A MULTI-AGENT SYSTEM FOR BUILDING LARGE-SCALE DISTRIBUTED, HIERARCHICAL CONTROL SYSTEMS*

V. Gyurjyan, C. Timmer, D. Abbott, G. Heyes, E. Jastrzembski, B. Moffit, E. Wolin,
Jefferson Lab, 12000 Jefferson Ave. MS-12B3, Newport News, VA 23606, U.S.A.

Abstract

The Multi-Agent Framework for Experiment Control Systems (AFECS) is a pure Java based software framework for designing and implementing distributed control systems. AFECS creates a control system environment as a collection of software agents behaving as finite state machines. These agents can represent real entities, such as hardware devices, software tasks, or control subsystems. A special control oriented ontology language (COOL), based on RDFS (Resource Definition Framework Schema) is provided for control system description as well as for agent communication. AFECS agents can be distributed over a variety of platforms. Agents communicate with their associated physical components using range of communication protocols, including tcl-DP, cMsg (publish-subscribe communication system developed at Jefferson Lab), SNMP (simple network management protocol), EPICS channel access protocol and JDBC.

INTRODUCTION

The CEBAF Online Data Acquisition (CODA) system continues to satisfy the ever growing requirements of the physics programs at Jefferson Lab (JLAB)[1], and demonstrates very good performance and reliability. On the other hand, the control component of CODA has not evolved much over the years partly due to the fact that run control has been dependably performing its simple task of controlling the data acquisition (DAQ) state machine. However, future experiments at JLAB have new expectations from run control, namely:

1. integration of new components into the DAQ system
2. creating feedback between slow controls and DAQ components
3. building expert systems, with their specific state machines
4. designing online data quality monitoring
5. coordination with data production processes
6. putting together online data calibration and distribution systems
7. creating alarm systems, etc.

To satisfy these challenging requirements, the Jefferson Lab DAQ group has developed a framework for building complex, hierarchical control systems. The unique feature which sets this framework apart from conventional control systems is its incorporation of intelligent agent concepts. An agent is a software entity capable of acting intelligently on behalf of a component or user, in order to accomplish a given task. A group of specialized agents cooperate and work together to solve problems that are beyond their individual capabilities.

AFECS provides a group of normative agents for agent management and coordination. These manager agents are responsible for creating and deploying agents on a platform, educating them (based on knowledge provided by the user), distributing them over the network, and recovering them in case of unsatisfactory behaviour. The efforts of these specialized agents ensure control system reliability and robustness.

Different types of "stem cell" agents are also provided by the framework. After creation they are specialized by the control system designer to become representative agents for various real world components of the control system.

DESIGN ARCHITECTURE

An experiment can be thought of as a well-defined set of components, each with their specific behaviors. These behaviors can be represented by simple, finite state machines. Control of the experiment is equivalent to controlling and synchronizing the state changes of those state machines, and AFECS provides the facilities to do this.

AGENTS

Control systems designed using AFECS are composed of agents that are groups of threads in a single process or separate processes running on different machines. The architecture of the AFECS based control system can be seen as a hierarchy of agents, each with responsibility for a component of the experiment. An agent encapsulates control/monitoring algorithms, as well as external interface details of the controlled, real-world component. This provides clear separation of the control and application layers of each component, and seamless integration of legacy software components into the experiment control environment (platform). Agents are active objects, having more than one behavior, and can engage in multiple, simultaneous activities. Their behaviors can be added or removed at run time.

An agent's state is a simplified external view of the current working condition of the component under its responsibility. Each agent is capable of receiving control messages from other agents or the outside world. These messages can cause an agent to change or monitor the state of the physical component. Agents can be organized into hierarchical tree structures that reflect the basic organization of the experiment control system itself. An agent in the tree can have only one supervisor agent and

can supervise many others. At the top of the tree is a single agent (grand supervisor), which represents the overall state of the entire experiment. A control system diagram is shown in Figure 1.

