Retrofitting Cyber Physical Systems for Survivability through External Coordination

Kun Xiao, Shangping Ren
Computer Science Department,
Illinois Institute of Technology
{kxiao3, ren}@iit.edu

Kevin Kwiat
Information Directorate
Air Force Research Laboratory
kwiatk@rl.af.mil

Abstract

Most Supervisory Control and Data Acquisition (SCADA) systems have been in operation for decades and they in general have 24x7 availability requirement, hence upgrading or adding new fault tolerant logic into the systems to sustain faults caused by cyber attacks when these systems evolve into a cyber environment is often difficult to achieve. In the proposed approach, an external coordination layer is constructed that only interfaces with the SCADA systems through events and separate from the process under control. The coordination layer is a combination of transparent management of fault-tolerant schemes of critical services of a SCADA system and a model for coordinating different critical services when faults caused by cyber attack occur in that system. In addition, security-related knowledge, such as cyber attack patterns and potential fatal states, etc., are also modeled and built into the coordination layer. The advantages of our approach are twofold: (1) the survivability-related knowledge and protection scheme are built in the coordination layer which is external to the SCADA systems and therefore the disturbance to the underlying systems is greatly reduced; (2) "separation of concern" principle is truly reflected in our model in that fault-tolerance, security and survivability concerns are separated from supervisory and acquisition. In addition, the external coordination model will enable us to accommodate future requirements that may not even be anticipated today.

1. Introduction

The Supervisory Control and Data Acquisition (SCADA) system is a specialized software package positioned on top of hardware that needs to be monitored and controlled. SCADA systems perform important roles in many of the nation's critical infrastructures, ranging from electric power generation, transmission, and distribution to railroads and mass transit [1]. In general, these infrastructures have two layers:

1. **Physical layer.** This layer consists of physical units and infrastructures, such as power distribution unit, plumbing, wiring, etc. that are used to deliver essential services.
2. **Cyber layer.** This layer contains computers, networks and data gathering sensors that are used to monitor and control the physical layer. The SCADA system is the main part of this layer.

Both the SCADA systems and the underlying physical systems have strict survivability requirements on a twenty-four-hours-a-day, seven-days-a-week (24x7) basis. Here survivability means the capability of a system to fulfill its mission in a timely manner, even in the presence of attacks, failures, or accidents [2]. Different from fault-tolerant systems which are generally engineered to tolerate random natural failures, system survivability must also consider unpredictable faults which may be caused by intentional attacks.

SCADA systems are developed to monitor and estimate the current operation state [9], collect, analyze, and diagnose fault alarms [10], as well as use redundant techniques to provide fault tolerance [11] for underlying physical systems. However, most existing SCADA systems themselves become a point of vulnerability when they evolve into a cyber environment. The available security technologies unfortunately are not targeted for protecting SCADA systems, and there are some misconceptions [3] as follows:

1. SCADA system resides on a physically separated and stand alone network.
2. Connections between SCADA systems and other corporate networks are protected by strong access control schemes.
3. SCADA systems require special knowledge, making them difficult for network intruders to access and control.

4. In the underlying physical layer, all fault alarms are assumed to be caused by hardware or software malfunctions, and can be treated by common fault tolerance techniques.

In recent years, operators of those critical infrastructures have come to realize the benefits of sharing SCADA information with corporate networks. However, the ability to access and control processes once isolated to standalone networks has rendered them vulnerable to cyber attacks from a variety of sources, including hostile governments, terrorist groups, disgruntled employees, and other malicious intruders. The 2003 incidence where a disgruntled Australia engineer released tons of dirty water upon city grounds to gain revenge against his supervisor is an example [25].

Most of national infrastructures, such as power grids, water management and supply systems, are built decades ago. These infrastructures have gradually evolved into cyber systems and have been enjoying the flexibility and productivity that modern technology, such as the Internet, has brought. However, the side effects and risks associated with these technologies in this very special area are nevertheless not fully addressed.

One of the main challenges is that these systems have a 24x7 availability requirement that inhibits the 'shutdown and upgrade' approach that otherwise is an effective way to handle emerging concerns. Furthermore, such a high availability requirement makes these systems highly sensitive to changes. These adversary properties of the SCADA systems hence require that any QoS enhancement must be done through a non-intrusive way. In addition, unlike traditional fault tolerance measures with which the central control and administration are sufficient, survivability in a cyber environment must address highly distributed, dynamic and unbounded environments that lack central control and unified policies [29].

