Advanced Process-based Component Integration in Telcordia's Cable OSS

Anne HH. Ngu, Dimitrios Georgakopoulos, Donald Baker, Andrzej Cichocki
Telcordia Technologies
106 E. Sixth Street, Litttlefied Building suite 415, Austin, TX 78701

1. Introduction

Operation Support Systems (OSSs) integrate software components and network elements to automate the provisioning and monitoring of telecommunications services. This demonstration illustrates Telcordia's Cable OSS, and shows how customers may use this OSS to provision IP and telephone services over the cable infrastructure. Telcordia's Cable OSSs is a process-based application, i.e., a collection of flows, and specialized components (e.g., a billing system, a Call Agent soft switch, network services and elements, cable modems, etc.) and corresponding adaptors that are integrated, coordinated, and monitored using CMI, Telcordia's advanced process-based integration technology. Customers interact with the Cable OSS by using web or IVR (Interactive Voice Response) interfaces.

The *Collaboration Management Infrastructure* (CMI) is an advanced technology for supporting the integration of applications/components and the coordination of humans and applications possibly working in different enterprises. CMI combines the capabilities that until now were provided by separate technologies such as production workflow, groupware, and object-based middleware. In addition to combining these technologies, CMI provides many novel primitives and corresponding system functionalities supporting: coordination flexibility, process extensibility and synchrony, awareness provisioning, and advanced process-based application integration. In ICDE2000, we demonstrated a CMI-based solution for medical crisis mitigation that illustrated CMI's flexible coordination and awareness capabilities. In the current demo, we focus on CMI's advanced system integration and conversational coordination capabilities that have been utilized in the development of Telcordia's Cable OSS.

2. CMI process model

CMI's process model consists of a Core Model (CORE) and several specialized extensions of it. CORE provides the basis for all extensions including an abstract notion of processes and activities as well as a common set of process resources, e.g., static (organizational) and dynamic scoped roles [3].

The extensions are designed to support coordination, awareness, and services. The *Coordination Model* (CM) provides standard data and control flow primitives for coordinating participants and process enactment. In addition, CM provides advanced primitives for specifying state-based control flow, inhibitors, data context resources, activity placeholders supporting late binding and polymorphism [2], as well as optional and group activities[3].

The *Awareness Model* (AM) [1] is a CORE extension that captures awareness. We define awareness as information that is highly relevant to a specific role and situation of a process participant. AM allows customizing awareness via awareness specifications. Awareness specifications, which are provided by process/awareness designers, define what information should be directed to what users based on their (possibly dynamic) roles in the process.

The *Service Model* (SM) [4] supports reusable service activities, conversational service coordination, and service agreements needed to support processes integrating and coordinating e-services (web-services). Service activities differ from traditional (process) activities, because they separate the activity declaration from an activity implementation. Service interfaces correspond to the declaration of a service, basic service activities and service wrapper processes are service implementations supported by the SM. At process execution time, services can be selected and coordinated dynamically through a late binding mechanism (activity placeholder). SM primitives are used extensively in the cable OSS to integrate the cable infrastructure, billing, and Call Agent services that are typically provided by different enterprises.

3. CMI system architecture

The CMI system is depicted in Figure 1. User tools in CMI are organized into to separate clients for *participants* and *designers*. Participants are humans or programs involved in performing an activity in a process. Designers create and maintain CMM process, awareness, and service specifications. Participants interact with the CMI enactment system using CMI client tools for participants. The client