AGENT CONTAINERS

The software architecture of the system is based on the coexistence of multiple JVMs (Java Virtual Machines). Software agents are grouped into virtual clusters or domains according to their specialized functionalities. Agents in the same domain may share a single JVM, which plays the role of a basic agent container. An agent container provides a complete run time environment for agent execution and allows them to concurrently run on the same host. Agents in the same domain may also be distributed over multiple containers.

The Front-End is a special container running the normative agents, taking care of agents' management and overall coordination of the system. It also maintains agents' registry database, and agent discovery facilities.

NORMATIVE AGENTS

The normative administrative (NA) agent (see figure 1) is responsible for agent management and recovery. This agent is in charge of creation, recovery and removal of agents, agent clusters and entire containers. It will also repair any failed agents or containers, thus achieving platform stability and fault tolerance.

The normative registration agent (NR) stores agent's registration information, including agent physical address and description of provided services. This agent is vital during agent discovery processes.

The normative configuration (NC) agent is the interface between a specific control system implementation and the framework. It enables the integration of non-agent software controls into an agent system. This normative agent will coordinate the creation and deployment of each physical component representative agent (A), as well as provide them with knowledge in the form of ontology. Ontology consists of instantiated classes generated from physical component state machine descriptions, which, in turn, are written using the control oriented ontology language (COOL). Agents in the platform can then communicate with each other by exchanging ontology objects and have each of them invoke actions on underlying hardware or software components. AFECS allows dynamic reconfiguration of the control environment in real time by auto-generating or specializing specific control agents whenever new control components are added, or control relationships are changed.

AGENT HIERARCHI

Coordination of groups of agents is accomplished by Supervisor agents (S). They ensure that partial, local solutions to the control problems of a specific domain are integrated into global experiment control. Agents in the hierarchal tree transmit messages between themselves containing commands and status information. Normally, a human operator sends commands to the "grand supervisor" agent, which forwards them to supervisor agents down the tree, who in turn forward them to component agents and so on. The results of the commands are sent back up the tree so that the human operator is made aware of any changes in the state of the system. Any agent in the hierarchy can perform actions on receiving a command and can return the results from that command.

IMPLEMENTATIONS

The new run control system for the JLAB data acquisition system (CODA) has been developed using AFECS [2]. This run control system is designed to configure, control, and monitor Jefferson Lab experiments. It controls data-taking activities by coordinating the operation of DAQ components (readout controller, event builder, event recorder, event transfer, etc.). The graphical user interface (GUI) for run control is intended to view the status of the data acquisition system and its components and to allow the user to control its operation. The GUI was developed not only for general users, such as shift operators, but also to provide DAQ experts the ability to control and debug the system. The run control system can have multiple instances of GUIs associated with a particular experiment. However, only one GUI can be a master, capable of controlling the DAQ system. Figure below shows a snapshot of the GUI in action. The GUI interacts with agents through the cMsg publish subscribe communication protocol [3].

AFECS provides support for bidirectional invocations of web services through java servlets for communicating with agents. AFECS Java servlets use cMsg to communicate with the agent platform of the experiment. The web site http://clasweb.jlab.org/clasonline/rc/hallB/e-cr.htm, dedicated to the development of the CLAS12 Experiment Control System, is an example of the AFECS visualization in action.

