DISTRIBUTED DIAGNOSTICS & RECONFIGURATION FOR SHIPBOARD CHILLED WATER SYSTEM

Stephen Chiu, Yi-Liang Chen, Greg Provan, Rockwell Scientific Company, USA
Francisco Maturana, Ray Staron, Ken Hall, Rockwell Automation, USA

ABSTRACT

The ability to automatically reconfigure a ship's engineering plant in response to changing mission or equipment conditions can dramatically increase a ship's capability and survivability. A COTS-based, distributed control architecture for providing mission- and diagnostic-driven reconfiguration of the engineering plant has been developed under a cooperative program between Rockwell and the Office of Naval Research. The architecture is based on the use of software agents distributed among networked automation controllers. Each agent represents a physical resource in the engineering plant (e.g., a chilled water plant, a generator, a pump); the agents cooperate autonomously to create a control plan that can achieve the mission objective based on the available resources/agents. Diagnostic models are embedded in the agents to identify faulty or damaged system components when abnormal operating behavior is detected. After a faulty component is identified, the agents negotiate to generate an alternative control configuration that circumvents the faulty component. Rockwell is working with the Naval Surface Warfare Center Carderock Division to demonstrate this distributed diagnostic and reconfiguration architecture on a chilled water system testbed. The control system reconfigures chilled water routing based on changing mission, equipment, and pipe conditions.

KEY WORDS

Distributed control, distributed diagnostics, agent, reconfiguration, fault tolerant, chilled water.

1. INTRODUCTION

Improving the fight-through capability of future combat ships necessitates an integrated ship control system that can monitor all shipboard engineering assets and adaptively reconfigure these assets during battle conditions. For example, if a chilled water supply unit is damaged during combat, a ship control system may temporarily overload another chiller to provide the required cooling capacity or shed non-combat heat loads to reduce cooling requirements. If the chilled water system still cannot satisfy combat cooling demand, the ship control system may selectively reduce the performance of combat systems to avoid complete system shutdown. To achieve such intelligent ship responses, all the various shipboard subsystems (e.g., chilled water system, firemain, propulsion, generator, combat system) must be tied together under an integrated control hardware and software architecture. To provide damage tolerance, the integrated ship control system must also be a physically distributed architecture with intelligent autonomous behavior embedded in low-level system components, such that damages to the control network or to a limited number of control components will not cause overall system failure.

Under an Office of Naval Research (ONR) cost-shared Dual Use Science & Technology program, Rockwell has developed a highly distributed architecture for intelligent control based on Rockwell Automation's commercial-off-the-shelf (COTS) industrial control hardware. The use of COTS hardware provides the added benefits of reduced cost, increased procurement flexibility, and global access to spare parts and support personnel from Rockwell Automation's global support network.

A highly survivable ship control system must have the often conflicting characteristics of both an integrated system and a distributed system; i.e., subsystems must work cooperatively in an integrated manner during system reconfiguration, but must work autonomously when communication with other parts of the system is lost. To this end, we have developed a hierarchical, distributed software architecture based on agent technology. Agents are self-contained software entities capable of communicating and making individual decisions [1]. They work autonomously, handle goals, maintain beliefs, and cooperate through negotiation to generate solutions. Important standardization work for the use of agent technology in industrial automation has been carried out by the Holonic Manufacturing Systems (HMS) consortium in the early 1990's [2]. This research has provided important results to build inexpensive, expandable, and fault-tolerant industrial control systems.

In order for agents to respond correctly to system damages, the agents must first accurately identify the damaged system components (e.g., a specific pump, valve, pipe section). We utilize model-based diagnostics [3] in the agents' decision logic to help pinpoint faulty/damaged components in the system. For large, complex systems, model-based diagnostics is more tractable compared to rule-based diagnostics. Model-based diagnostics involves creating a rigorous description of how a system works based on first principles. In particular, relationships among system components are explicitly represented in the model. Given a model, model-based diagnostic algorithms can guarantee soundness and completeness of all diagnoses generated. In addition, the diagnostic model can be easily updated to track changes in the physical system while guaranteeing that the overall diagnostic system always remains self-consistent.

Previous model-based diagnostic systems do not support true distributed processing. Diagnostic evaluation is performed either on a large model residing in a centralized processor or on carefully partitioned independent subsystem models residing in local processors (i.e., creating "islands" of diagnoses). Diagnoses based on a large, centralized model typically provide more accurate result because the centralized model captures the interrelationships among subsystems, enabling it to "see the big picture" when tracking down a faulty component. However, such a centralized diagnostic system presents a single-point-of-failure. To address this problem, we have developed distributed diagnostic software that allows a large system model to be arbitrarily partitioned into sub-models residing in different processors; diagnostic information is propagated among sub-models, allowing the distributed system to piece together the global model for accurate diagnoses. In addition, failure of a few local processors will not disable the overall diagnostic system, but only result in reduced certainty of the diagnoses.

