ARCOS: A Component-Based Architecture for the Construction of Robust Control and Supervision Applications

Conference Paper · January 2005

CITATION
1

READS
55

3 authors:

Sandro Andrade
Federal Institute of Education, Science, and Technology of Bahia (IFBa)
37 PUBLICATIONS **49** CITATIONS
SEE PROFILE

Raimundo J de A Macêdo
Universidade Federal da Bahia
78 PUBLICATIONS **649** CITATIONS
SEE PROFILE

Alirio Sá
Universidade Federal da Bahia
13 PUBLICATIONS **35** CITATIONS
SEE PROFILE

ARCOS: A Component-Based Architecture for the Construction of Robust Control and Supervision Applications

Sandro **Andrade**[⋆], Raimundo **Macêdo**, and Alírio **Sá**[⋆⋆]

Distributed Systems Laboratory – LaSiD
Computing Science Department, Federal University of Bahia
Campus de Ondina, CEP: 40170-110, Salvador-BA, Brazil
sandros@ufba.br, macedo@ufba.br, aliriosa@ufba.br

Abstract. Integration and interoperability are major challenges of modern supervision and control industrial systems. Such needs arise from the use of equipments from different vendors, with operating systems and communications incompatibilities, up to diverse factory machinery - each of them usually supplied by a specialized company (PLC's, numerical control machines, robot arms). The ARCOS platform has been developed to address this important issue of modern industrial systems. It is based on the CCM model, which combines component-based middleware (easy composition of new applications and maintainability) and the openness of the CORBA standard. This paper overviews the ARCOS platform and discusses its components designed to provide a flexible failure detection mechanism. Such components can be customized to handle distinct QoS requirements and be adaptive to the current system and network load.

1 Introduction

Heterogeneity is an inherent characteristic of modern industrial systems. Indeed, such systems are usually composed by a number of sub-systems acquired from different vendors; each of them specialized in efficient solutions for specific needs. Consequently, factory automation has to face the challenge of integration and interoperability. On the other hand, the increasing use of software-based solutions in industry in conjunction with recent advances in hardware and communications technologies has enabled considerable changes in industrial real-time supervision and control (S&C) systems. Hence, besides integration and interoperability, the design of such systems has now to consider new requirements such as distribution, scalability, adaptation, reusability, and web access [1–3]. To address these new challenges, new methodologies for real-time software development emerged, adapting conventional software engineering techniques to the

⋆ Master Student of the Mechatronics Program at UFBA
⋆⋆ Master Student of the Mechatronics Program at UFBA

real-time scenario (e.g., [4, 1, 5–7]). 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 [8]. Component-based technology has also been employed for the development of architectural model designs and frameworks[1] [9]. Besides to leverage the development productivity by promoting the reuse of software components, the adoption of a framework reduces possible design flaws.

Interoperability in industry requires the integration of distinct pieces of software running over a variety of platforms, from shop floor to manufacturing business management. CORBA [10] and OPC/DCOM [11] 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 [12]. 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.

The CORBA Component Model (CCM), which is part of the CORBA 3.0 standard, was released in June 2002. Though there are a few implementations of CCM such as MICO [13] and OpenCCM [14], they do not usually address real-time features, needed for time critical applications like industrial supervision and control. The CIAO implementation is one exception as it builds on the real-time ORB TAO [15]. However, much effort is needed to validate its use in the real-time industry scenario. On the other hand, since CIAO does not fully implement all facilities prescribed in the CCM model, some complementary effort is needed when developing such a component-based framework adherent to CCM.

Following this context, this paper contributed by discussing the ARCOS (Architecture for Control and Supervision) framework, introduced by us in [16], built atop CIAO and in conformance with the DAIS standard. 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). Furthermore, this paper introduces the main components of ARCOS intended to implement a generic failure detection service. The rest of this paper is organized as follows. In section 2 is given an overview of

[1] A framework is a general implementation of recurrent features for a given application domain.

ARCOS. In section 3 it is briefly described the components for adaptive failure detection in ARCOS, and, finaly, in section 4 some conclusions are drawn.