To overcome this challenge and ensure software system dependability in cyber environments, a model that captures the characteristics of the system and the environment becomes essential. As critical information systems emerge from "closed castle" into distributed paradigms, the co-operation among distributed elements which compose of the larger cyber systems inevitably becomes the focus of such systems.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 presents our solution for improving survivability of SCADA systems in detail. Section 4 performs a case study to further illustrate the use of our approach. Section 5 draws conclusions and points out our future work.

2. Related Works

Research and continuous re-evaluation of standard practices have been conducted to study ways of improving the survivability of critical infrastructures where errant or malicious computer operations could result in a catastrophe. However, few of them demonstrate a non-intrusive approach focusing on cyber attacks in SCADA systems by integrating domain specific security knowledge into survivability solutions.

Pollet proposes a Network Rings of Defense model to provide a layered security strategy for the SCADA system [4]. In such a structure, developing an appropriate SCADA security strategy involves analysis of multiple layers including firewalls, proxy servers, operating systems, application system layers, communications, and policy and procedures. Risk analysis are applied on all these layers and known vulnerabilities, such as password, key stroke logging, and Denial of Service (DoS) attack protection, etc [5].

An agent-based system is proposed to monitor the SCADA system in a distributed way to provide quick local fault recognition and response [7]. Firewalls [6] and intrusion detection techniques are also studied to help repel and localize cyber attacks [8].

Protection-Shell [17], also known as a Safety Kernel [15,16], is "an independent computer program that monitors the state of the system to determine when potentially unsafe system states occur or when transitions to potentially unsafe system states may occur. The Safety Kernel is designed to prevent the system from entering the unsafe state and return it to a known safe state." Leveson et al. [20] describe the term "Safety Kernel" as a technique which focus on centralizing a set of safety mechanisms. These mechanisms are used to enforce usage policies that are established in a given system to ensure system safety. Kevin G. Wika and J.C. Knight gave an evaluation of the feasibility of the safety kernel as a software architecture for the enforcement of safety policies [15].

System Fault-Tree Analysis [17, 26] is a widely used safety analysis technique and also an important technology in assessment of the safety-critical systems. System Fault-Tree Analysis helps to make fault dependability predictions, and identify root causes of equipment failures. Although different versions of software replications on different hardware units are used to tolerate both hardware and software faults, the management of these replicas in a distributed environment is intertwined with the functional logic being protected.

Until today, most of research efforts have focused on applying available general purpose IT security technologies to SCADA systems. Little effort has been put on developing SCADA-specific strategies. One of the major characteristics of SCADA systems is that it could take a decade or more to renovate the existing SCADA systems to take full advantage of general IT security technologies, but on the other side, these legacy systems still have a considerable amount of serviceable life remaining [8]. Hence, compensating and non-intrusive approaches for improving legacy systems survivability in a cyber environment must be sought-after.

Exogenous control-driven coordination models, such as ARC[ren-coord06], ABT [14], LGI [15], ROAD [16], IWIM [11] and CoLaS [17] isolate coordination by considering functional entities as black boxes. For example, in the ARC model, QoS constraints are mapped into coordination constraints and are enforced through message manipulations which are transparent to the underlying computations modeled as asynchronous message passing systems. The ABT model and its language Reo [14, 18] extend the IWIM by treating both computation and coordination components as composable Abstract Behavior Types (ABT). Similarly to IWIM, ABT is a two-level control-driven coordination model where computation and coordination concerns are achieved in separate and independent levels. [30]

The coordination transparency inherent in the exogenous coordination model presents itself as a possible ramification for retrofitting legacy SCADA system for survivability in a cyber environment.

3. Retrofitting SCADA Systems through External Coordination

In this section, we present our exogenous coordination model for retrofitting legacy SCADA systems with fault tolerance in a cyber environment.

3.1. An Exogenous Coordination Model

The ARC (Actor, Role, Coordinator) coordination model is developed to model open distributed systems with non-functional requirements (or QoS requirements in general), such as survivability and attack-tolerance requirements [17].