In the following sections, we describe the autonomous cooperative agent and model-based diagnostic methodologies, and the integration of these two methodologies to create a highly distributed, intelligent control architecture using Rockwell Automation's COTS industrial control hardware. We also discuss the application of this control architecture to a chilled water system testbed, in which the chilled water routing is adaptively reconfigured based on changing mission, equipment, and pipe conditions.

2. DISTRIBUTED AGENTS

Our goal is to develop a COTS-based distributed control architecture that can provide integrated health monitoring and control reconfiguration for essentially all engineering systems onboard a ship. The architecture must provide fault/damage tolerance and be easily scalable to include additional ship systems. Several levels of control automation exist in this architecture, as shown in Figure 1. These levels are:

1. Mission Level – This is the highest level in the control architecture; the mission-level controller sets priorities and performance objectives for all ship engineering systems based on the overall mission goals. For example, in a damage control situation, resources would be shifted from less critical services, such as drinking water production, to fire-fighting systems. Similarly, control and use of the ship's resources for the various system functions can be optimized for dockside, normal steaming, battle stations, and damage control conditions.

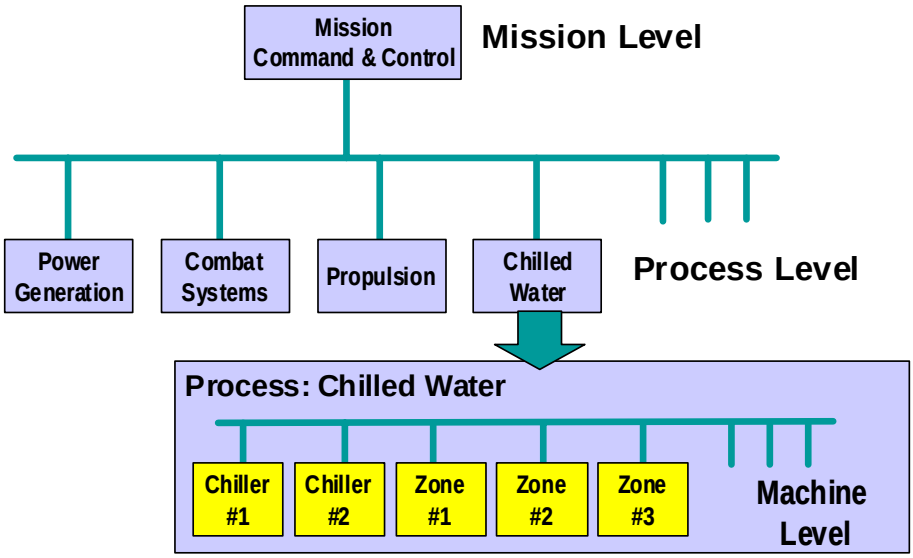

Fig. 1. Control system hierarchy is composed of three levels.

2. Process Level – Each process-level controller provides a general ship service (e.g., chilled water service, propulsion service, electrical power service), and will reconfigure its system operation based on the performance objectives set by the higher level (i.e., mission-level) controller. For example, the chilled water service may temporarily overload a water chiller unit when it senses another unit is down, in order to maintain the cooling rate objective set by the mission-level controller. This will result in optimized machine and energy use, and automatic reconfiguration of resources to maintain service availability in the event of faults or damage.

3. Machine Level – Controllers at this level monitor and control individual machine units (e.g., a chiller) to maintain machine availability and achieve the machine setpoints requested by the process-level controller. For example, if excessive pump vibration is detected, the machine-level controller may alter the pump speed to avoid exciting a critical frequency and thus avoid damage.

In the agent-based control system, each agent (called an Autonomous Cooperative Unit, or ACU) represents a physical process or equipment and coordinates its operation with other agents. Control actions are the result of coordinated decisions among all ACUs. Figure 2 shows the control software architecture, composed of a collection of mission-level ACU, process-level ACUs, and machine-level ACUs. The ACUs at each level receive task commands from a higher level ACU; each ACU then translates its commanded task into a series of sub-tasks for lower level ACUs. The lower level ACUs capable of performing a specified sub-task bid on the sub-task, analogous to subcontractors bidding to participate in a large project. The relative "cost" for an ACU to perform a particular sub-task depends on the ACU's performance capabilities and its current duties. (As an analogy, a subcontractor bids a price based on the difficulty of the job relative to his skills and on whether he already has a heavy workload.) The control plan with the lowest overall cost is then selected for execution. Note that in this agent-based control architecture, a damaged equipment or controller will simply not participate in the bidding process.

