ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOMPL007

THE DESIGN OF INTELLIGENT INTEGRATED CONTROL SOFTWARE FRAMEWORK OF FACILITIES FOR SCIENTIFIC EXPERIMENTS

Z. G. Ni†, L Li, J. Luo, J. Liu, X. W. Zhou
Institute of Computer Application, China Academy of Engineering Physics, Mianyang City, China
Y. Gao, Stony Brook University, New York, USA

Abstract

The control system of the scientific experimental facility requires heterogeneous equipment access, domain algorithm, sequence control, monitoring, log, alarm and archiving. We must extract common requirements such as monitoring, control, data acquisition. Based on the TANGO framework, we design typical device components, algorithms, sequence engines, graphical models and data models for scientific experimental facility control systems developed to meet common needs, and are named the intelligent integrated Control Software Framework of Facility for Scientific Experiments (iCOFFEE). As a development platform for integrated control system software, iCOFFEE provides a highly flexible architecture, standardized templates, basic functional components and services for control systems that increase flexibility, robustness, scalability and maintainability. This article focuses on the design of the framework, especially the monitoring configuration and control flow design.

INTRODUCTON

Large scientific facilities [1] generally have tens of thousands to hundreds of thousands of irregular control points, and the types and quantities of controlled devices are huge. The control system must be highly automated and robust, requiring continuous operation for many days. The construction period of the project is long, and the demand in many fields will constantly change. The control system must accept these changes and adapt to the changing and expanding needs.

We urgently need to consolidate the common requirements of monitoring, control, data acquisition and storage of the control system to meet the functional performance requirements of the facility's control system. iCOFFEE is a distributed, hierarchical, object-oriented control software framework through which many similar application software systems are built.

This control software framework [2] is devised to address the general problem of providing distributed control for large scientific facilities that do not require real-time capability within the supervisory software. Sometimes real-time control is also necessary, which is partly solved by an integrated time system or an industrial control system, which is not discussed in this article.

SYSTEM ARCHITECTURE

The control system architecture of a typical large scientific facilities is a two-tier architecture consisting of a monitoring layer of the network structure and a control layer of the fieldbus structure. The monitoring layer is deployed on the virtual server and the console computer to provide centralized operations for control, status, and data storage. The control layer is deployed on the virtual server or embedded controller to provide real-time collection and control of the device.

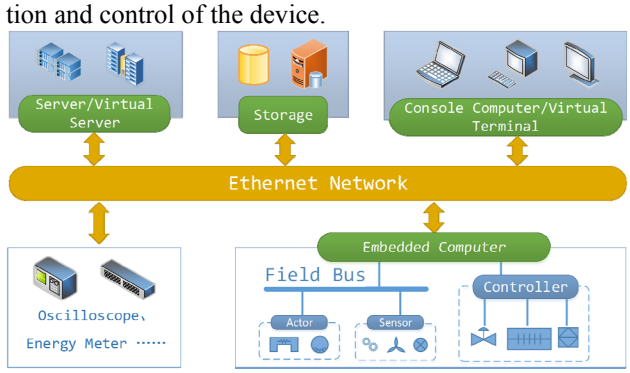

Figure 1: The control system architecture of the typical large scientific facilities [3].

As provided in Fig. 1, the monitoring layer is a software system based on Ethernet structure. It consists of a network switching system, a server system, and a console computer. It provides system services and human-machine interfaces for the facility control system, including control, monitoring, and data management.

The control layer is a fieldbus-based data acquisition and control software system consisting of a network switching system, a server system and an embedded controller. The device service software is used to collect and control the device.

The control point consists of sensors and actuators that are connected to the control layer via a fieldbus or network interface. A large number of IO devices are accessed through the PLC controller, and several intelligent controllers directly access the aggregation switching system through a serial port server or a network interface.

SOFTWARE ARCHITECTURE

In order to realize the special requirements of the facility control system, the software architecture will adopt a hierarchical SOA model, which is a typical pyramid model, which is aggregated layer by layer, and the granularity is larger. The software is divided into three layers: device

† email address: drops.ni@caep.cn

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOMPL007

service layer, system service layer and integrated monitoring layer, as provided in Fig. 2.