More specifically, the ARC model has the following characteristics:

- The Actor model is used to model the concurrent computational part of a distributed cyber information system, while an independent coordination model is developed to address individual composing entities' "cooperation", or coordination. Further, the QoS requirements in general, survivability and attack-tolerance requirements in particular, are achieved through specific coordination among the asynchronous entities.
- The concept of a *role* is introduced into the coordination model. The role provides an abstraction for coordinated behaviors that may be shared by multiple actors and also provides localized coordination among its players.
- Coordination in our model is divided into inter-role and intra-role coordination to ensure clearer separation of responsibilities and reduce the complexity of individual coordination entities. This setting further ensures that both the coordination constraints and coordination activities are decentralized and distributed among the coordinators and the roles.
- The survivability and attack-tolerance requirements are mapped to coordination constraints and are transparently imposed on actors through message manipulations carried out by roles and coordinators.

The ARC model may be conceptualized as the composition of three layers, with each of the three components of the model associated with a dedicated layer, as illustrated in Figure 1. The separation of concerns is apparent in the relationships involving the layers. The actor layer is dedicated to functional behavior and is oblivious to the coordination enacted in the role and coordinator layers. The roles and coordinators constitute the coordination layer responsible for imposing coordination and QoS constraints among the actors.

The coordinator layer is oblivious to the actor layer and is dedicated to inter-role coordination. The role layer bridges the actor layer and the coordinator layer and may therefore be viewed from two perspectives. From the perspective of a coordinator, a role enables the coordination of a set of actors that share the static description of abstract behavior associated with the role without requiring the coordinator to have fine-grained knowledge of the individual actors that play the role. From the perspective of an actor, a role is an active coordinator that transparently manipulates the messages sent and received by the actor. The roles in the role layer and the coordinators in the coordinator layer are active state-based objects, enabling the coordination policies within an application to adapt over time. While actors communicate via messages that are subject to delay, the information required by roles and coordinators is communicated via atomic events that are processed atomically by all interested roles and coordinators.

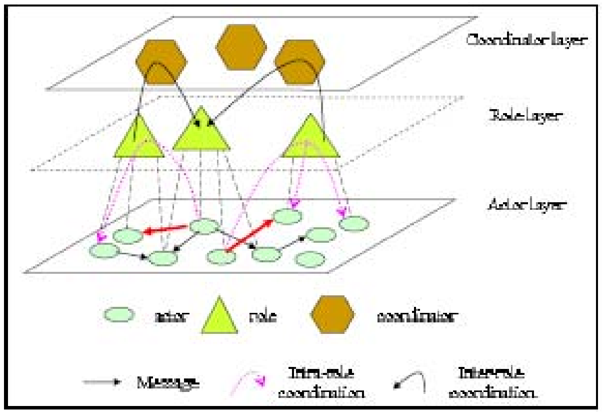

Figure 1. The ARC Model

Actors

Actors in our ARC model are based on the actor model in [1]. More specifically, actors are active objects. They have states and behaviors. The states and the current behavior of the actors decide how they process messages (operations).

Roles

Roles serve two purposes. First, roles provide static abstractions (declarative properties) for functional behaviors that must be realized by actors. Coordination based on roles is therefore relatively stable, even though the underlying actors may be of large quantity and dynamic. In addition, roles actively coordinate the actors playing the roles to satisfy fault tolerance requirements. The intra-role coordination coerced by roles complements the inter-role coordination enacted by coordinators.

The declarative criteria in the roles not only abstract the behaviors of actors, but also present a static interface to coordinators. Coordinators, therefore, do not have to directly coordinate actors, but implicit groups of actor, i.e., roles. Although in cyber environment, actors are very dynamic, they join or leave the system frequently; with role abstraction, coordinators are refrained from such dynamics.

Coordinator

Similar to the roles and actors, coordinators also have states and are active. They are able to observe events and make corresponding state adaptations. The declarative constraint policies are state-based and apply to roles only. The actors and coordinators are mutually transparent: though changes on actors or coordinators may impact on each other, such impacts are only passed through roles.

3.2. Separating Fault-Tolerance Concerns from Supervision and Acquisition Logics

To simplify our discussion, we focus on critical components and their constraints that keep a SCADA system in safe states.

