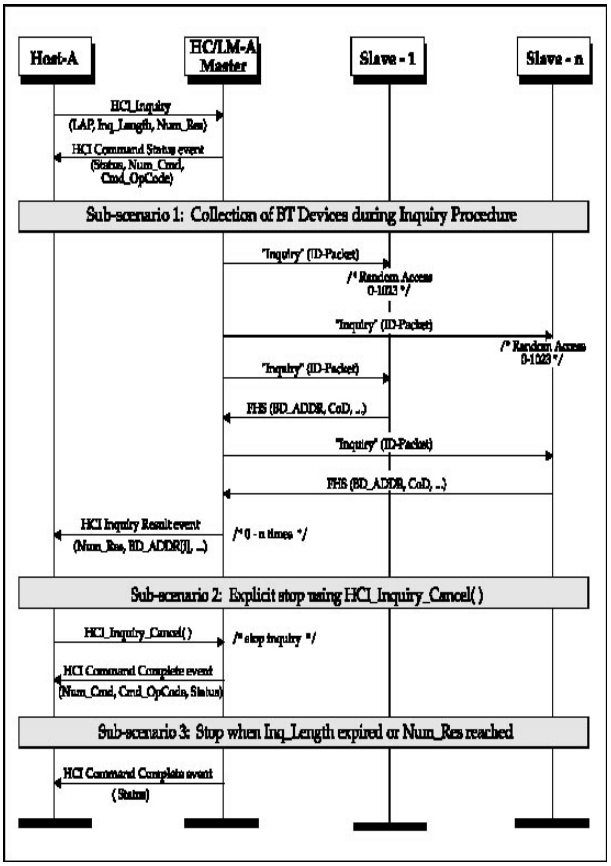

Fig. 5. HCI Inquiry MSC [10]

Fig. 5 shows the MSC for the Host Controller Interface (HCI) Inquiry taken from the Bluetooth™ Specifications book version 1.1. It shows the exchange of signals between devices and between each layer. The parameters passed from one layer to the next are also shown in the MSC. The MSCs derived from the Bluetooth™ Specifications comprise the requirements analysis phase. Since the process is iterative, general MSCs are considered first. Each successive iteration details the MSC for each scenario or use case.

The objective of the *architectural design* activity is to define the logical (software) architecture of the system. The need of architectural design is totally covered by the Specification and Description Language (SDL) notation [11]. A description of the system architecture is made up of the SDL hierarchy diagrams and the set of corresponding interconnection diagrams.

The Bluetooth™ protocol stack was modeled as blocks in the SDL implementation [6]. The Bluetooth™ system was composed of two devices, A and B, with a point-to-point connection as shown in Fig. 6. One device would serve as a client, the other one a server. The applications were modeled as Application Programmer's Interfaces (APIs) communicating with the protocol stack. The physical layer/hardware was modeled as a single communication channel that provides a reliable link between the two devices. The design was simulated and validated by comparing the generated MSCs with the MSCs from the requirements analysis phase and the MSC test cases found in the Bluetooth™ specifications [9].

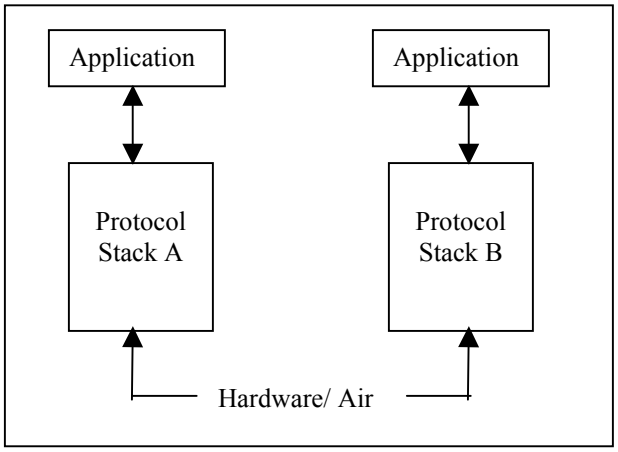

Fig. 6. Bluetooth™ System Block Diagram

Test design allows developers to ensure separate system components behave as intended before integration and it also checks the operation of the software obtained by gradual integration of the implemented components.

Test design is performed in parallel with architectural design to refine the architecture. MSCs are also used in this process.

The Bluetooth™ test cases defined in the specification that were used to specify the use case scenarios in the *requirement analysis* phase, are now used in the *test design* phase. The architecture is tested for compliance with these specifications. Errors are weeded out early in the design process. This is one of the strengths of OORT. It fosters a rapid prototyping environment where the architectural level design can be tested early for design errors.