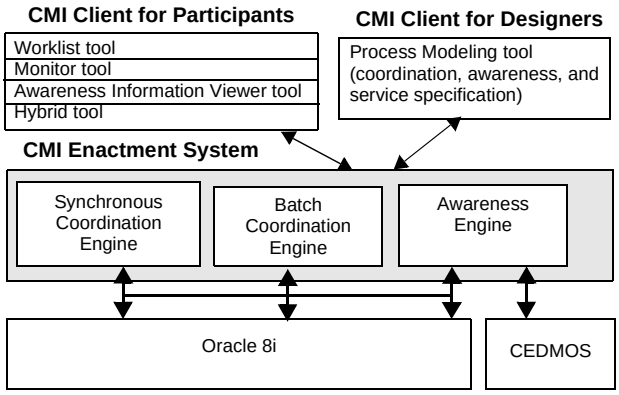

Figure 1. CMI System Run-time Architecture

tools subsume the tools defined by the Workflow Management Coalition. In particular, participant tools include a worklist tool, an awareness information viewer, a process monitoring tool, and a hybrid tool. The CMI worklist divides the activities each participant is eligible to perform to mandatory or *optional* [1] work items. The awareness information viewer maintains a participant's awareness event queue and displaying awareness events to him/her. The process monitor is similar to those provided by COTS workflow management systems. The hybrid tool combines a worklist with a viewer of the nesting of activities that appear in the worklist.This tool is particularly useful for debugging process specification.

The CMI enactment system contains several engines that implement the CMM. The Coordination Engines implements CORE, CM, and SM. The CMI engine design employs a pair of synchronous and batch engines to improve throughput and reduce response time. The coordination engine pair is designed to support the division of activities into suboperations (e.g., start, complete, resolve role, inhibit, cleanup) and the pipelining and parallel execution of suboperations that originate form the same activity. The Awareness Engine that implements AM heavily leverages features provided by the event processing system CEDMOS (Composite Event Detection and Monitoring System) [2].

4. Demonstration

The provisioning of telephone services over cable infrastructure involves integrating and coordinating diverse system components and applications. The goal is to empower the customer to use a web browser or an IVR to provision the telephone services without any human operator or cable technician. There are two conditions that the customer must satisfy in order to do self provisioning. The customer must have a cable TV service connected to his/her home and must have purchased an integrated telephony cable modem (ITCM) certified by the cable company (these are currently available in several electronics store chains). In this demo, we will show that after the customer plugged in the cable modem, a form for signing up for new service web page will be displayed at the PC connected to the cable modem. The customer can then follow the instructions to set up new services. Alternatively, the customer can dial "611" on a telephone connected to the ITCM, which will connect the customer to the IVR interface at the cable company to self provision the telephone services.

Figure 2 shows the different components that CMI needs to coordinate and integrate in order to provision telephone services over cable. Note that the components are located at at least three different premises: customer's home, cable company and local exchange telephone providers (Call Agent, or CA Providers). The different components include the customer self-care system (web/IVR); the customer billing system; a telephone number server; various network elements such as DHCP server, DNS and TFTP server; the cable modem at customer's home and possibly several call agent soft switches from third-party phone service providers. A variety of interfaces are used in the different components. For example, the interaction with the local exchange phone services providers (call agent) and IVR are conversational in nature and involve complex processes. The interaction with cable modem involves SNMP protocol, those with IVR involves MGCP protocol and CORBA. Simple TCP/IP and XML is used for interaction with billing system and SOAP protocol is used with telephone number server.

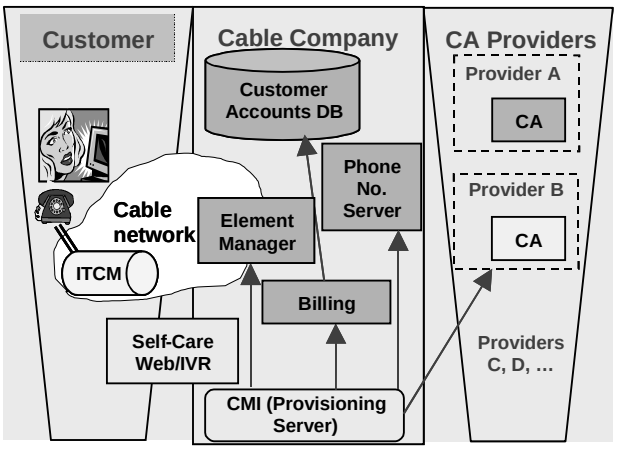

Figure 2. Cable OSS Architecture

The main activities involved in the process are show in Figure 3. For the ease of explanation, it is shown as a UML sequence diagram (a monitoring view) rather than a block diagram produced by the CMI process design tools. The process involves the customer submitting a request to the Web site (or IVR) which then sends an XML message to CMI. CMI identifies the customer, then configures the network services, updates the customer's billing record and finally notifies the customer about the status of the provision-

ing process. Depending on the service provider chosen by the customer, CMI needs to notify or configure the correct Call Agent (soft switch). The cable modem also needs to be reset with the correct service parameters depending on the services selected by the customer.