From a workflow's perspective, each essential component in the physical layer has a corresponding node in the workflow. Each node has input and output ports for communication with other nodes and stores the Process Variable (PV) values of the corresponding device in the physical layer. A PV is a named piece of data associated with the current status of a process under control, such as setpoints and parameters. These values can be retrieved from the existing SCADA system. As the PV values represent the current device states in the physical layer, and the control system mathematical models represent the devices functional behaviors, the simulations of control commands or faults on the workflow realistically reflect their impacts on the real systems.

For a complex device in the physical layer, the corresponding node in the workflow can be recursively decomposed into a workflow of simpler nodes each of which performs relatively simpler activities. In other words, our workflow is a hierarchal structure [21] with subworkflows nested within composite nodes.

In addition to reflecting the essential services provided by the physical layer, the workflow also contains domain-specific security knowledge. The security-related knowledge is modeled by meta-nodes in the workflow. More specifically, depending on the roles the nodes play, they are distinguished as:

1. Computational nodes. They represent system functional entities that compose the essential service parts in the physical layer.
2. Non-functional nodes or meta-nodes. They are not the nodes that will be involved in simulating real system behaviors, but are the entities responsible for monitoring the states of computational nodes and help detect whether the system states or behaviors are in potential risks.

Currently, we have defined two types of meta-nodes. They are the Pattern Checker and the Status Checker, which carry out attack pattern recognition and node states monitoring, respectively.

Attack patterns are derived from the domain specific security knowledge. In our current study, an attack pattern is defined as a series of states of a set of computational nodes. Such states in this specific set of nodes represent an abnormal system behavior that may have been caused by a cyber attack. It is formalized by a conjunctive normal form expressing a conjunction of statuses, where a status is a specific state in a computation node. To be more specific, during a simulation the Pattern Checker is responsible for monitoring the state of the conjunctive formula, while the Status Checkers are responsible for monitoring the state of an individual computation node. If a given formula turns into "TRUE", it represents the match of the pattern.

A simulation in a workflow can be triggered by two events generated from the SCADA system, i.e. the *CommandIssue* event and the *FaultOccur* event. The Simulation Manager is responsible for monitoring and storing the occurrences of events, suspending the commands and starting/terminating the simulations accordingly. The next two subsections discuss, in detail, about the simulation process.

Statically, a workflow contains the mathematical models of the physical devices and attack patterns derived from domain knowledge. At run time, the simulations on the workflow verify the behavior of the physical system and identify potential faults through attack pattern matching.

3.3. Attack Detection

As we proposed in [28], through simulation of workflow combined with matching of attack patterns, cyber attacks in physical system can be detected.

However, some smart attackers may attack the system in a subtle way. This kind of attack is an accumulating process which consists of a series of commands. Actually, before the abnormal symptoms appear, the attacks have happened in the system for a while. So if we take these commands into consideration individually, all of them are legal. The mechanism discussed above cannot detect such an attack until the last control command, (creating the onset of abnormal symptoms) is sent to the SCADA.

For these subtle attacks,, we will take command history into account... When commands are entered, they are tracked and time slices are used to analyze the commands. With analysis of the evolution of these slices, we determine if a series of states match a pre-defined pattern. In the event of a match, warning messages are issued or some security technology, such as RSE [27], is invoked to further identify the intention of the command.

For example, we are concerned with the following pattern P in a time dimension, which is a fraction of the whole workflow. We project the pattern P onto the time dimension, then we get a series of states of pattern P in history time order, P(t1), P(t2), P(t3), as described in the following figure:

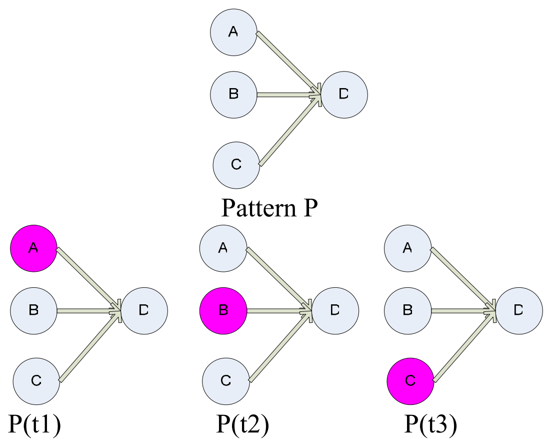

Figure 2 Pattern of Bad Behavior

The shaded nodes are those PV values that fall into a range indicating potential attacks. And we can formulate a potential attack pattern as follows:

P(t1)[A, PV1]$\in$R1,
P(t2)[B, PV2]$\in$R2,
P(t3)[C, PV3]$\in$R3,
where t1<t2<t3.

