Development of Dynamic Reconfigurable Integrated Management and Monitoring System for Large Scale Weapon System

Bup-Ki Min[1], Hyeon Soo Kim[1], Seunghak Kuk[2], Chumsu Kim[2] and Woonggie Han[2]

[1]Dept. of Computer Sc. & Eng. Chungnam National University
[2]Agency for Defense Development
{bkmin, hskim401}@cnu.ac.kr, {shkuk, chskim, hahaha}@add.re.kr

Abstract

This article proposes an integrated management and monitoring system based on the information model for the large weapon system on the heterogeneous distributed environments. Since the large weapon system is the large scale distributed system composed of a variety of heterogeneous subsystems, it is very difficult to manage and integrate them. In order to manage and control consistently the subsystems that operate on the different platforms, a standardized method must be employed to maintain information about them. Also, when the configuration of the subsystems is changed, information about subsystems must reflect the changed configuration. This paper uses the information model to manage a large weapon system on the heterogeneous environments in a consistent manner. In addition, the methods for dynamic reconfiguration of information model are suggested to dynamically reflect the changes in the configuration of the subsystems into the integrated management and monitoring system. With these methods, we expect that the large weapon system can be managed reliably.

Keywords: *large scale weapon system, integrated management and monitoring system, IMMS-CMS*

1. Introduction

The large scale weapon system is an automated combat system with integrated command and armament features, designed to provide optimal combat capabilities. Today, enemy threats have become increasingly faster and stealthy according to the trends of modern warfare. In response, the large scale weapon system is designed to accurately detect these threats and mount countermeasures. It also comprehensively processes data obtained from the ship's sensors and externally in combat situations in which many different things occur simultaneously. The large scale weapon system, therefore, is made up of various subsystems as shown in Figure 1, each of which are responsible for performing different functions, including detection, control, and combat (armament control, identification and tracking of targets, command and control). For example, via sensor input, the system detects enemies, controls armament, and engages in combat, each of which is implemented by a different subsystem. The overall functionality of the system is given by the combination of the functionalities of these subsystems [1].

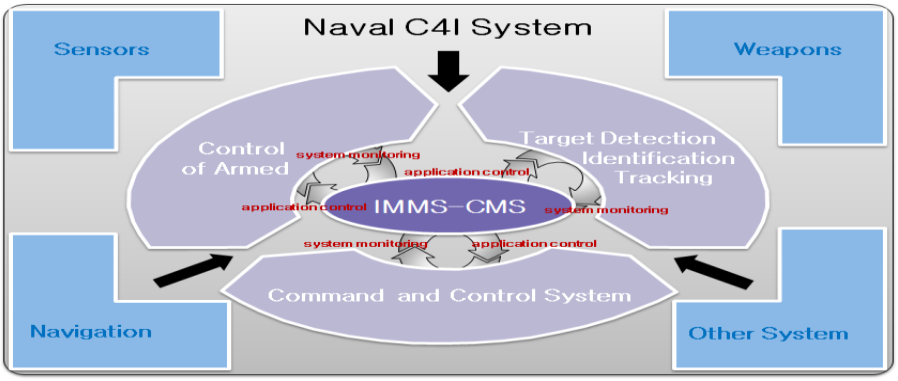

Figure 1. Naval C4I Systems

However, subsystems of the large scale weapon system may be independently developed by different vendors. In this case, management and monitoring of these subsystems have to be dependent on the particular implementation. Therefore, each subsystem had to have its own management system, different from the rest [2]. To convey events or data generated from each of the subsystems, the administrator had to remember them or they had to be written down in a document. As a result, it was difficult to respond in real-time and the accuracy of data often suffered. From these limitations arose a need for a system that can manage data from each of the subsystems and provide them in real-time. For a single environment system, it would be easy to integrate the subsystems or manage them in an integrated way. It is, however, not easy to do so for the large scale weapon system, as it is made up of heterogeneous subsystems. Therefore, a method is needed to effectively maintain and manage data generated in subsystems of the large scale weapon system (made up of heterogeneous systems in a distributed environment) under a single integrated management system.

In this paper, various services are defined using information models, and architecture is designed and implemented in a hierarchical way, so as to effectively deliver these services. Using the developed architecture, data generated in the large scale weapon system can be provided to the user in various forms, and effective control of subsystems is made possible.