2 The ARCOS architecture

The ARCOS platform has been designed to be flexible, reusable, and interoperable. Figure 1 is a general picture of ARCOS and related technologies. In the lowest layers are the communication sub-system and operating system, the intermediate layers are composed by CIAO, ACE (ADAPTIVE Communication Environment) and TAO. CIAO together with ACE and TAO form the basic runtime facilities upon which the platform is built. The platform layer corresponds to component specifications for data acquisition, control, and supervision. Finally, in the uppermost layer are the industrial applications.

Some modern industrial systems require many-to-many, uncoupled, and nonblocking 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 achieve such desirable communication flexibility [17]. Besides making possible an uncoupled and nonblocking communication, that service enables priority-based event dispatching, which leads to a more predictable environment.

To address the interoperability demanded by industrial systems, the ARCOS platform adopted the DAIS CORBA standard. The main goal of DAIS is to map data collected from a specific data acquisition device to standardized CORBA interfaces. Other standards for data acquisition were proposed in the last years. Between them, we can highlight OPC (OLE for Process Control) [11]. Despite the large acceptance of OPC by industry, it presents as a drawback the limited use in proprietary platforms, restricting the desired interoperability scope.

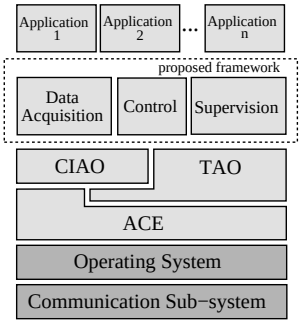

Fig. 1. General view of the architecture

Finally, we would like to observe that the real-time scheduling service provided by TAO handles 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) algorithms [18].

Data Acquisition, Control, and Supervision Data acquisition in ARCOS (see fig. 1) is realized by components implementing the DAIS standard. These components provide a reusable framework that can be specialized to diverse data acquisition devices used in industry. Below are the main components for this layer.

- **DAIS Server component**. It is responsible for direct communication with clients, creating data access sessions and acquisition groups. 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.
- **AbstractDAISProvider component**. It defines abstract interfaces that act as a connection contract between the DAIS server and data providers.
- **Specific concrete DAIS providers**. These components have the goal of applying the platform to a specific data acquisition situation. In our prototype there is a DAIS provider for an Ethernet-connected PLC. The use of other technologies can be achieved by implementing the DAIS provider for the target device and by configuring XML descriptor files, which redirects the CCM facet/receptacles connections.
- **DAISWriter component**. It consumes data from the event service and it uses the DAIS Server component in order to update device data, such as set-points or control variables.

A similar approach is used for designing the control layer components. 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.

We have implemented, in the ARCOS prototype, three applications that play supervision roles: the DAIS Monitor, the generic DAIS browser, and a specific supervisory client for a chemical reactor simulator. The DAIS Monitor 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.

3 Adaptive Failure Detection in ARCOS

The TAO's ORB implementation supports fault tolerance, following the features defined in the FT CORBA specification [19]. However, such an implementation does not support QoS-tuning failure detection, nor the possibility of adapting to new failure prediction strategies. In contrast, the component-level adaptive failure detection service of ARCOS is able to tune the deliverable QoS according to specific application requirements, and also can be adaptive in relation with diverse failure detection algorithms.

The ARCOS's approach makes use of the encapsulation and modularisation characteristics inherent to the component technology to allow for the implementation of distinct detection models (*push* or *pull*) and prediction algorithms (e.g., distinct adaptive detection strategies [20–22]). Similarly to the work presented in [20], our approach follows a layered structure with the following layers, each one implemented by a distinct component:

- QoS Adaptation Layer, responsible for mapping specific application failure detection requirements into QoS failure detection metrics (detection time, mistake recurrence time, and mistake duration time)[23];
- QoS Tuning Layer, responsible for getting the QoS metrics from the Adaptation Layer and to promote the necessary run-time adjustments to keep the system operation according to the required QoS, adapting to possible run-time changes such as message transmission/reception rates, message loss probability, communication delays, etc.;
- and, the Basic Detection Layer, responsible for monitoring the process states.