Machine ACUs contain self-awareness and self-assessment capabilities. The interaction of Machine ACUs is founded on the creation and dissolution of *virtual control clusters*. A virtual control cluster consists of Machine ACUs that may potentially take part in carrying out a specific task; formation of the virtual control cluster involves tracing through the physical relationships among Machine ACUs. For example, the task of cooling a specific heat load may potentially involve certain chillers, pipes, and valves associated with different water flow paths to the heat load. The ACUs that represent these chillers, pipes, and valves would form a virtual control cluster and would provide individual bids for taking part in the task. The lowest cost solution (i.e., the flow path that involves the lowest cost chiller, pipes, and valves) then emerges from the virtual control cluster.

When a requested task (or performance goal) cannot be accomplished, Machine ACUs negotiate to obtain several alternative solutions that are close to meeting the performance goals while satisfying both local and shared constraints. The sequence of communication is as follows: Machine ACUs provide Process ACU with a number of solutions to be prioritized according to goals and constraints. Process ACU then uses its private goals to select the best solution. In the same way, when a process-level goal cannot be achieved, Process ACUs also negotiate and provide the Mission ACU with a number of alternative solutions.

Cooperative software agents permit the creation of highly flexible control systems based on both autonomy and cooperation. The ACU framework is well-suited for solving resource allocation problems on distributed

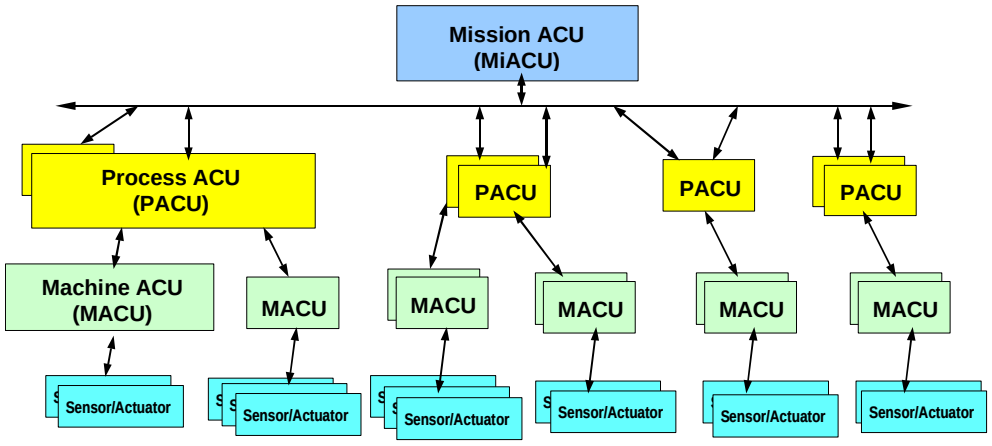

Fig. 2. Hierarchical software architecture showing three ACU levels, corresponding to Mission, Process and Machine. The machine level ACU interacts with sensors and actuators.

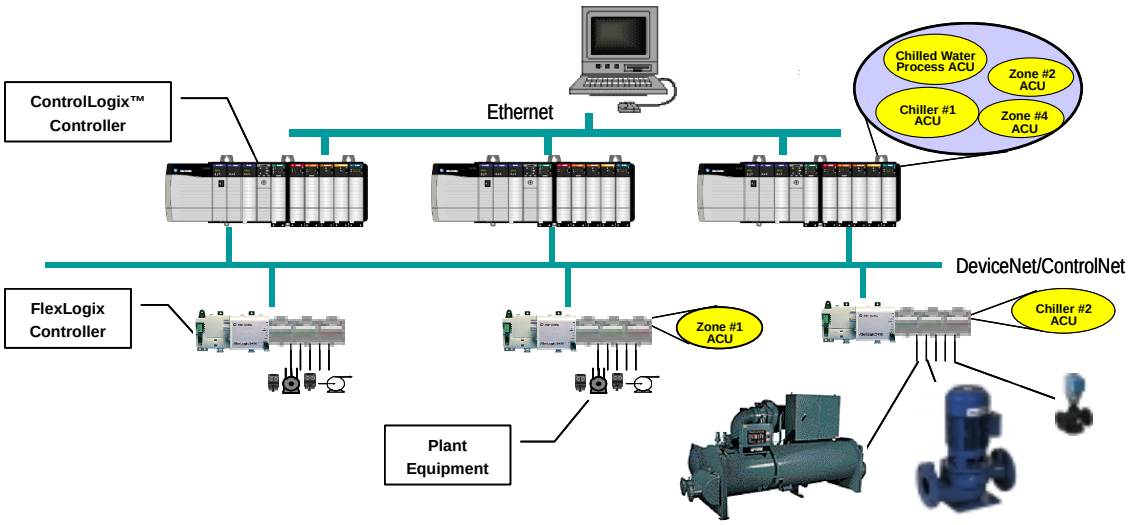

Fig. 3. Mapping of ACUs onto networked control hardware.

hardware. The resultant software is easily scalable to large systems. When a new machine is added to a system, the overall software is updated simply by adding a corresponding agent to represent the new machine. Autonomous capability is directly designed into an agent so that it can continue to operate in the event of loss of communication or loss of other agents.

