A Component-Based Real-Time Architecture for Distributed Supervision and Control Applications

Sandro Santos Andrade and Raimundo José de Araújo Macêdo

Distributed Systems Laboratory (LaSiD), Pos-Graduation Program on Mechatronics *
Federal University of Bahia, Campus de Ondina, 40170-110, Salvador-BA, Brazil

Abstract

Nowadays, the development of flexible and interoperable software platforms for industry is an important issue. The CCM model captures two paramount features of such platforms, as it combines component-based middleware (easy composition of new applications and maintainability) and the openness of the CORBA standard. Though there exist an implementation of CCM devoted to real-time systems (CIAO platform), much effort is needed to validate its use in the real-time industry scenario. This paper contributes to this goal by presenting the design and implementation of a new framework over CIAO, which conforms to the DAIS standard (Data Acquisition from Industrial Systems). We discuss our design decisions and show how the framework can be used to develop distinct S&C applications. We also discuss implementation details and show performance data from a series of experiments.

1. Introduction

The increasing use of software-based solutions in industry, in conjunction with recent advances in hardware and communications technologies such as the switched Ethernet, has enabled considerable changes in real-time supervision and control (S&C) systems. Nowadays, the design of that kind of systems begins to consider requirements such as distribution, flexibility, scalability, adaptation, intelligent algorithms, interoperability, reusability, and web access [21, 22, 30]. As a consequence, new paradigms and methodologies for real-time software development emerged with the purpose of managing the complexity generated by these new requirements, [14, 13]. Such methodologies usually adapt existing software engineering techniques to the real-time scenario, such as middleware technologies [7, 21], distributed components [11, 20], design patterns [23], and architectural models [26]. Among these, component-based system development is a promising technique, due to its inherent adequacy to distribution (a component is an independent entity) and easy maintainability (components can be easily replaced). The goal of component-based system development is to construct new systems from the composition or assembly of existing software components, which executes in an environment called application server or component server. The component server is responsible for managing the life cycle of components, providing services such as component localization, persistence, and security. With this approach, the component developer concentrates just on the functional (or application) code [6]. Therefore, the use of component technology in real-time systems development has been object of recent research, which are usually concerned with the proposal of new component models [28] or adaptation of existing solutions for using in real-time applications [3, 29]. Component-based technology has also been employed for the development of architectural model designs and frameworks1 [9]. Besides to leverage the development productivity by promoting the reuse of software components, the adoption of a framework reduces possible design flaws.

Another important requirement for industry software is that of interoperability, since it allows integration of distinct pieces of software running over a variety of platforms, from shop floor to manufacturing business management. CORBA [12] and OPC/DCOM [10] are two major middleware technologies that are being used to promote such integration. Whereas the predominance of Microsoft/Windows platform has made OPC/DCOM a widely used platform, CORBA, as an open architecture, has been designed specifically to handle interoperability among distinct platforms from distinct vendors. In particular, there exist a CORBA standard directed to data acquisition for industrial systems, the DAIS standard [17]. Being a CORBA-based standard, enables clients aware of the standardized interfaces, to take full advantage of the related implementations in despite of programming language, operating system, and hardware platforms. Moreover, the use of the DAIS standard certainly contributes to promote the reuse of industrial software components.

1A framework is a general implementation of recurrent features for a given application domain.
The framework architecture of ARCOS defines interoperability standards, through component specifications, for the three basic entities present in industrial systems (data acquisition, control, and supervision).

Communication among components of the proposed framework is accomplished by an event-based mechanism, implemented by the TAO’s (The ACE ORB) Real-Time Event Service [24]. Besides making possible an uncoupled and non-blocking communication, that service enables priority-based event dispatching, which leads to a more predictable environment. Figure 1 is a general picture of the proposed ARCOS framework and related technologies. In the lowest layers are the communication sub-system and operating system, the intermediate layers are composed by CIAO, ACE and TAO. As mentioned before, CIAO is an implementation of CCM devoted to real-time systems, and together with ACE and TAO form the basic run-time facilities upon which the framework is built. The framework layer corresponds to component specifications for data acquisition, control, and supervision. Finally, in the uppermost layer are the industrial applications.

The rest of this paper is organized as follows. Section 2 discusses related work and section 3 discuss the design decisions that lead to our framework and presents the framework architecture. Section 4 discusses implementation issues and section 5 shows how the framework can be specialized for distinct applications. In this section, a supervision application of a chemical reactor is detailed. Finally, section 6 draws some conclusions and points for future works.

2. Related work

The use of distributed components in real-time systems development is a recent effort. The majority of the real-time component model implementations constitute ongoing researches and there are few works validating those implementations. Therefore, the design and implementation of reusable and interoperable real-time platforms still represents a major challenge.

Previous work has applied software engineering techniques on industrial S&C systems. In [23] and [26] the use of design patterns for control systems have were investigated. In [5] a CORBA-based architecture for S&C systems development has proposed, which corresponds to CORBA services and interfaces for supervision and data acquisition activities. Moreover, an object-based and formal methodology for industrial systems development was defined. However, aspects such as guarantee of temporal constraints and the use of standardized interfaces were not considered in their work. Furthermore, the proposed architecture is not based on component technology, implying in a less extensible and flexible platform.