4 Conclusions

This paper discussed the design and implementation of a component-based, interoperable architecture for distributed S&C systems, and briefly presented its generic failure detection service. 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). The ARCOS prototype has been implemented in a Debian GNU/Linux platform, using the C++ programming language.

References

1. Sanz, R.: A corba-based architecture for strategic process control. Annual Reviews in Control **27**(1) (2003) 15–22
2. Sanz, R., Alonso, M.: Corba for control systems. Annual Reviews in Control **25**(1) (2001) 169–181

3. Wills, L., Kannan, S., Heck, B., Vachtsevanos, G., Restrepo, C., Sander, S., Schrage, D., Prasad, J.: An open software infrastructure for reconfigurable control systems. Proc. 19th Amer. Control Conf. (ACC-2000), Chicago, IL (2000) 2799–2803
4. Diaz, M., Garrido, D.: Applying rt-corba in nuclear power plant simulators. In: ISORC. (2004) 7–14
5. Fredriksson, J., Åkerholm, M., Sandström, K., Dobrin, R.: Attaining flexible real-time systems by bringing together component technologies and real-time systems theory. In: EUROMICRO. (2003) 399–403
6. Sanz, R., Zalewski, J.: Pattern-based control systems engineering. IEEE Control Systems **23**(3) (2003) 43–60
7. Selic, B.: An architectural pattern for real-time control software (1996)
8. Crnkovic, I., Larsson, M.: Building Reliable Component-Based Software Systems. Artech House, Inc. (2002)
9. Fayad, M.E., Schmidt, D.C., Johnson, R.E.: Building application frameworks: object-oriented foundations of framework design. John Wiley & Sons, Inc. (1999)
10. Group, O.M.: Corba Specification. (http://www.omg.org/gettingstarted/corbafaq.htm)
11. Foundation, O.: OLE for Process and Control Standard. http://www.opcfoundation.org (1997)
12. (OMG), O.M.G.: Data Acquisition from Industrial Systems (DAIS), Request for Proposal (RFP), OMG Document: dtc/99-01-02. http://www.omg.org/techprocess/meetings/schedule/Data_Acquisition_RFP.html (1999)
13. Puder, A., Römer, K.: Mico – mico is corba. dpunkt-Verlag (1998)
14. : OpenCCM - The Open CORBA Component Model Platform. (http://openccm.objectweb.org)
15. Schmidt, D.C., Levine, D.L., Mungee, S.: The design of the TAO real-time object request broker. Computer Communications **21**(4) (1998)
16. Andrade, S., Macêdo, R.J.A.: A component-based real-time architecture for distributed supervision and control applications (full paper). In: Proc. of 10th International Conference on Emerging Technologies and Factory Automation, Catania, Italy, IEEE (2005)
17. Harrison, T.H., Levine, D.L., Schmidt, D.C.: The design and performance of a real-time CORBA event service. In: Proceedings of OOPSLA '97. (1997) 184–200
18. Stewart, D.B., Khosla, P.: Real-time scheduling of dynamically reconfigurable systems. In: IEEE Int. Conference on Syst. Engineering. (1991) 139–142
19. Wilson, D.and Totten, S.: Fault Tolerant (FT) Corba Services. http://www.cs.wustl.edu/ (2002)
20. Bertier, M., Marin, O., Sens, P.: Performance Analysis of Hierarchical Failure Detector. In: Proc. Of The International Conf. On Dependable Systems And Networks, San-francisco,Usa, IEEE Society Press (2003) 635–644
21. de Sá, A.S., Macêdo, R.J.A.: An adaptive failure detecion approach for real-time distributed control systems over shared ethernet (to appear, full paper). In: Proc. of 18th International Congress of Mechanical Engineering - Mechatronics Symposium (full paper), Ouro Preto, Brazil, COBEM2005 (2005)
22. Nunes, R.C., Jansch-pôrto, I.: Qos of Timeout-based Self-tuned Failure Detectors: The Effects Of The Communication Delay Predictor And The Safety Margin. In: Proc. of Int. Conf. On Dependable Systems And Networks. (2004) 753–761
23. Chen, W., Toueg, S., Aguilera, M.K.: On the quality of service of failure detectors. IEEE Trans. Comput. **51**(1) (2002) 13–32