The ACUs will reside in Rockwell Automation's Logix family of automation controllers, which includes the high-end ControlLogix controller suitable for mission- or process-level control and the FlexLogix controller suitable for machine-level control. SoftLogix, which is a software package that allows control programs to run on a PC, may also be appropriate for mission-level control. The mapping of ACUs into a physical hardware architecture is shown in Figure 3. Each ControlLogix controller can accommodate multiple ACUs while each FlexLogix controller may accommodate only one or two ACUs due to memory constraint.

Figure 4 shows the agent software structure in a controller, which consists of three main components: a data table, middleware, and one or more ACUs. The data table contains input/output data for sensing/controlling the devices attached to the controller hardware as well as data for the embedded applications. The middleware supports the agent system services, enabling inter-agent communication and collaboration across networked hardware. The ACU provides application-specific decision logic and contains the following application components:

1. an equipment cost model, which computes the cost of operating the equipment at specific equipment settings;
2. a diagnostics engine, which computes the health status of the associated machine or process;

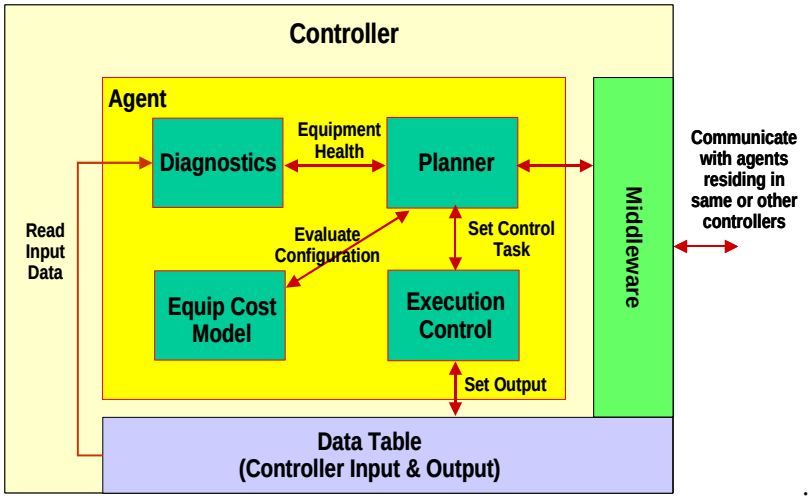

Fig. 4. Agent software structure in a controller.

3. a planner, which computes appropriate control reconfiguration plans (i.e., low cost plans that meet specified constraints) whenever a fault condition arises or system goals change;

4. an execution control module, which controls the attached devices and can execute the reconfiguration control plans.

The agent middleware provides a unified communication service between agents, appropriately routing messages regardless of whether the agents reside in the same controller or different controllers. In addition, the agent middleware handles agent activation (instantiation) and registration. When an agent is instantiated, it registers itself in a global "Yellow Page" directory that provides a listing of agents, their capabilities and their locations. As a particular task arises, agents with relevant capabilities are found in the global directory to form a cooperative group to handle the task.

To prevent having a single point of failure, the agent software infrastructure supports distributed, redundant directories. When an agent registers its location and capabilities with a directory, this information is automatically propagated to copies of the directory residing on different controllers. When a directory fails to respond to a request from an agent, the request will be automatically routed to another directory.

3. MODEL-BASED DIAGNOSTICS

While any type of diagnostic algorithm can be inserted into an agent's diagnostic engine, our approach is to use model-based diagnostics. Compared to rule-based systems, model-based diagnostic systems can better guarantee the completeness of the diagnoses and are easier to maintain. Specifically, we employ causal-net models for diagnostics.

A causal-net model consists of interconnected component models in which each component has various operational modes; for example, a valve component may have the following operational modes: normal, stuck open, and stuck closed. An input/output behavior is defined for each mode of the component; for example, if a valve is in normal mode, then the output pressure is equal to the input supply pressure multiplied by the valve position.