Table D.1: General operation

Item	Capability	Reference	Status	Support: [Yes] or [No]
1	Use definitions of channel identifier	D, 2.1 D, 2.2	M	◉ ○
2	Support of signal channel	D, 2.2	M	◉ ○
3	Support of connection oriented data channel	D, 2.2	M	◉ ○
4	Support of connectionless reception channel	D, 2.2	O	○ ◉
5	Support of a channel group	D, 2.2	O	○ ◉
6	Support of MTU size larger than DH1 packet payload size	D, 2.4	C.1	○ ◉
7	Support MTU size larger than the largest baseband packet	D, 2.4	O	○ ◉
8	Support of configuration process	D, 6.4	M	◉ ○

Fig. 7. PICS Proforma for L2CAP part D [12]

Bluetooth™ PICS[4] Proforma from the Test Specifications were also used in conjunction with the MSCs. They serve as simple check lists to ensure that the software supports mandatory requirements. Fig. 7 shows a screen shot of the PICS Proforma for the Logical Link Control and Adaptation Protocol (L2CAP) layer.

The architecture is refined during the *detailed design* phase. Behavior is added to each block by defining state machines using SDL. The system is simulated again to verify that the MSCs that it generates comply with the specification. Fig. 8 shows the *Block Level* of the ASTI Bluetooth™ Host-side Protocol Stack. It shows each layer modeled as a block and it also shows the signals passed from one layer to another. State machines describe the behavior of each block. ObjectGEODE provides a function that easily derives a state machine based on an MSC. This makes the transition from *Requirements Analysis* to *Detailed Design* quite easy and it reuses code.

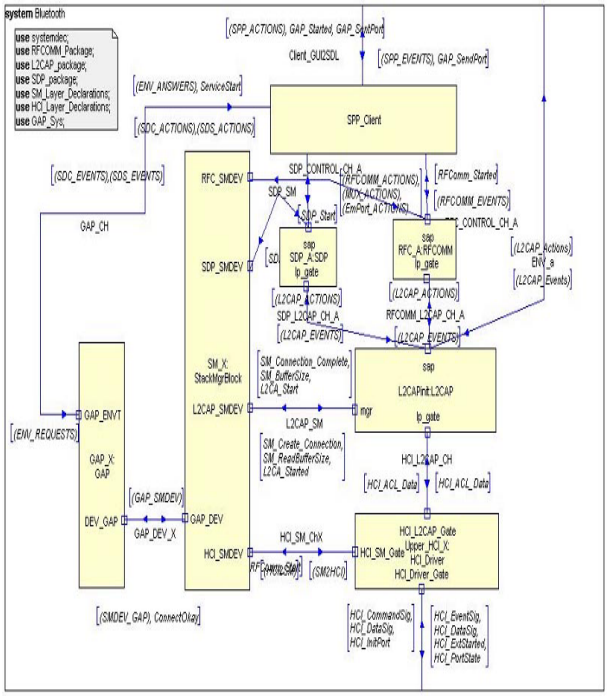

Fig. 8. ASTI Bluetooth™ Host-side Protocol Stack

Targeting, using formal methods throughout the SEP, greatly improves the implementation activity. Formal descriptions can be automatically translated into fully executable codes. The processes in the SDL model of the Bluetooth™ stack are translated into fully executable code by Code Generation. The C code generated is ported to the Win32 platform and it is tested. Real-time errors that were not detected during the simulation can now be corrected. The errors are corrected by changing the model, not the generated code.

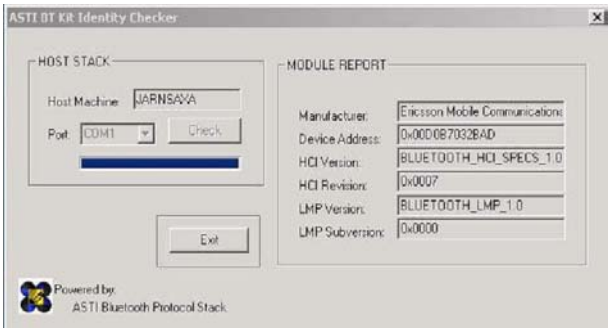

Fig. 9. ASTI BT Kit Identity Checker GUI

Fig. 9 shows a screen shot of the GUI for the ASTI BT Kit Identity Checker, the application on top of the ASTI Bluetooth™ Host-side Protocol Stack. The application was used to verify the correctness of the protocol stack

[4] Protocol Implementation Conformance Statements (PICS) is a detailed list of questions reporting to the Bluetooth™ core/profile specification document.

in a real-time environment. The application runs on Windows 2000.

