Real-Time Object-Oriented Modeling of Private Branch Exchange Systems

Saehwa Kim
Department of Information and Communications Engineering
Hankuk University of Foreign Studies

This paper presents a real-time object-oriented model for private branch exchange (PBX) systems servicing cell phones. We selected OMG UML 2.0 as a specific real-time object-oriented modeling language. Our PBX model has a high degree of dynamic structures and behavior while having a typical layered structure where the bottom proxy layer accepts external inputs. This case study can be used as a reference model for designing any other kinds of complex real-time communications systems.

1. Introduction

As object-oriented modeling technologies are ever proliferating, the efforts of applying them to real-time domains resulted in a wide variety of commercial real-time object-oriented tools on the market. Modeling tools such as IBM Rational RoseRT [1], ARTiSAN Real-Time Studio [2], IBM Telelogic Rhapsody [3], IBM Telelogic Tau [4], and IAR visualSTATE [5] render it possible for designers to model real-time systems, analyze them via executable models, and generate executable code for the systems.

However, many real-time applications still are not modeled in real-time object-oriented models due to many reasons including the steep learning curve for real-time object-oriented models. To overcome this problem, we argue that various case studies should be performed for modeling real-world real-time systems. Specifically, we argue that a reference real-time object-oriented model should be provided that has a reasonable degree of complexity.

In this paper, we present a real-time object-oriented model for private branch exchange (PBX) systems servicing cell phones. While there are various modeling languages for real-time object-oriented modeling, we selected OMG UML 2.0 [6] as a specific real-time object-oriented modeling language since UML 2.0 is a de facto industrial standard. Our PBX system model is a typical layered model where the bottom proxy layer accepts external inputs. We describe our

---

1 This work was supported by Hankuk University of Foreign Studies Research Fund of 2009.
UML 2.0 PBX system model focusing on its structural and behavioral design, which shows how the model has a high degree of dynamic structure and behavior. The objective of this study is to provide a reference model for any real-time communication systems that are needed to be designed as real-time object-oriented models due to their complexity and evolving characteristics.

The remainder of the paper is organized as follows. Section 2 describes UML 2.0 modeling language. Section 3 describes the top-level structured model of our PBX system. Section 4 explains detailed structural and behavioral models for each top-level class. The final section concludes the paper.

2. UML 2.0

UML 2.0 is a general purpose modeling language developed by the OMG, and contains corrections and new content based on user feedback on the UML 1.x modeling language. One of the important additions in UML 2.0 is the concept of structured-classes. This concept makes it possible to define the run-time structure of a class as the composition of multiple structured-classes connected together. It has been developed to properly represent complex, event-driven, potentially distributed real-time and embedded systems. The additions to UML 2.0 are inspired by ROOM [7]; another object-oriented modeling technique for real-time systems.

The basic element of model construction in UML 2.0 is a structured-class. A structured-class represents an object within the system that communicates with other structured-classes exclusively through interfaces called ports. Structured-classes connected together define the run-time structure and communication channels of an application. A finite state machine, represented by a state diagram, represents the behavior of a structured-class. Receiving messages that are composed of signals and optional data via ports causes the state machine to make transitions, executing the logic contained in the structured-class.

The full behavior of a system is defined by the composition of all structured-classes, their connections, and their state machines. The structure of a structured-class is defined in a structure diagram. In this diagram other classes can be used as parts of the composition. These are referred to as structured-class-parts. A structured-class-part can be fixed, optional or plug-in. All fixed structured-class-parts contained in a system are instantiated when the system is initialized. Alternatively, a structured-class-part can be marked as optional or plug-in and such a structured-class-part is instantiated dynamically according to the needs of designers. They are not
instantiated at initialization but must be explicitly created and destroyed by a state transition. A plug-in structured-class-part is not an actual instance, but is a reference to an existing structured-class instance in the model, and is created by importing a reference to an instance of an incarnated optional or a fixed structured-class-part.

3. Top-Level Structural Model

Our PBX system model is a typical layered model where the bottom proxy layer accepts external inputs. The model exploits 29 structured-classes and consists of four top level structured-classes: ProxyManager, DeviceManager, OAMSubsystem, and CallController as shown in Fig. 1. In the following section, we describe the behavior and structure of these structured-classes.