Here P(t1)[A, PV1]$\in$R1 means, in the pattern P of time t1, the value of PV parameter PV1 in node A fell in range R1. We can consider such an order as potentially dangerous.

Besides defining attack patterns, we can also define acceptable behavior patterns. Considering the above example, we define the following order-of-actions as acceptable behavior. We formulate it as follows:

P(t1)[C, PV4]$\in$R4,
P(t2)[B, PV5]$\in$R5,
P(t3)[A, PV6]$\in$R6,
where t1<t2<t3.

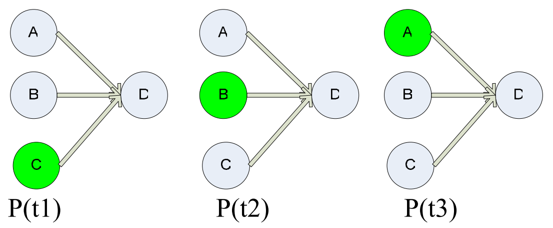

Figure 3 Pattern of good behavior

An important advantage of defining acceptable behavior patterns is that the number of acceptable behavior patterns in a system is limited, while the number of attack patterns may be unbounded.

3.3. Attack Handling

For the nodes in workflow, we can model them as actors and group the actors based on functionality. The actors with same functionality are assigned to the same group - called a role - in the ARC model. The ARC model stipulates that, at any time, an actor can only belong to one role. Therefore, in general, there are two possible roles an actor can be assigned: a role representing normal behaviors or one representing abnormal behaviors.

When the workflow simulation detects that the system is being attacked (perhaps in a subtle way), the corresponding actor of the attacked device will transit from a normal role to an abnormal role. Therefore under this circumstance, the coordinator will coordinate the roles to eliminate the actors from abnormal roles.

For example, we can consider such a scenario in Figure 4. After simulation in workflow, valve V-4 matches a bad behavior pattern for a valve, so it is transited to abnormal role for valve. When the coordinator finds the corresponding abnormal role status for the valve is changed (for example, the number of abnormal valves > 0), it will coordinate a normal role for the valve from the abnormal role for valve V-4. For instance, simple solution would be to select a valve in normal role to temporarily backup valve V-4, and reset V-4 so as to make it behave normally, and then V-4 can return to the normal role for a valve.

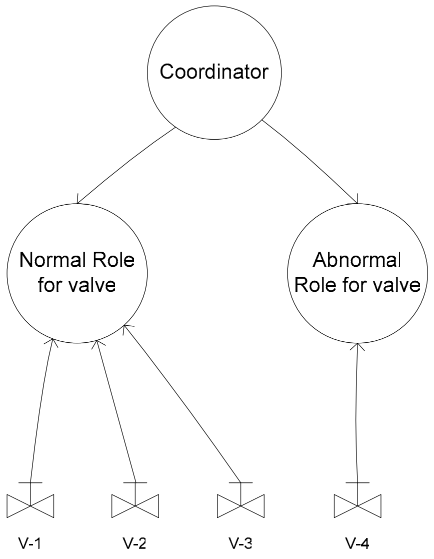

Figure 4 Example of valves in ARC model

3.4. Preventing Fault Propagation through Coordinators

Faulty states in the physical layer are monitored by SCADA systems. Through workflow analysis, extensions of the current fault can be identified, which provides valuable information for the SCADA system to forecast the potential fault propagations in the physical system and take necessary actions [28].

When fault propagation is predicted by workflow, we can apply ARC model to prevent the propagation. In the following figure, Valves V-5, V-6, V-7, V-8 belong to the role of valve, while level meter L1 and L2 belong to the role of level meter. The actions on V-5, V-6, V-7, and V-8 may change the output of level meter L1 and L2. For instance, the workflow simulation finds a fault propagation path from V-5 to Level meter 2. The coordinator can coordinate role for valve and role for level meter. Based on the coordination between roles, the role for valve will select a suitable valve to prevent the fault propagation.