A simple pump loop and its corresponding causal-net model are shown in Figure 5. The causal-net model takes equipment settings (or commands) as input, and simulates the machine or process behavior with all components assumed operating in normal mode. The causal-net model thus computes the values of the expected sensor readings by propagating the equipment setting values through the model. These computed sensor values are compared with the actual sensor readings. If there is a discrepancy, the diagnostic engine is invoked to determine the most likely cause of this discrepancy. During diagnostics, the actual sensor values are propagated through the causal-net model to determine the faulty component and its operational mode that would produce the abnormal sensor values. When multiple diagnostic solutions are found in the model (i.e., when the observed abnormal behavior can be explained by several different failure conditions), the best solution can be determined by evaluating probabilities associated with the various component failure modes.

Causal-net models can be created from a library of component models by interconnecting the components. For example, the pump loop shown in Fig. 5 is composed of 6 component types: motor/pump, pipe, valve, tank, pressure sensor and flowmeter. We have developed software tools for defining component models, interconnecting the components to form causal-net models, and generating embeddable C code from the causal-net models.

Our model-based diagnostic engine is an integrated component in an agent, and uses the agent middleware for communication between subsystem models residing in different agents/controllers. This framework supports true distributed diagnostics, in which a large, complex system model can be distributed among multiple agents residing in multiple automation controllers. A subsystem model residing in a local controller can communicate with other subsystem models to piece together system-wide health information that leads to more accurate diagnoses for local equipment compared to using isolated local diagnostic models.

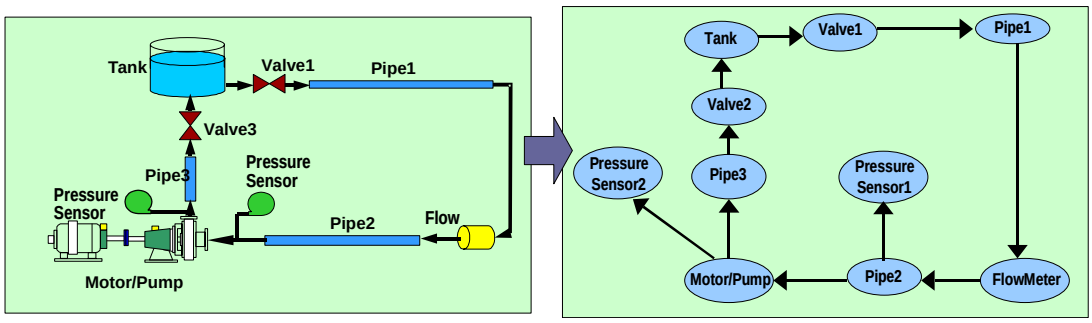

Fig. 5. Pump loop system and its corresponding causal-net model

4. SOFTWARE TOOLS

A set of software tools is needed to allow an engineer to easily program intelligent agents, translate the agent description into executable code, and download the code into the appropriate automation controller for execution. We have developed a PC-based environment in which a user can define agent types and their attributes, capabilities, and behaviors, thus establishing a reusable library of agent types. The user can then easily build control software for similar applications from the library of agent types. For example, once a library of agent types for controlling a chilled water system has been established, building the control software for various chilled water systems that have different number of pumps, valves, and loads can be easily accomplished by instantiating the appropriate number pump, valve, and load agents from the library of agent types.

Figure 6 shows the PC-based environment as the user builds a group of agents by creating instances of several agent types. The left side panel lists the name of each agent, while the right side panel lists the attribute parameters of a selected agent. The user can click on each agent to customize their individual attribute parameter values; for example, the attribute parameters of a load agent may include the load's desired temperature range and temperature data sample rate.

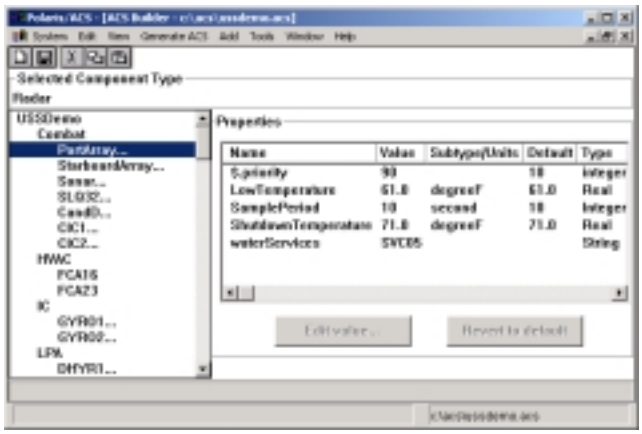

Fig. 6. PC-based agent software development environment. A user can quickly create a group of agents for controlling a specific system from a library of agent types.

After a group of agents for an application has been built, the user can then proceed to assign the agents to the available automation controllers through a graphical interface. Software tools are then invoked to translate each agent description into C source code, compile the source code into object code, and link together, for each controller, all the code assigned to that controller. Finally, Rockwell Automation's standard COTS software tools are invoked to download the code from the PC to the assigned controllers across the network.