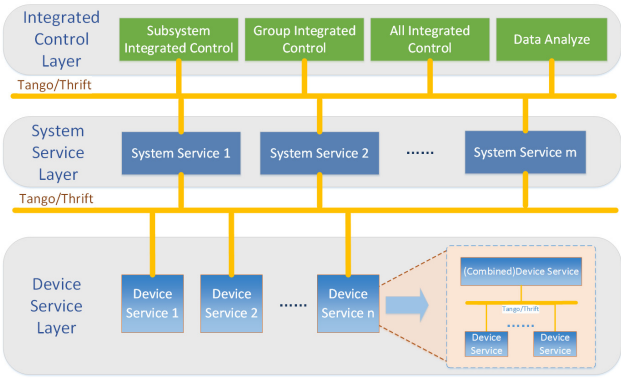

Figure 2: The software architecture of the typical large scientific facilities [4].

The integrated monitoring layer provides a unified integrated operating environment for the control operation of the entire facility, enabling centralized control, monitoring and data management of the entire facility; the integrated monitoring layer can be divided into three different dimensions of integrated control functions according to actual needs: facilities, systems, groups.

The system service layer provides a combined control function for a single system under a bundle group, which is a large-scale control function aggregation of the device service layer, and this layer provides parameter delivery and experimental data archiving.

The device service layer provides software mapping for independent devices to implement device control, status acquisition, device diagnostics, and self-test, using device drivers. Therefore, all heterogeneous devices implement the unification of software interfaces and provide consistent standard services on network protocols.

THE SOFTWARE DEVELOPMENT ENVIRONMENT

The entire life cycle of software development is managed by software quality assurance measures. EA (Enterprise Architect) supporting UML language is used to implement a series of software design operations from requirements analysis to summary design, detailed design, database design, testing, release, and deployment.

The EA is used to analyze and design the requirements in detail, define classes, and model the interfaces and attributes of the software objects. The skeleton code of the device service is generated by the TANGO [5] code generation tool, and the QT integrated development environment is used to fill the personalized code to realize the development of the device service.

The configuration of graphical components, data components and monitoring schemes is achieved through the framework's integrated development tools.

In order to develop cross-platform GUI software, QT software was used for development. Two software middleware, TANGO and Thrift, were chosen.

THE CONTROL SOFTWARE FRAMEWORK DESIGN

The control software framework named iCOFFEE is a collection of interconnected abstract components, as provided in Fig. 3. The framework provides standard models and configuration tools, which effectively improves the reuse of code, reduces the construction cost and software quality of the application software, and provides a basis for long-term maintenance and update of the software. iCOFFEE is more like a factory that can quickly build applications for control systems for specific scientific facilities.

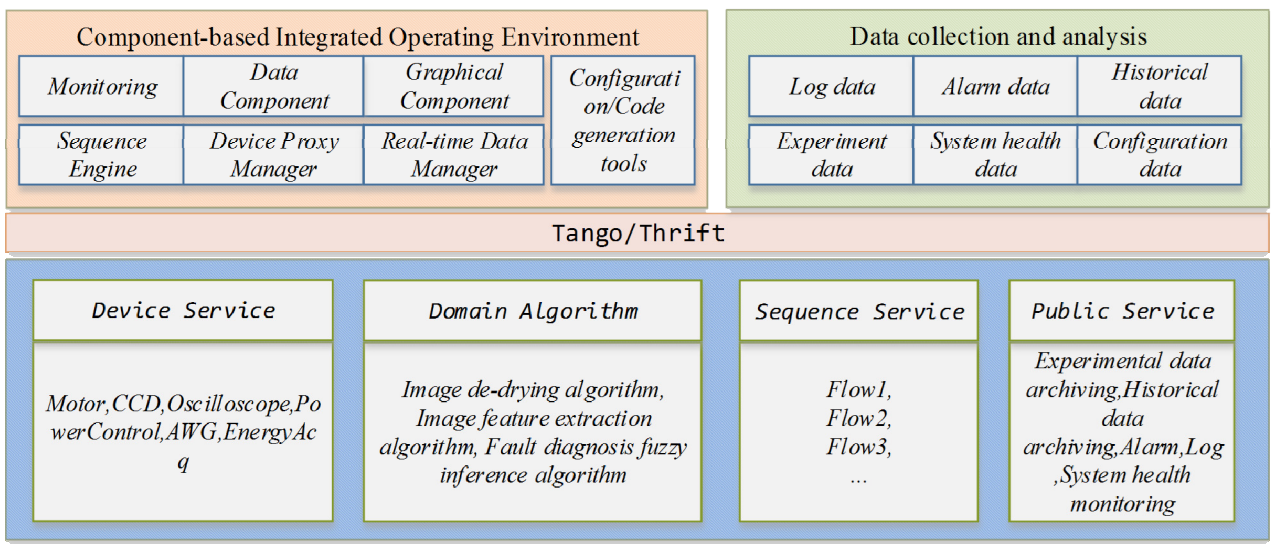

Figure 3: The main components of the iCOFFEE framework.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOMPL007