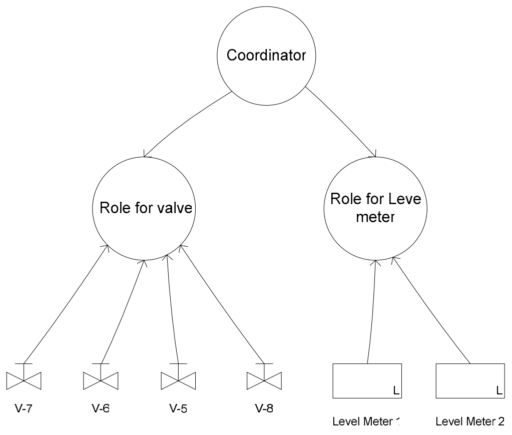

Figure 5 Example of valves and level meters in ARC model

4. Case Study

In this section, we use our approach on a simplified Water Treatment System as a case study. To simplify our discussion, we omit non-essential services of the system.

In the water treatment system, there are six valves (V1~V6) which control the fluid velocity and four pumps (P1~P4) which are used to pump raw water into the process system and distribute the purified water to consumers. In the normal condition only P3 and P4, called primary pumps, will operate. P1 and P2 are backups and will be activated only when the primary pumps are out of order.

To detect the faults on the pumps, two sensors (S1 and S2) are attached to the primary pumps to monitor their status. As soon as the status of the primary pump is abnormal, the backup pumps are activated. We also have a pressure vessel in which raw water is buffered and where elementary filtering is applied. Normally, over pressurization will not occur even when up to three pumps are activate because the filter can also release some pressure; however, when all four pumps are running simultaneously can the vessel be over pressured. This condition is a rare, abnormal situation.

A pressure release container is attached to the pressure vessel as a safety mechanism. A sensor (S3) is used to detect the pressure level in the pressure vessel. When its sensed pressure value exceeds a threshold, the valve (V6) for the pressure release container will be activated to release water from the pressure vessel. We assume that both sensors are highly reliable. Figure 6 depicts the simplified water treatment system.

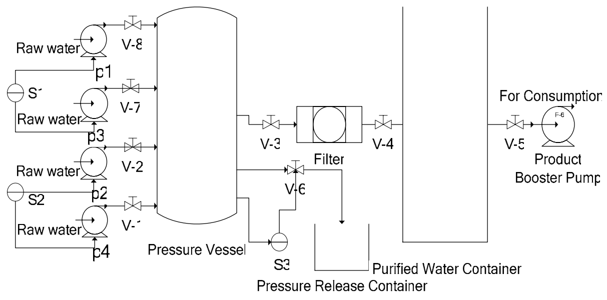

Figure 6 A Simplified Water Treatment System

We first define cyber attack patterns on this system using our domain-specific security knowledge. This knowledge allows us to specify that the condition that can over pressurize the pressure vessel over pressured is when all four pumps are activated and the pressure relief valve (V6) simultaneously fails. This low probability scenario can be seen as an attack pattern. We can describe such an attack pattern in a conjunctive normal form as follows:

$$C_1 = p_1 \wedge p_3$$

$$C_2 = p_2 \wedge p_4$$

$$C_3 = v_6$$

$$P = C_1 \wedge C_2 \wedge C_3 = (p_1 \wedge p_3) \wedge (p_2 \wedge p_4) \wedge v_6$$

Here, P is the Attack Pattern and conditions C_1, C_2 and C_3 are the output of three Status Checkers. Literals in the formula are described in the following table.

Literal	Description (Running Status of)
p1	backup pump P1 is active and normal
p2	backup pump P2 is active and normal
p3	primary pump P3 is active and normal
p4	primary pump P4 is active and normal
v6	valve V6 is abnormal

Table 1. The Description of Literals

Based on the above information, we build a workflow [28]. In this workflow, Status Checkers SC1, SC2, SC3 and Pattern Checker PC1 are meta-nodes containing the security knowledge. Other entities are computation nodes which have counterparts in physical layer. The connections between computation nodes are based on both the data flows and control flows in the physical layer, while those between meta-nodes are based on the cyber attack patterns. Meanwhile, at any time an actor will be assigned to a role. For example, in this case, there are the following roles, normal/abnormal role for a pump, normal/abnormal role for a valve, normal/abnormal role for a sensor, normal/abnormal role for the pressure vessel, and normal/abnormal role for the container. At the beginning, all the actors are assigned to normal roles, as depicted in the following figure:

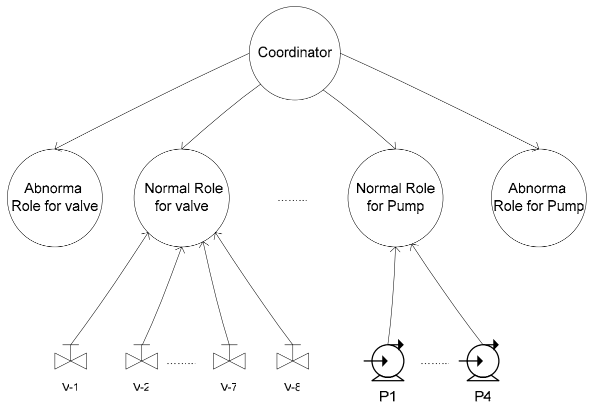

Figure 7 Initial Actor Assignment

The most primitive information comes from the Process Variables (PVs) stored in individual computation nodes. Based on this information, the Status Checkers SC1, SC2, SC3 decide whether conditions C1, C2, C3 are satisfied. By collecting the outputs of Status Checkers, the Pattern Checker PC1 can detect whether this pattern is matched. For attack resistance, workflow simulation anticipates the effect that control commands will have on essential service (i.e. those listed in Table 1). For example, assume there is a new command issued from one of the terminals in the SCADA system to "Activate Pump1". This command will be noticed by the Simulation Manger, which will trigger a simulation of the command on the workflow. Assume that in the current system state the Pump2~4 are all active, and the V6 is closed. In such a condition, the Pattern Checker will find a matching pattern such that application of the command on the real system is prevented. At this time, backup pump P1 and P2 will transit from a normal role to an abnormal role. When coordination detects the status change to an abnormal role, it will coordinate all the roles to eliminate the adverse effect.

For the preventing damage propagation, consider that the PV value in the Pressure Vessel indicates that the device is over pressurized. The Simulation Manager then starts the simulation immediately. We further assume that by simulation, the first propagated fault is located as the over pressurization of the filter. The Simulation Manager then suspends the simulation and compares this fault with the fault events stored in the fault event queue by content and time stamp. Unfortunately, by comparison, it finds that at this moment the fault has already happened in the real system. It then continues the simulation and locates the next propagated fault, which is the overflow of the Purified Water Container. It repeats the action described above. Fortunately, the fault is captured before the error occurs on this device in the actual system. The simulation is terminated and the fault propagation information will be reported to the ARC model. Then coordinator will take measurements to prevent the propagation (as we discussed in subsection 3.3).

5. Conclusions and Future Work

In this paper we have presented a coordination-based non-intrusive approach for enhancing the survivability of critical infrastructure. The advantages of our approach are twofold: (1) survivability-related knowledge and a protection scheme are built in the coordination layer which is external to the SCADA systems and therefore the disturbance to the underlying systems is greatly reduced; (2) the "separation of concerns" principle is truly reflected in our model in that fault tolerance and survivability concerns are separated from supervisory and acquisition. Such separation enables us to accommodate future requirements that may not even be anticipated today.

References

[1] United States General Accounting Office. Critical Infrastructure Protection – Challenges and Efforts to Secure Control Systems. Report to Congressional Requesters. March 2004.

[2] R. J. Ellison, D. A. Fisher, R.C. Linger, H. F. Lipson, T. Longstaff, N. R. Mead. Survivable Network Systems: An Emerging Discipline. Technical Report, CMU/SEI-97-TR-013. Nov. 1997.

[3] Understanding SCADA Security Vulnerabilities. Technical Report. Riptech, Inc. 2001.

[4] J. Pollet. Developing a Solid SCADA Security Strategy. SICON. Houston. TX. 2002.

[5] F. Haji. L. Lindsay. S. Song. Practical Security Strategy for SCADA Automation Systems and Networks. CCECE/CCGEI, Saskatoon. May 2005.

[6] C. L. Bowen. T. K. Buennemeyer. R. W. Thomas. Next Generation SCADA Security: Best Practices and Client Puzzles. In Proceedings of the IEEE Workshop on Information Assurance and Security. West Point, NY. 2005.