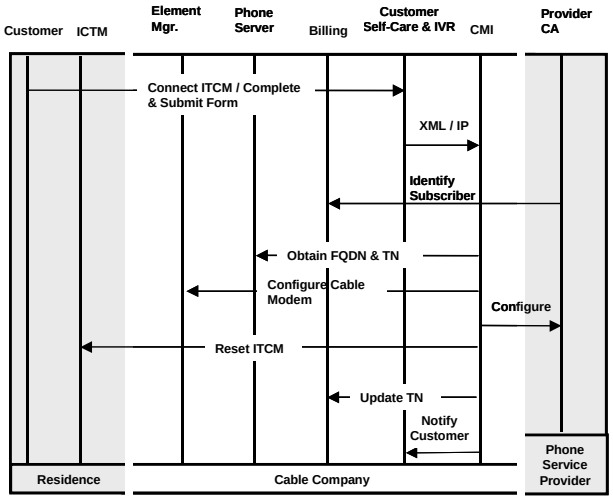

Figure 3. Provisioning Process

The main advantages of using CMI are its abilities to deal with heterogeneous components and evolving processes, cope with the autonomy of integrated components, capture the semantics of the inter-organizational conversational interaction, provide flexibility, and support for automatic adaptation.

CMI addresses the integration of heterogeneous components by providing object oriented proxies which can have one interface (specified as a state machine) with multiple implementations. For example, the proxies for interaction with different components (call agent, billing system, IVR, network elements etc.) all share the same state machine interface but have different implementations ranging from a simple JDBC program, XML with TCP/IP, XML with SOAP, simple scripts to CORBA services.

CMI provides application-specific state machines and operations for modeling of services. This allows selective monitoring of state changes in external services. This is particularly useful for end-to-end process monitoring of telephone service provisioning processes from the cable company to the third party local exchange service provider. For example, critical states of local exchange telephony provisioning process could be checked by the cable company.

CMI's placeholder activity primitive allows late-binding of processes based on certain selection criteria. This allows controlled process extension at runtime. This feature is critical for implementing the multiple trading relationships with different local exchange service providers. In the demo, this primitive allows the customer to choose the provider that he/she wants and bind to the correct provider at service provisioning time.

State-dependent control flow coupled with the state machine for an activity allows handling of exceptions without explosion of activities. For example, there are uniform interaction patterns to handle recoverable error and fatal error. State-dependent control flow also enables the coordination of conversational activities (for example, interaction of user with Web/IVR).

These capabilities make CMI an excellent platform for integrating diverse and evolving system components, and developing sophisticated OSSs faster and cheaper. The development of the Cable OSS using CMI to integrate and coordinate its components took less than 6 man months. Developing such a complex OSS using conventional OSS development typically requires years of teamwork and corresponding development costs.

References

[1] Baker, D. ; Georgakopoulos, D. ; Schuster, H. ; Cassandra, A.R. ; Cichocki, A.: Providing Customized Process and Situation Awareness in the Collaboration Management Infrastructure. In: *Proc. Fourth IFCIS Conference on Cooperative Information Systems (CoopIS'99)*, Edinburgh, 1999.

[2] Schuster H.; D. Georgakopoulos; A. Cichocki; and D. Baker: Modeling and Composing Service-based and Reference Process-based Multi-enterprise Processes. In: Proceedings of the 12th Conference on Advanced Information Systems Engineering, Stockholm, Sweden, June 2000**.**

[3] Georgakopoulos, D. ; Schuster, H. ; Baker, D. ; Cichocki, A.: Managing Escalation of Collaboration Processes in Crisis Mitigation Situations. In: *Proc. ICDE'2000*, San Diego, 2000.

[4] Georgakopoulos, D. ; Schuster, H. ; Cichocki, A. ; Baker, D.: Managing Process and Service Fusion in Virtual Enterprises. In: *Information Systems, Special Issue on Information Systems Support for Electronic Commerce*, **24**(6), 1999.