2. Backgrounds

2.1. CIM

CIM (Common Information Model) is a system management standard proposed by DMTF (Distributed Management Task Force) for the distributed environment [3]. In CIM, a single model is defined for management information and service semantic system and all factors are mapped 1:1 to the semantic system of this model. A single independent model is used for detailed hardware information, service configuration, and software running on hardware. As a result, data can be reused, and consistency can be maintained in expressing data between developed products in a heterogeneous distributed environment. Another advantage of CIM is flexibility and extendibility of the model itself, for which purpose CIM provides a meta model. Using a meta model, the user can extend the model to a more detailed information model or one specialized to one specific area.

2.2. AMSM

AMSM (Application Management and System Monitoring for CMS Systems) is an integrated management standard established by OMG for the navy's naval combat system [2]. AMSM is an extended model of the common model provided by CIM. It provides information models optimized to the naval combat management system.

2.3. Related Work

Research on a system that manages all subsystems of a large scale weapon system in an integrated way has just recently got underway. Past management systems for a large scale weapon system managed subsystems on a subsystem by subsystem basis, which existed independently. They could not show all subsystems in a single integrated way [2].

In the paper [5], a management system is implemented that measures network performance and monitors network equipment. SNMP, a network management standard, is applied to the existing management system in monitoring network equipment through the SNMP manager and SNMP agent. To this end, network equipment management data are described by SNMP MIBs. In the above studies, however, only the network of the naval combat system could be managed, not the entire system. In this paper, the aim is to manage all subsystems that make up the naval combat system in an integrated way. The paper [6] proposes a monitoring method for P2P-based distributed systems. In this paper, monitoring is done in the following way: a system called P2PMonitor is implemented and a module called Alerters is added to each subsystem so that events generated are delivered to P2PMonitor. Note that in this paper, monitoring is done only on P2P applications that actually exist in the host. As the size of the system gets bigger and as the number of applications that need to be managed in a single host increase, Alerter needs to be extended. Moreover, there is not a method proposed for management or monitoring in the case that applications are added in heterogeneous environments other than the P2P system. The paper [7] proposes an integration method and a control mechanism for heterogeneous distributed systems. In the paper, policies for assembling various heterogeneous distributed systems are proposed, as well as a control delivery method called Law-Governed Interaction. In our paper, however, a system is developed that can monitor and control heterogeneous systems in an integrated way. Although the paper [7] focuses on system integration, our paper places the focus on both system integration and management. To this end, integrated management is performed that uses information models. In our paper, as information models are used whatever the application, various systems can be managed and monitored. In addition, various services in user-desired combinations can be provided.

3. Necessity of Integrated Management System for Large Scale Weapon System

Combat systems installed on ships are huge and assembled on a variety of computing platforms. Research on management techniques and system monitoring for deploying and controlling countless applications to be used in a combat system, made up of hundreds to thousands of pieces of equipment, is still in early stages. Current limitations are summarized below.

- Combat systems installed on ships are huge and assembled in diverse computing platforms.
- It is difficult to appropriately deploy countless applications on multiple computing platforms.

- The ways of managing applications are different depending on the computing platform.
- There is no consistency even for the basics of application management, such as starting/stopping.
- There are no specific ways available to control applications to ensure QoS.
- It is not easy to monitor the status of the current system.
- It is difficult to manage the overall system (requires multiple system administrators).

These limitations are placing a lot of burden on application developers, platform integrators, and even system administrators. To overcome them, research on standard integrated management and monitoring systems that spans the entire combat system is needed. To this end, in our paper an integrated management system based on information models shown in Figure 2 is developed for the large scale weapon system. The use of information models allows collection and maintenance of data from diverse subsystems that make up the large scale weapon system. Furthermore, the data are combined via services of various types. The services can provide to the user data in various forms via user interfaces.