[7] D. Gamez. S. N. Tehrani. J. Bigham. C. Balducelli. K. Burbeck. T. Chyssler. Dependable Computing Systems: Paradigms, Performance Issues, and Applications. Wiley, Inc. 2000.

[8] InTech Inc. Intrusion Detection and Cybersecurity. Technical Report. May 2004.

[9] Y. A. Grishin, I. N. Kolosok, E. S. Korkina, L. V. Em. State Estimation of Electric Power System from New Technological systems. In Proc. Of Electric Power Engineering. 1999.

[10] M. Blanke, M. Staroswiecki, N. E. Wu. Concepts and Methods in Fault-tolerant Control. In Proc. Of the American Control Conference. Arlington, VA. 2001.

[11] B. Selic. Fault Tolerance Techniques for Distributed Systems. IBM Technical Report. 2004.

[12] Reverse Social Engineering: Countering the Insider Attack by Simulating a Human Overseer. Submitted to SCSC, 2006.

[13] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, Yang Zhao, Scientific Workflow Management and the Kepler System, Concurrency & Computation: Practice & Experience, 2005.

[14] J. Eker, J. Janneck, E. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, Y. Xiong, Taming Heterogeneity---the Ptolemy Approach, Proceedings of the IEEE, v.91, No. 2, January 2003.

[15] Wika, K.J., Safety Kernel Enforcement of Software Safety Policies, Ph.D. dissertation, Department of Computer Science, University of Virginia, Charlottesville, VA, 1995.

[16] NATO AC/310 Ad Hoc Working Group on Munition Related Safety Critical Computing Systems, "Safety Design Requirements and Guidelines for Munition Related Safety Critical Computing Systems," NATO Standardization Agreement (STANAG) 4404 (Draft), March 1990.

[17] Knight J. C. Nakano L. G. Software test techniques for system fault-tree analysis. In Proc. SAFECOMP 97, 1997, pp. 369-380

[18] Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.

[19] Paul Brutch, Tasneem Brutch, and Udo Pooch, “Electronic Quarantine: An Automated Intruder Response Tool”, Proceedings of the 1998 IEEE Information Survivability Workshop (ISW’98), October 1998.

[20] Leveson, N. G., T. J. Shimeall, J. L. Stolzy, and J. C. Thomas, “Design for Safe Software,” in Proceedings AIAA Space Sciences Meeting, Reno, Nevada, 1983.

[21] S. Bowers and B. Ludäscher. Actor-oriented design of scientific workflows. In Proc. of the Intl. Conf. on Conceptual Modeling (ER), 2005.

[22] T. M. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, R. M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics, 20(17):3045–3054, 2004.

[23] S. Majithia, M. S. Shields, I. J. Taylor, and I. Wang. Triana: A Graphical Web Service Composition and Execution Toolkit. In Proc. of the IEEE Intl. Conf. onWeb Services (ICWS). IEEE Computer Society, 2004.

[24] http://ptolemy.eecs.berkeley.edu

[25] Wayne Labs. Technology Brief (Issue 2, 2004). How secure is your control system? http://www.automationnotebook.com/2004_Issue_2/technologybrief_September2004.html

[26] NUREG-0492, Fault Tree Handbook, U.S. Nuclear Regulatory Commission, January, 1981.

[27] Nancy Leveson. A New Accident Model for Engineering Safer Systems . Safety Science, Vol. 42, No. 4, April 2004

[28] Kun Xiao, Nianen Chen, Shangping Ren, Kevin Kwiat, et al. A Workflow-based Non-intrusive Approach for Enhancing the Survivability of Critical Infrastructures in Cyber Environment. 3th International Workshop on Software Engineering for Secure Systems(SESS), Minneapolis, MN, May 2007

[29] Shangping Ren, Limin Shen, Jeffrey Tsai: Reconfigurable Coordination Model for Dynamic Autonomous Real-Time Systems. SUTC (1) 2006: 60-67

[30] Shangping Ren, Yue Yu, Nianen Chen, Kevin Marth, Pierre-Etienne Poirot, Limin Shen: Actors, Roles and Coordinators - A Coordination Model for Open Distributed and Embedded Systems. COORDINATION 2006: 247-265