A separate PC-based software environment exists for building causal-net diagnostic models. Analogous to the agent development environment, the user first establishes a library of component model types (e.g., pump, valve, and pipe model types) that each contains customizable attribute parameters. A system or subsystem diagnostic model is built by creating instances of component model types, setting the appropriate attribute parameter values for each component model, and then graphically interconnecting the component models according to their physical relationships. The software environment for building causal-net diagnostic models is shown in Figure 7. Diagnostic code can be automatically generated from the causal-net model graph and downloaded into a controller as part of an agent's code. Agents that have associated diagnostic code are marked by including diagnostic tasks in their list of capabilities.

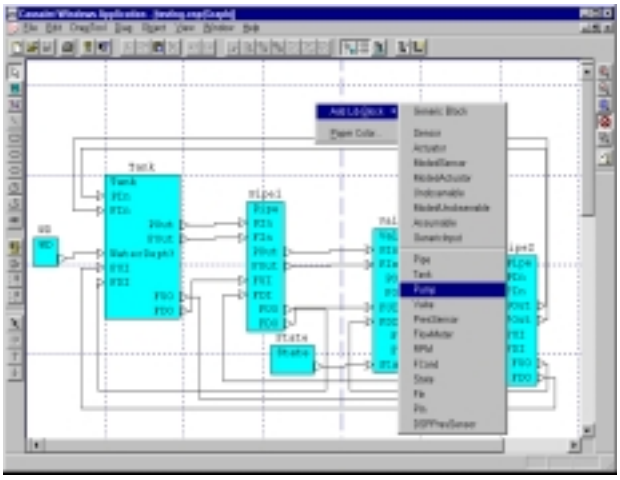

Fig. 7. PC-based diagnostic software development environment. A user can build a system or subsystem diagnostic model by creating instances of component model types from a library and then interconnecting the component models.

We have also developed a PC-based control system debugging tool called the "Sniffer"; this tool monitors inter-agent communications across the control network and displays the agents' interactions in a clear, graphical form.

5. CHILLED WATER SYSTEM APPLICATION

The agent control architecture has been applied to a chilled water system testbed at the Naval Surface Warfare Center Carderock Division (NSWCCD) in Philadelphia. The NSWCCD testbed is a reduced-scale model of a shipboard chilled water system; it delivers cooling water to various heat loads and is built with redundant chillers and flow paths to allow for system reconfiguration in case of equipment failure. A simplified diagram of the NSWCCD chilled water system is shown in Figure 8. Two chillers,

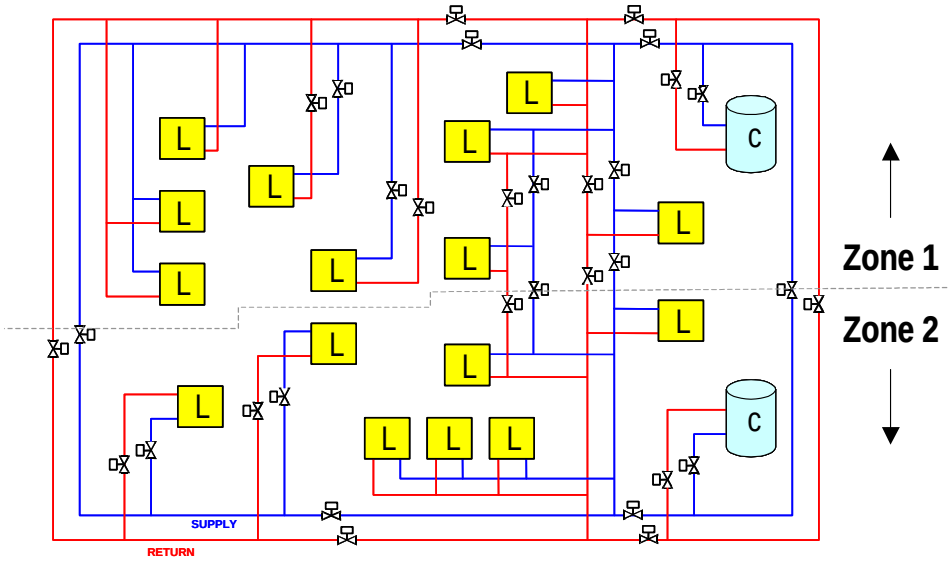

Fig. 8. Simplified diagram of chilled water system testbed at NSWCCD. C = Chiller, L = Load.

marked by "C", can supply cooling water to any heat load in the system. The heat loads (marked by "L") are divided into three types: non-vital, vital, and combat system heat exchanger. The loads are located in two different physical zones (port and starboard zones), and it is desirable to segregate these two zones during battle, with one chiller dedicated to cooling one zone. The main objective of control reconfiguration during battle is to maintain delivery of cooling water to combat systems regardless of pump failure, valve failure, or pipe damage. Under certain damage conditions (e.g., when one chiller fails) there is no way to deliver cooling water to all combat systems while maintaining zone segregation. The control system will open segregation valves to allow cooling water to flow across the two zones when absolutely necessary.