In [16] a Java-based framework for web integration of industrial processes has been presented. The authors propose the term "virtual plant" as the basic mechanism for mapping real processes in pre-defined objects. This project does not address, however, mechanisms for temporal requirements specification and guarantee. Furthermore, their work does not consider objects for automatic control loops.

In [4, 15] a CORBA-based S&C system has been proposed, but the idea of a reusable and extensible architecture was not addressed.

The CoSMIC (Component Synthesis with Model Integrated Computing) [25] project represents a complementary effort regarding real-time architectural models and component middleware integration, as instead of providing industry oriented frameworks, its main concern is to generate code from UML specifications to the real-time CIAO environment.

The work presented in [8] introduced a component-based industrial message service based on CCM, and in this sense it has similar characteristics to the framework ARCOS presented in this paper. However, their implementation is based on the MICO system [19], a CCM implementation that does not tackle timeliness requirements. In contrast, the implementation of ARCOS is based on CIAO, as mentioned before, a real-time oriented implementation of CCM.

3. The proposed architecture

The framework architecture we present in this paper, devoted for the industrial S&C systems domain, was designed to be flexible, reusable, and interoperable. In the following, we first discuss the design decisions behind the achievements of these three requirements. After that, we
The need for flexibility arises from the very fact that earlier real-time systems were usually designed for specific applications, implying in high development costs to handle system evolutions or modifications to accommodate new requirements. By adopting a component-based model, we achieve a more maintainable solution by the use of flexible component communication and relationship models. The CORBA Component Model (CCM) [18] and CIAO (Component-Integrated ACE ORB) [29], were the component technologies adopted on our framework. CIAO extends the CCM specification, providing a real-time component server which collects temporal constraints in deployment time. Other researches [3] present alternative technologies for using components in real-time, embedded systems. However, we chose CIAO due to the fact that it is based on consolidated real-time technologies, such as ACE (ADAPTIVE Communication Environment) and TAO [24].

Another aspect of flexibility is that of some modern industrial systems require many-to-many, uncoupled, and non-blocking communication - for instance, when a sensor node needs to send some information to a group of receivers (e.g., controllers and supervision systems). Hence, we chose to use the TAO's Real-Time Event Service to handle system evolutions or modifications to accommodate new requirements. By adopting a component-based approach, which defines generic components for S&C systems. In order to instantiate the framework for a given application, the developer’s job is reduced to a few class specializations and component assembly configuration.

Heterogeneity in industrial systems has demanded integration solutions that must regard distinct technologies and standardizations, from the factory’s ground floor - with equipments from different vendors - up to the supervision systems that increasingly require integration with other systems and environments. Thus, defining and adopting standards for interoperability becomes an important issue, as communication technologies are more available and heterogeneity in computational environment is virtually unavoidable. To address this point, the proposed framework has adopted the DAIS (Data Acquisition from Industrial Systems) CORBA standard. The main goal of DAIS is to map data collected from a specific data acquisition device to standardized CORBA interfaces, as illustrated in figures 2 and 3. This mapping provides a tree data structure containing all data available from a target device. DAIS clients will then be able to search and select nodes of such a tree. For instance, a DAIS server can export registers from a PLC (Programmable Logic Controller) by building a tree branch for each register type (B3 - internal bits, I1 - input ports, O0 - output ports). Figure 4 presents the UML sequence diagram of the protocol used by DAIS for data acquisition. As can be seen in the figure, after obtaining the DAIS server reference, the data access session is created. From this object, the group factory object is obtained (group home) and with the group home object, several data groups can be created. For each data group created, a new group management object is instantiated (group manager). The data acquisition is realized by a callback object provided by the DAIS client whose reference is informed to the group manager by the callback() method. The group manager object is also responsible for including new data items in the group. These data items are obtained by searching the DAIS tree as depicted in the example of figure 3. After this, the on_data_change() method on callback is invoked with the rate informed during the group creation, updating the client about changes in desired data.

Other standards for data acquisition were proposed in the last years. Between them, we can highlight OPC (OLE for Process Control) [10]. Despite the large acceptance of OPC by industry, it presents as drawback the limited use in proprietary platforms, restricting the desired interoperability scope.

Finally, we would like to observe that the real-time scheduling service provided by TAO handle timeliness constraints by storing temporal requirements provided by the applications and by executing schedulability tests. Currently, the Real-Time Scheduling Service of TAO supports the RMS (Rate Monotonic Scheduling) and MUF (Maximum Urgency First) [27] algorithms.

In the next subsection we detail the ARCONS component specifications used for data acquisition and control. Supervision applications are developed by clients consuming items provided by the data acquisition interfaces.

### 3.1. Data Acquisition

The data acquisition layer is composed by components implementing the DAIS standard. These components provide a reusable framework that can be easily specialized to diverse data acquisition devices used in industry. Figure 5
Figure 4. UML sequence diagram of a DAIS session.

shows the CCM components that compose this layer.