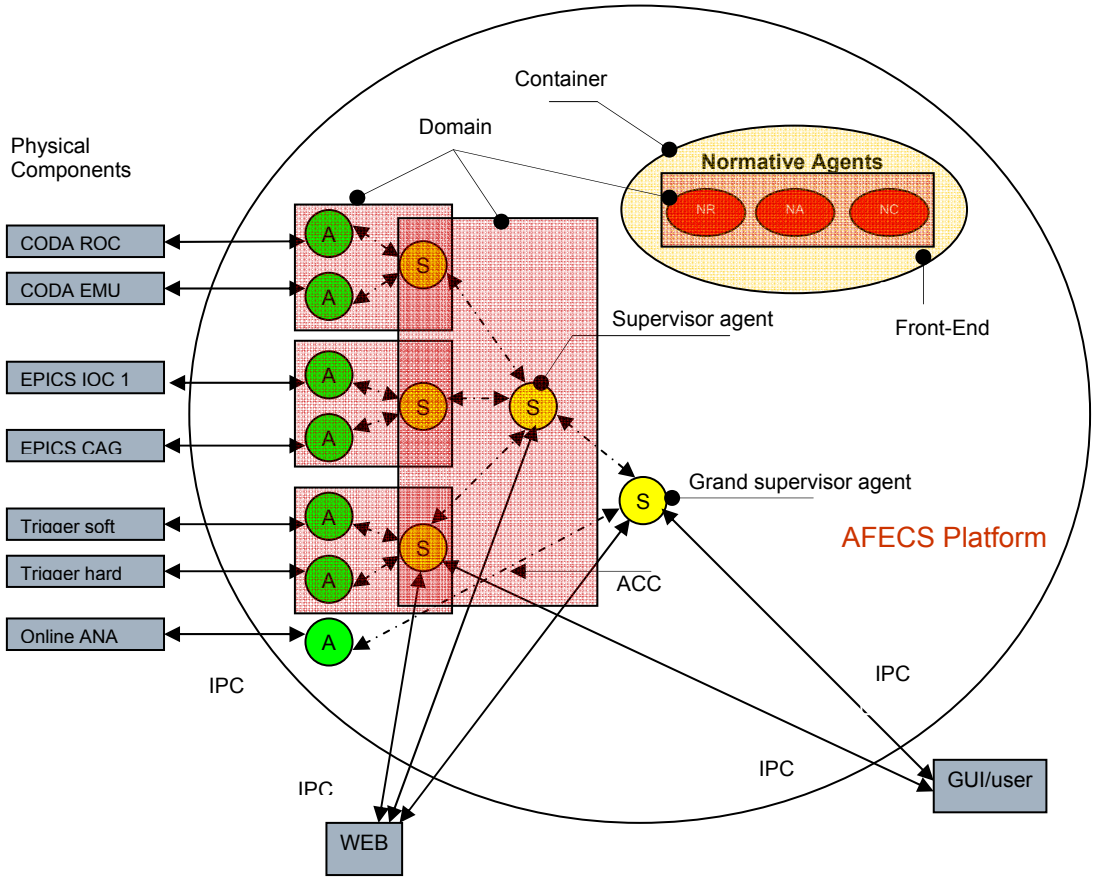

Figure 1: A control system block diagram. IPC – Inter Process Communication protocols (cMsg, DP, SNMP, JDBC, shell), used between agent and physical component. ACC – Agent Communication Channel: cMsg[3] used for inter agent communications. NR, NA, NC – normative agents for platform registration, administration and configuration respectively.

SUMMARY AND CONCLUSIONS

A Java based software framework for designing and implementing hierarchical, distributed control systems has been developed using intelligent agent technologies. The framework encourages abstraction, encapsulation, and overall system modularity.

The AFECS framework provides a special language to describe a hierarchical control structure, as well as control logic and finite state machines.

This framework has been successfully used to develop a new run control system for the JLAB data acquisition toolkit and a CLAS experiment web-based monitoring system.

REFERENCES

[1] Heyes G, et al., "The CEBAF on-line data acquisition system", Proceedings of the CHEP 1994 Conference.

[2] V. Gyurjyan, et al., "Jefferson Lab Data Acquisition Run Control System", Proceedings of the CHEP conference. CERN-2005-002, Volume 1, page 151.

[3] E. Wolin, et al., "cMsg Publish/Subscribe Package for Real-Time and Online Control Systems", Proceedings of the 14th IEEE-NPSS Real Time Conference, Stockholm, Sweden 2005.