To achieve highly distributed control with intelligence embedded at the component level, the control system for this application consisted of 68 agents distributed across 22 automation controllers (2 ControlLogix and 20 FlexLogix controllers). The chilled water system testbed with installed automation controllers is shown in Figure 9. The agents in the control system include a ship agent (which provides mission-level control), ship service agents (which are process-level agents representing chilled water system, combat system, power system, HVAC system, etc.), zone agents, chiller agents, load agents, valve agents, and pipe agents. The distribution of intelligence across a large number of agents/controllers provides a high level of damage tolerance to the control system.

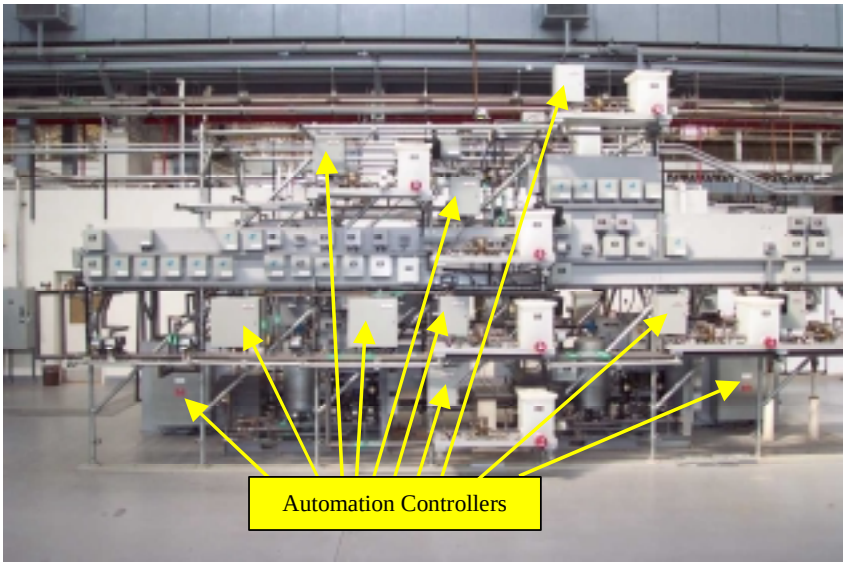

Fig. 9. Chilled water system testbed at NSWCCD with automation controllers installed.

We have successfully demonstrated control reconfiguration of the chilled water system for a set of scenarios that includes:

1. Automatic segregation and desegregation of chilled water service zones based on ship's operating mode.
2. Single valve failure.
3. Combined single chiller pump failure and single valve failure.
4. Single pipe segment breakage.

In fault/damage scenarios, diagnostics and reconfiguration are typically initiated by load agents that observe abnormal conditions (e.g., load temperature is sharply rising and soon to exceed temperature limit). The alert triggers distributed diagnostics to identify the faulty/damaged equipment, followed by distributed planning of a new system configuration that can accomplish the required tasks without the faulty/damaged equipment.

6. CONCLUSION

We have developed and demonstrated a highly distributed, intelligent control system for providing damage-tolerant reconfiguration of a ship's engineering assets. This advanced control system was implemented by integrating Rockwell Automation's COTS industrial control hardware and software with new agent software and diagnostic software. The use of agent technology supports flexible control behavior that is both autonomous and cooperative, and provides an architecture that is easily scalable to large systems. The agents' collaborative decision-making mechanism based on bids and negotiation allows the control system to inherently adapt to loss of equipment and controllers. Model-based diagnostics is embedded in the agents to ensure equipment damages are accurately identified. Both the agent technology and model-based diagnostic technology promote software reuse, enabling similar system applications to be quickly constructed from libraries of agent types and component model types.

This highly distributed, intelligent control architecture was successfully demonstrated on a chilled water system testbed at the Naval Surface Warfare Center Carderock Division. We expect this new generation of COTS-based shipboard automation systems with reconfigurable mission- and diagnostic-driven control will significantly improve the sustainability and survivability of Navy ships.

7. ACKNOWLEDGMENT

Financial support for this work from the Office of Naval Research and Rockwell Automation is gratefully acknowledged. The authors also thank Mr. Don Dalessandro, Mr. Mike Zink, and Mr. John Overby at NSWCCD for their technical assistance in the chilled water system demonstration; Mr. Pavel Tichy, Mr. Petr Šlechta, and Mr. Sivaram Balasubramanian at Rockwell Automation for co-developing the agent software; and Mr. Kirk Cerise at Rockwell Scientific for integrating model-based diagnostics into the agent software.