The DAIS Server component is responsible for direct communication with clients, creating data access sessions and acquisition groups. In order to enable service reusability in environments with diverse acquisition technologies, we created the role of DAIS Provider (implemented by the AbstractDAISProvider component). The AbstractDAISProvider component defines abstract interfaces that act as a connection contract between the DAIS server and data providers. By defining a receptacle with type AbstractProviderFacet, the DAIS Server can be connected to any component that provides a facet implemented by an interface inherited from AbstractProviderFacet.

The DAIS Server plays also the role of event supplier, sending data to the real-time event service. The event service, in its turn, makes the priority-base dispatch to all components interested in this information. The DAISWriter component is responsible for consuming events from controllers and supervisory systems and uses the DAISServer component in order to update data in devices, usually actuators.

In our prototype implementation we have implemented a DAIS provider for acquiring data from an Ethernet-connected PLC (DAISEthernetPLCProvider component). For utilizing other device technologies such as a parallel port or a industrial network, it suffices to implement the DAIS provider for such a specific target device and to configure the corresponding XML descriptor file to redirect the corresponding facet/receptacles connections.

3.2. Control

A similar approach was used for designing the control layer components, as illustrated in figure 5.

The ControlManager component consumes, through the event service, messages produced by the DAIS Server, it executes the control algorithm properly connected, and then sends actuation messages back to the event service. Finally, the actuation messages are received by the DAISWriter component, which accomplishes the actuation in the system.

Currently, in our prototype, there is a PID controller implementation where the tuning parameters were developed as CCM attributes and configured at component deployment time.

3.3. Supervision

We have implemented in the ARCOS prototype two applications that play supervision roles, the DAIS Monitor and the DAIS browser. The DAIS Monitor (figure 11) enables the visualization of the DAIS server state, presenting data sessions and data groups inside each data session. Moreover, the DAIS Monitor is also responsible for activating the real-time event service, executing the schedulability tests and starting the dispatching of messages.

The DAIS Browser is a generic viewer for any DAIS server. With the DAIS Browser tool, the user is able to create data groups, insert new data items into groups, and, after the event service activation in DAIS Monitor, it is possible to monitor the current state of a plant. The DAIS Browser is currently available in desktop and web versions.

4. Implementation Issues

The ARCOS prototype was developed in a Debian GNU/Linux platform and using the C++ programming language. However, all used technologies, from CCM implementation to graphic user interfaces, are based in portable libraries, facilitating system interoperability.

Data acquisition loop, clients, and web integration

An interesting feature of TAO’s Real-Time Event Service
is the sending of timeout events, a fundamental feature to develop time-based applications. As such, timeouts are just internal events generated by the event service, in a previously informed rate, to which conventional event consumers can be connected. Figure 6 show how such timeout events are used to accomplish periodic data transfer in our system.

The Group Manager component plays at the same time the roles of timeout event consumer, data event producer, and data event consumer (see the UML class diagram in figure 7). Back to figure 6, whenever a new timeout event is received, the Group Manager component asks, to the connected DAIS provider, new values for all group items. Then, it informs the DAIS client, by invoking on_data_change() method of callback object, and sends this information to the event service. The event service forwards, in its turn, the data update to other interested clients.

The supervision systems (DAIS Monitor and DAIS Browser) were implemented using the WxWidgets [2] portable toolkit, in order to enable a viable migration to platforms like Microsoft Windows.

The integration of legacy systems into the web is an indispensable activity nowadays. In industrial environments, web systems for plant supervisioning and monitoring are even more required. In our project prototype, it was implemented a web-based version of the DAIS Browser, allowing plant information visualization from any web browser. This system is composed by JSP (Java Server Pages) pages using stubs compiled from DAIS IDL (Interface Definition Language) files. For IDL compilation we used OpenCCM, an on-going CCM implementation for the Java programming language.

5. Example application

The ARCOS platform, designed as a framework for the supervision and control application domain, provides a reusable infrastructure that can be instantiated for using in particular S&C situations. The implemented component communication architecture, regarding modularity in data acquisition and control levels, can be used in a range of systems, with different requirements of networking, devices, and control algorithms. In the following, we illustrate the use of ARCOS in some typical scenarios, and detail its use in an example application that we have implemented, chemical reactor simulator.

- Complex and/or intelligent control systems. As control algorithms became more complex, more robust software platforms are needed for reliable control ex-
Figure 7. UML class diagram for data acquisition loop.

Figure 8. Supervisioning activities of a chemical reactor simulator.

6. Conclusion and future work

This paper presented the design and implementation of a real-time, component-based, interoperable architecture for distributed S&C systems. ARCOS possesses a number of desired features for industrial systems: flexibility (new data providers or control algorithms can be easily connected to the platform); interoperability (due to the use of the CORBA and DAIS standards); predictability (by using real-time event service and component server of TAO); low cost (by using commercial-of-the-shelf hardware and software components).
Currently, we are evaluating the use of ARCOS for automatic control loops, and the development of components for activities registration and alarms. We are also investigating the definition of new components and services concerned with system dependability (failure detectors and replication) and adaptation (meta-components and dynamic facet-receptacle reconnection).

References