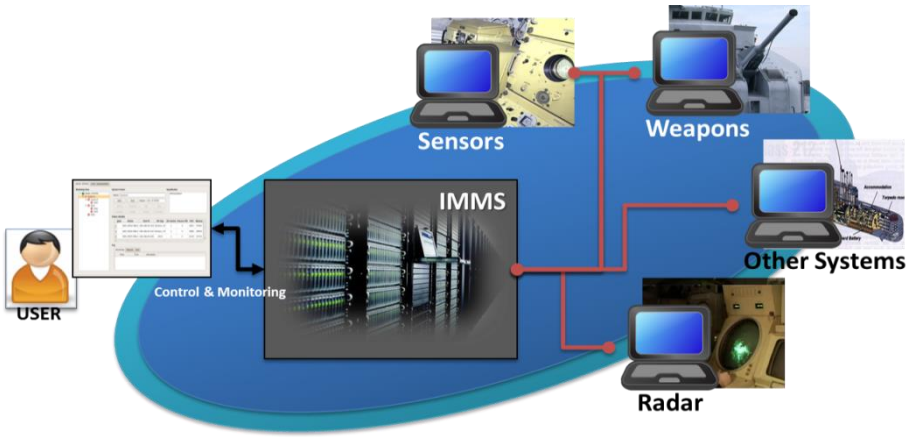

Figure 2. Integrated Management System Diagram

4. Information Model

The integrated management system for the large scale weapon system proposed in this paper performs controlling and monitoring based on information models, which extract only the independently needed data regardless of the hardware platform or the type of software. The managed data are provided to the user as various services via the user interface of the integrated management system. The information models are mapped 1:1 to the actual system hardware and running software.

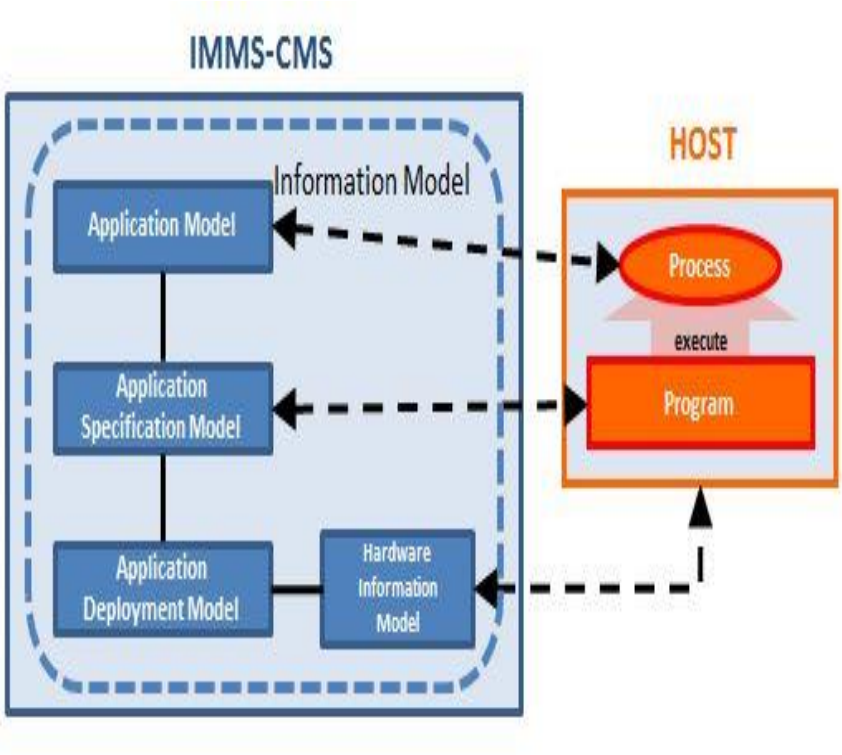

Figure 3. Mapping between the Information Model and the Actual System

Figure 3 shows the mapping between the information model and the system hardware and the running software being managed. Information models of the integrated management system are made up of the following: Application Model, which contains information of the actually running process; Application Specification Model, which contains information of applications that are not running; Hardware Information Model, which contains hardware information; and Application Deployment Model, which contains deployment information about on which hardware the application is running. Based on these pieces of information, information on the entire hardware and software running on the large scale weapon system can be maintained and managed.

5. IMMS-CMS

As the purpose of the integrated management system for the large scale weapon system implemented in this paper is the management and monitoring of subsystems that make up the large scale weapon system, it is called IMMS-CMS (Integrated Management and Monitoring System for Combat Management System). In this section, the design of the architecture for IMMS-CMS is described, as well as the design and implementation of the actual system.