The control software framework is based on the typical distributed software architecture of the software bus (TANGO/THRIFT) [6], which consists of an integrated runtime environment, a public service, a device service library, a domain algorithm library, and an experimental sequence library. The integrated operating environment is the software portal, which mainly implements monitoring interface configuration, operation and service management. At the same time, application software such as configuration tools, code generation tools, public services and debugging tools can be called in this environment. Public services provide services such as logging, alarming, and data archiving. The software equipment library includes software equipment library, domain algorithm library, and control sequence library, which are continuously built by the integrated development environment and will be more and more over time.

The main components of the framework are described in detail in subsequent articles.

DEVICE LIBRARY BASED ON DEVICE MODEL

Realize software equipment library, domain algorithm library, control flow library and other equipment based on TANGO software middleware. The Thrift-based IDL language defines input and output type devices such as serial servers and OPC [7] servers.

a) Software equipment library: motor, camera, oscilloscope, energy meter, etc.

b) Domain algorithm library: image de-drying algorithm, image feature extraction algorithm, fault diagnosis fuzzy inference algorithm, etc.

c) Experimental process library;

d) Public service library: experimental data archiving, historical data archiving, alarms, log collection, system health monitoring, etc.

COMPONENT-BASED INTEGRATED OPERATING ENVIRONMENT

Monitoring and Control

The application software provides a graphical human interface for the operator, on the console computer or terminal. The human-machine interface is realized based on the framework configuration, which ensures the quality of the code. The configuration of the monitoring and sequence is based on the configuration tools provided by the framework.

Graphical Component

Build a complex graphical display based on simple graphical element definitions, saved as a new graphical component that continues to build more complex graphical components as an element. These components are stored in the graphics library as assets for reuse by new applications.

The update of the status data uses the asynchronous event mode, and the command uses the synchronous scheduling mode.

Data Component

The data component is a scriptable input and output model, and the input and output parameters are in JSON format. You can build complex nested key-value models using four simple types (Integer, Float, Boolean, and String) to continue building more complex data artifacts as a new element. Data components currently only support synchronous scheduling mode, and asynchronous event mode is supported in the future.

Device Proxy Manager

It is responsible for creating and managing multiple device agents. The user does not care about the connection of the device service. It only automatically obtains control capabilities based on the command name. After the execution is completed, the execution result is normal or abnormal. In order to cope with the large number of device services, the device proxy manager is implemented in a multi-process architecture similar to Chrome software, as provided in Fig. 4.

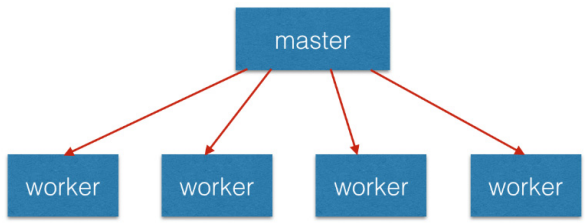

Figure 4: Multi-process architecture.

Real-time Data Manager

The real-time data manager uses the factory mode to create and manage state data. The user does not care about the connection of the device service, and only automatically obtains the state data according to the attribute name, by polling or event mode.

To cope with the proliferation of device services, real-time data manager uses a multi-process architecture similar to Chrome software.

Sequence Engine

The sequence engine implements an ordered scheduling capability of the software device service based on the DAG model to implement sequence loading, starting, resetting, and stopping. The sequence engine supports serial mode, block mode, parallel branch mode, and conditional (abnormal) selection mode. It can implement concurrent execution of no less than 100 nodes, support nesting, and support more than 3 levels of nesting levels. The example of sequence running was shown in Fig. 5.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOMPL007

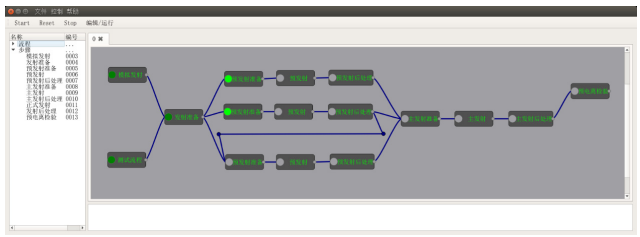

Figure 5: Display of sequence engine.