REFERENCES

[1] M. Wooldridge and N.R. Jennings, Intelligent Agents: Theory and Practice, *Knowledge Engineering Review*, 10(2), 1995, 115-152.

[2] J.H. Christensen, Holonic Manufacturing Systems: Initial Architecture and Standards Direction, *Proc. First European Conference on Holonic Manufacturing Systems*, Hanover, Germany, 1994.

[3] G. Provan and Y.-L. Chen, Component-Based Modeling and Diagnosis of Process Control Systems, *Proc. 1999 IEEE Symposium on Computer Aided Control System Design (CACSD)*, Kohala Coast, Hawaii, August 1999, 194-199,

BIOGRAPHY AND CONTACT INFORMATION

Stephen Chiu is the Rockwell Program Manager for the ONR/Rockwell cost-shared program to develop reconfigurable shipboard automation systems based on COTS hardware. He is also Manager of the Control and Power Systems Department at Rockwell Scientific. Mr. Chiu received a double BS in Mechanical Engineering and Nuclear Engineering from UC Berkeley (1983) and an MS in Mechanical Engineering from MIT (1985). He joined Rockwell Scientific in 1985, where he conducted research in intelligent control, data mining, and robotics, and has held program management responsibilities in the control and electric actuation area for the past 7 years. He can be reached at schiu@rwsc.com.

Yi-Liang Chen is Research Scientist in the Diagnostics and Modeling Dept. at Rockwell Scientific. Dr. Chen holds a Ph.D. (1997) and an MS in Electrical Engineering from the University of Michigan, Ann Arbor; and a BS in Electrical Engineering from National Taiwan University. He is recognized through his works in various areas of Discrete Event Systems (DES), which include modeling, control, diagnosis, conflict resolution, and auto-reconfiguration. Since joining Rockwell Scientific in 1997, he has contributed to projects in modeling of DES, enterprise control and diagnostic program generation, distributed diagnostics, and condition-based maintenance. He can be reached at ylchen@rwsc.com.

Gregory Provan is Manager of the Diagnostics and Modeling Department at Rockwell Scientific. He holds a Ph.D. in Mathematics from the University of Oxford (1990), an MS from Stanford University and a BSE from Princeton University. Dr. Provan is recognized as one of the leaders in model-based diagnostics technology, and has been involved in the design of Rockwell Scientific's causal-net diagnostic system. Prior to joining Rockwell

Scientific in 1995, Dr. Provan was on the Computer and Information Science faculty at the University of Pennsylvania. He can be reached at gprovan@rwsc.com.

Francisco Maturana is Senior Research Engineer at Rockwell Automation's Advanced Technology division. He received a Ph.D. in Intelligent Manufacturing Systems from the University of Calgary in 1997, where he conducted research in distributed artificial intelligence for manufacturing applications, including federated organizations (Mediator agents), multi-agent learning and self-organization. At Rockwell Automation, he developed coordination protocols for autonomous agents for the rod mill steel industry and leads the agent software development effort. Dr. Maturana's current research focuses on distributed control architectures and autonomous cooperative system for continuous and discrete industry and production control. He can be reached at fpmaturana@ra.rockwell.com.

Ray Staron is Senior Principal Engineer at Rockwell Automation's Advanced Technology division. Dr. Staron received a B.S in electrical engineering from Case Western Reserve University (CWRU) in 1974, an M.S. from MIT in 1976, and a Ph.D. in computer science from CWRU in 1993. His interests are in the areas of control algorithms (including distributed control), object-oriented analysis and design, and software development. During his 23-year employment with Rockwell Automation, Dr. Staron has investigated and worked on numerical controllers, control networks, several families of programmable controllers, and advanced programming and configuration software for multi-controller distributed systems. He can be reached at rjstaron@ra.rockwell.com.

Ken Hall is Vice President of Architecture and Systems Development at Rockwell Automation's Advanced Technology division. He received his BEE from Cleveland State University Fenn College of Engineering in 1974. He has been the principal architect for all of Allen-Bradley and now Rockwell Automation's large Programmable Controllers Systems. His interests are in the development of large complex systems and how to partition functionality between hardware/ASICs, real-time embedded control software and supervisory and configuration software for large control systems. He is responsible for defining research direction for automation control systems and directs the research activities at Rockwell Automation Advanced Technology Labs in Cleveland and Prague. He can be reached at khhall@ra.rockwell.com.

Presented at the Thirteenth International Ship Control Systems Symposium (SCSS) in Orlando, Florida, on 7-9 April 2003.