5.1. IMMS-CMS System Architecture

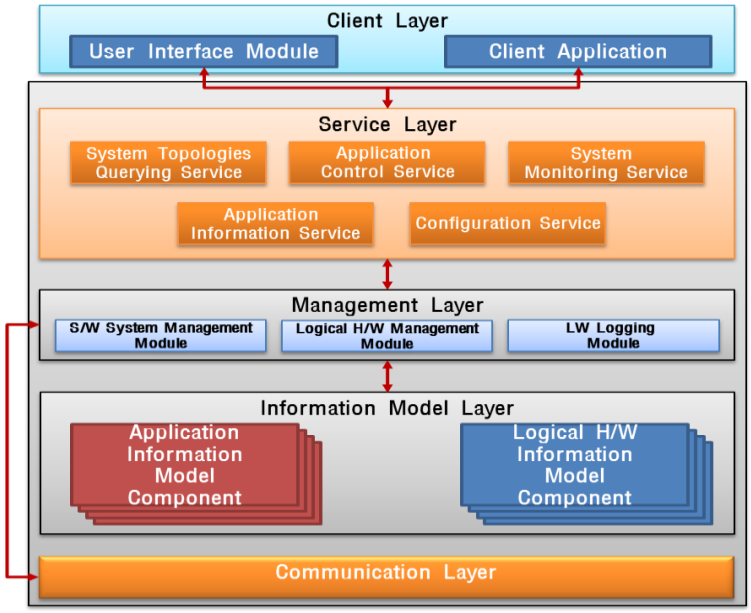

Figure 4. IMMS-CMS System Architecture

The architecture for the implementation of IMMS-CMS is proposed (see Figure 4). The architecture was designed in a way suitable for managing multiple systems that exist in diverse distributed environments under a single system. To this end IMMS-CMS is made up of different layers, each responsible for a specific kind of task. In addition, various services were made that use information models, so that services can be provided to the user in an effective way.

Client Layer

The client layer is where various services are provided to the user from the IMMS-CMS. Information models being kept in the IMMS-CMS are provided to the user so that applications in distributed environments can be controlled. In addition, monitoring information is provided to the user.

Service Layer

In the service layer, various pieces of information about diverse software and hardware are provided to the user in the form of services. The core services provided by the service layer are as follows:

- System Topology Querying Service: Provides query processing functions of the system topology, which represents the deployment information of software and hardware being managed.
- Application Control Service: This module controls applications. Specifically, it controls the life cycle of applications being managed, including running, stopping, and deploying.

- System Monitoring Service: This module monitors the system. It periodically checks the status of the software and the hardware systems being managed, and notifies when there is a change in status.
- Application Information Service: This module manages and provides information needed for operation of applications being managed, or the information of the applications themselves.
- Configuration Service: This module performs addition, deletion, or modification of the information model according to the changes to the configuration of the managed system. The changes to the configuration are as follows: addition or deletion of hardware, addition or deletion of applications, or changes of dependency information among the applications.

Currently, only the above five services are provided for the IMMS-CMS. However, more services can be created using information models as per the user's request. These services then can be provided to the user in various forms via a user interface.

Management Layer

In the management layer, software and hardware systems managed by the IMMS-CMS are managed in a substantial way. It is broadly made up of the software system management model, hardware system management module, log management module, control management module, monitoring management module and configuration management module.

- Software system management module: it manages the information model about applications
- Hardware system management module: it manages the information model for hosts
- Log information management module: it collects and manages information about the all events (for instance, monitoring, network status, error, and so on) that occurs in the course of work of IMMS-CMS

Information Model Layer

The information model layer maintains information of hardware and software systems managed by the IMMS-CMS. It manages information models of applications and hardware of each of the systems in the large scale weapon system. The system can be monitored and controlled based on this layer.

Communication Layer

The communication layer provides communication functions with the host being managed. Using various communication methods according to the type of the interface of the communication layer, control and monitoring data are exchanged between the IMMS-CMS and the host. Although at present only socket communication (TCP/IP, UDP) is supported, support for various middleware technologies is planned for the future, including DDS and COBRA.

5.2. Design and Implementation of IMMS-CMS

Design and Implementation of IMMS-CMS Information Models

In this paper, IMMS-CMS, an integrated management system for the large scale weapon system, is implemented based on the AMSM specification. The AMSM specification, however, only describes information models for the large scale weapon system, along with environments and ideas for the implementation, not specifying how to actually implement it in

any detail. Therefore, the actual design and implementation of the system is as shown in Figure 5.