A sequence consists of one or more atomic steps or sequences in a certain relationship. The constituent elements (sequences) are abstracted into one node. A node is an ordered nested assembly. A node consists of one or more child nodes. There is an ordered relationship between the child nodes, which can be expressed by a directed acyclic graph (DAG). The sequence is designed to be a node class that runs automatically. The sequence is created by the node factory class. The state of the node is marked by the state machine. The automatic execution of the node is implemented through the thread pool. The node information class holds configuration information of the node. Remote scheduling of atomic nodes is implemented by a node execution class.

DATA COLLECTION AND INTELLIGENT ANALYSIS

The data mentioned here includes experimental data, equipment historical data, log data and system health data. The data is collected and stored in a centralized manner and intelligently analyzed by various data services. Various experts use the big data analysis platform according to the needs of their respective fields. Data is used for data mining and analysis in various dimensions. (See Fig. 6.)

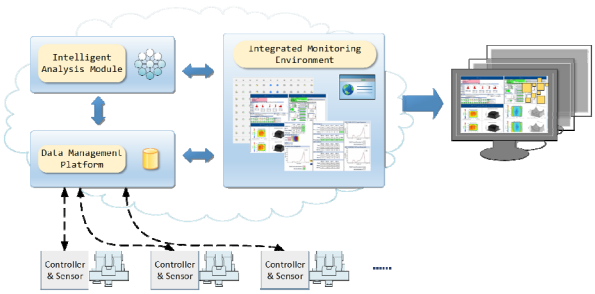

Figure 6: Data collection and intelligent analysis.

Experimental Data Archiving and Historical Data Archiving

Data archiving is divided into experimental data archiving and historical data archiving. [8]

The experimental data is archived to collect and store the experimental data of the device, according to the number of the experiment. Historical data archiving collects data status information during device operation for later analysis of equipment failures and other technical analysis, effectively improving equipment maintenance efficiency and reliability.

Message Log Collection

The message log is automatically generated by the TANGO software, and the service based on Thrift generates a log structure similar to the Tango software. The log uses file local storage, collects logs through open source software, and implements intelligent analysis of logs through intelligent data analysis systems.

System Health Monitoring

Build a system health collection service. It collects the operating data of the system's network, CPU, and memory, and automatically alarms according to the health algorithm. All data is archived through the data archiving service for later big data analysis.

At the same time, the system health monitoring system has a central view of all service states, which can partially amplify the service status view of a certain area and expand layer by layer to evaluate the overall performance of the system [9].

CONCLUSION

Construction of the iCOFFEE incorporates many of the latest advances in distributed computer and object-oriented software technology. Primary goals of the design are to provide an open, extensible, and reliable architecture that is used by many entities and provides long-term maintenance and upgrades. The original intention of the design was to reuse the software and quickly build the application software.

Based on the framework of Tango and Thrift, the framework uses the factory architecture and component technology to continuously make the software reuse to a higher level, and build a big data analysis platform based on data collection.

In the future, we will add artificial intelligence technology to the framework, and do more research and exploration in the acquisition control system and data analysis and processing.

REFERENCES

[1] J. A. Paisner and J. R. Murray, "The National Ignition Facility for Inertial Confinement Fusion", in *Proc. Fusion Engineering*, Nov. 1997. doi:10.1109/FUSION.1997.685664

[2] R. W. Carey, K. W. Fong, R. J. Sanchez, J. D. Tappero, and J. P. Woodruff, "Large-Scale CORBA-Distributed Software Framework for NIF Controls", in *Proc. 8th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'01)*, San Jose, CA, USA, Nov. 2001, paper THAI001, pp. 425-429.

[3] D. J. Yao *et al.*, "Research on Software Architecture of Centralized Control System for High Power Laser Facilities, Computer Engineering and Design", vol. 28, pp. 1737-1740, 2007.

[4] Wikipedia, https://en.wikipedia.org/wiki/Hierarchical_control_system.

[5] https://tango-controls.readthedocs.io

[6] Wikipedia. Remote Procedure Call [EB/OL], http://www.en.wikipedia.org/wiki/Remote_procedure_call, 2011.11.28

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOMPL007

[7] T. Hannelius *et al.*, "Roadmap to adopting OPC UA", in *Proc. 6th IEEE International Conference on Industrial Informatics*, pp.756-761, 2008. doi:10.1109/INDIN.2008.4618203

[8] L. Pivetta *et al.*, "New Developments for the HDB++ TANGO Archiving System", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 801-805. doi:10.18429/JACoW-ICALEPCS2017-TUPHA166

[9] Z. Ni, J. Liu, J. Luo, and X. Zhou, "The Design of Tango Based Centralized Management Platform for Software Devices", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 1121-1124. doi:10.18429/JACoW-ICALEPCS2017-THBPL04