The system engineering activities, each with its corresponding FDT is shown in detail in Fig. 10 [2]. OMT and MSC are used during the *requirements analysis* phase. SDL is used to model the behavior of the system during the *architectural design* and *detailed design* phases. Abstract Notation One (ASN.1) [13] can be combined with SDL constructs to design the complex packet data units (PDUs) in the *detailed design* phase [14]. MSCs are also used to verify the design during the *test design* phase. Finally, the system is targeted to a real-time environment and it is tested to correct undetected errors in the model during the *targeting* and *testing* phase.

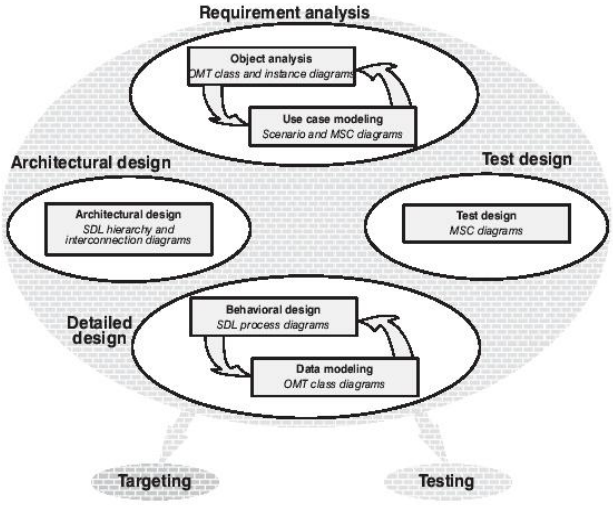

Fig. 10. The OORT Engineering Process [2]

V. CONCLUSION

ASTI was able to successfully develop a Bluetooth™ System using OORT. MSCs were derived from the Bluetooth™ Specifications book version 1.1 during the *Requirements Analysis* phase. They also served as Test Cases during the *Testing* phase. The Bluetooth™ System Architecture was modeled as two devices communicating through a physical link. It was further detailed when each layer was modeled as a block providing services to upper layers and using the services of lower layers. The code was targeted to Windows 2000. The CASE tool, ObjectGEODE, has a code generator that generates C code from the model and targets it to the operating system. An application was written to verify the correctness of the system and it was called the ASTI BT Identity Kit Checker. The software will be targeted to other platforms and applications will be written on top of the ASTI Bluetooth™ Host-side Protocol Stack.

VI. REFERENCES

[1] J. Bray, C. Sturman, *Bluetooth™ Connect Without Cables*, Prentice-Hall, NJ 1999

[2] Verilog, "Generalized Use of the OO technology", ObjectGEODE *Method Guidelines,* Verilog SA, 1996

[3] M. E. Fayad, et al. *"Object-oriented Real-Time Systems Analysis and Design Issues"*. http://csce.unl.edu/~fayad/publications/columns/p105-fayad.pdf

[4] E. Holz. *"Towards an Application of FDT within OOAD"*.http://www.informatic.hu-berlin.de/~holtz/Literatur/FBT.pdf

[5] T. Jeron, et al. *"Validation and Test Generation for Object-oriented Distributed Software"*. Proceedings of PDSE '98. IEEE. Kyoto Japan. http://www.irisa.fr/triskell/publis/1998/Jeron98a.pdf

[6] A. M. Caccam, M. Dideles, et al. *"Development of a Bluetooth™ Host-side Protocol Stack using Formal Design Techniques"*. 2nd National ECE Conference, Manila, Philippines, Nov. 2001.

[7] http://www.omg.org/

[8] Telecommunication Standardization Sector of ITU. *Message Sequence Charts*. ITU–T Recommendation Z.120, International Telecommunication Union, 1993.

[9] http://www.bluetooth.com/

[10] Bluetooth SIG, Bluetooth Specifications Book version 1.1, February 2001.

[11] Telecommunication Standardization Sector of ITU. *Specification and Description Language (SDL)*. ITU–T Recommendation Z.100, International Telecommunication Union, March 1993.

[12] Bluetooth SIG, PICS Proforma for Logical Link Control and Adaptation Protocol, February 2001.

[13] Telecommunication Standardization Sector of ITU. *Abstract Syntax Notation.1*. ITU–T Recommendation X.680, International Telecommunication Union, 1993.

[14] Telecommunication Standardization Sector of ITU. *SDL Combined with ASN.1*. ITU–T Recommendation Z.105, International Telecommunication Union, 1993.