Fig. 1. Simplified structured-class diagram of PBX system (ports were omitted for simplicity).
4. Structural and Behavioral Models

4.1 ProxyManager

This structured-class forms the bottom layer of the model that accepts external incoming messages. As shown in Fig. 10, it has PhoneUIListener, PhoneProxy, and UIAccessPoint as its parts, where UIAccessPoint is an imported part. When the PBX system initializes, a number of PhoneProxies are incarnated equal to the number of extensions in the system. External messages are delivered to PhoneUIListener and UIAccessPoint relays messages to PhoneUIFilter that is part of PhoneProxy. Incoming messages to the PhoneUIListener are the numeric buttons (button[0-9]), Send/End buttons (bSnd and bEnd), the power button (bPwr), and ack, which is the response to its ping signal from ConnectionObserver contained in PhoneProxy. ConnectionObserver monitors the connection so as to detect failures of a connected cell phone device. PhoneUIFilter processes the delivered messages according to its current state as shown in Fig. 2. As shown in its On::Ready state, PhoneUIFilter defers numeric button messages and recalls all together when bSnd message is delivered.

4.2 DeviceManager

This structured-class manages available phone devices that can access the PBX system. As shown in Fig. 1, it has Phone and PhoneProxy as its parts, where PhoneProxy is an imported part. When a physical cell phone device sends a power on (bPwr) message, one instance of Phone is incarnated. Then DeviceManager will import the PhoneProxy which received the bPwr message, and this will be connected to the newly incarnated Phone instance. From this a complete Phone abstraction results, where external messages are relayed by a PhoneProxy to a Phone instance, which interacts with the rest of the system.

When the Phone instance either places a call, or is called, it incarnates a specialization of the Session structured-class, namely OrigSession if it originating the call or TermSession if it is the terminal end of a call. The Phone also imports an instance of ExtensionAccessPoint from OAMSubsystem. This becomes a communication channel between the individual Phone instance and the OAMSubsystem. If a physical cell phone device sends a power off signal, the corresponding Phone instance is destroyed.
4.3 OAMSubsystem

This structured-class is for Operation, Administration, and Management (OAM) of PBX systems. As shown in Fig. 1, it has HLR (Home Location Register), GMSC (Gateway Mobile Switching Center), and SystemAdminProxy as its parts, where the last two structured-class-parts are place holders for further extension.

HLR maintains a mapping of telephone numbers to extension indices and the replication index of a PhoneProxy instance is considered the extension index. HLR also contains a number of Extension instances equal to the number of PhoneProxies. Each Extension instance is associated with one particular PhoneProxy instance. When a cell phone is powered on, the new Phone instance will import the Extension instance which is the counterpart to the PhoneProxy that received the power on message. In this way each Phone instance can communicate directly with OAMSubsystem.

![Fig. 2. State transition diagrams for PhoneUIFilter](image)
4.4 CallController

This structured-class is responsible for connecting two cell phones in a call. As shown in Fig. 1, it has Calls as its parts, each of which also has Sessions as its part. Fig. 3 shows the state transition diagram of Call. When a call is placed, an instance of the Call structured-class is incarnated and importOrig message is sent to the incarnated Call instance. As shown in Fig. 3, Call then imports the OrigSession of the originating Phone by transiting from Empty state to CallRequestInProgress state. The Call instance will then proceed to check whether the dialed cell phone number exists by communicating with ExtensionAccessPoint. If the number is valid, the Call instance transits to RoutingCall state and checks whether the called Phone instance is busy. If the called Phone is not busy, the Call instance will import the TermSession of the terminal Phone instance and transits to CallProcessing state, which means the call is connected. When one end of the call hangs up, the Call instance is destroyed.

Fig. 3. State transition diagrams for Call.
5. Conclusion

One of the major hurdles for adopting real-time object-oriented models are the their steep learning curve. As an effort to overcome this problem, we presented a reference real-time object-oriented model for a real-world complex and real-time communication system, which is a private branch exchange (PBX) system servicing cell phones.

While there are various modeling languages for real-time object-oriented modeling, we selected OMG UML 2.0 as a specific real-time object-oriented modeling language since UML 2.0 is a de facto industrial standard. Our PBX system model is a typical layered model where the bottom proxy layer accepts external inputs. We described our UML 2.0 PBX system model focusing on its structural and behavioral design, which shows how the model has high degree of dynamic structure and behavior.

References


사설 교환기를 위한 실시간 객체지향 모델

김세화
정보통신공학과
한국외국어대학교

요약: 본 논문에서는 휴대폰을 서비스하는 사설 교환기(PBX)를 위한 실시간 객체지향 모델을 제시한다. 실시간 객체지향 모델링 언어로서는 OMG UML 2.0을 선택하였다. 이 PBX 모델은 하부 대리 계층이 외부 입력을 받는 전형적인 계층적인 반면 그 구조와 행위가 고도로 동적이다. 이 사례 연구는 다른 종류의 복잡한 실시간 통신 시스템을 설계하기 위한 참조 모델로 사용될 수 있다.

2 본 논문은 2009년도 한국외국어대학교 학술연구비 지원에 의해서 연구되었음