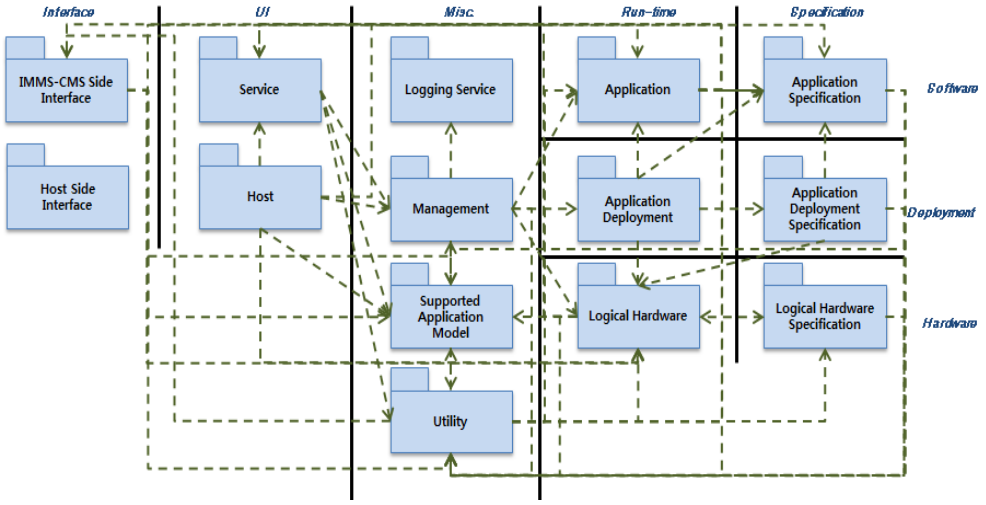

Figure 5. Information Model Packages

There are the following information model packages: Management Package, which is the starting point of IMMS-CMS services and which manages all running applications and hosts; Application Package, which manages information models of applications and software systems; Application Specification Package, which contains information for modeling applications and software systems; Application Deployment Package, which contains information of the deployment environment in which applications and software systems run; Application Deployment Specification Package, which contains information needed for modeling the deployment environment and deploying; Logging Service Package, which provides log information in the integrated management system; Logical Hardware Package, which represents hardware information models; Logical Hardware Specification Package, which contains hardware environment settings; Supported Application Model Package, which supports application and software models; Utility Package, which contains utilities needed for implementation of IMMS-CMS; Service Package, which shows application and software status information via a user interface; Host Package, which shows the host status information via a user interface; IMMS-CMS Side Interface Package, using which commands are sent from IMMS-CMS to the host, and status information is received; and Host Side Interface, using which commands are received by the host and status information is sent to IMMS-CMS.

Design and Implementation of the User Interface

The user interface provides a service environment for the user in which to monitor the status of each host connected to the IMMS-CMS, and in which to send commands to them. It is implemented using two tab controls so that the user has access to various services provided by the service layer. As the service layer gets expanded, the user interface can be expanded along with it. Currently it is made up with IMMS-CMS Service, in which applications being managed can be controlled and monitored, and Host Management, in which information of hosts connected to the IMMS-CMS can be monitored.

IMMS-CMS Service UI

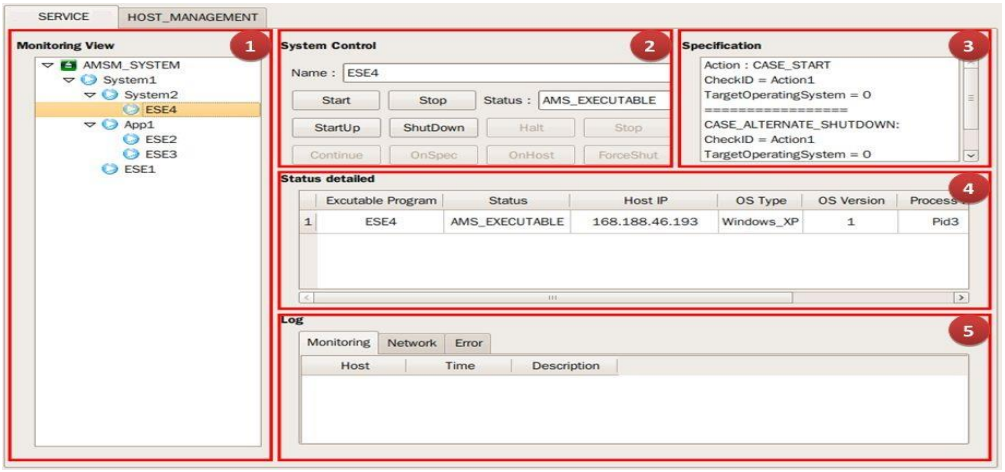

Figure 6. IMMS-CMS Service UI

1. Monitoring View: Shows all applications currently being managed by the IMMS-CMS.
2. System Control View: A single application can be selected among the ones shown in the pane 1 to control it. It corresponds to the Application Control Service in the service layer. In this window applications can be started and stopped. It also provides the current status information.
3. Specification View: Detailed information is shown, including execution conditions of applications. It corresponds to the Application Information Service in the service layer.
4. Status Detailed View: Detailed information of applications managed by the IMMS-CMS is shown, including their status information, execution host, and memory used. It corresponds to the Application Information Service in the service layer.
5. Log View: Logs of events generated during the course of execution can be viewed.

Host Management UI

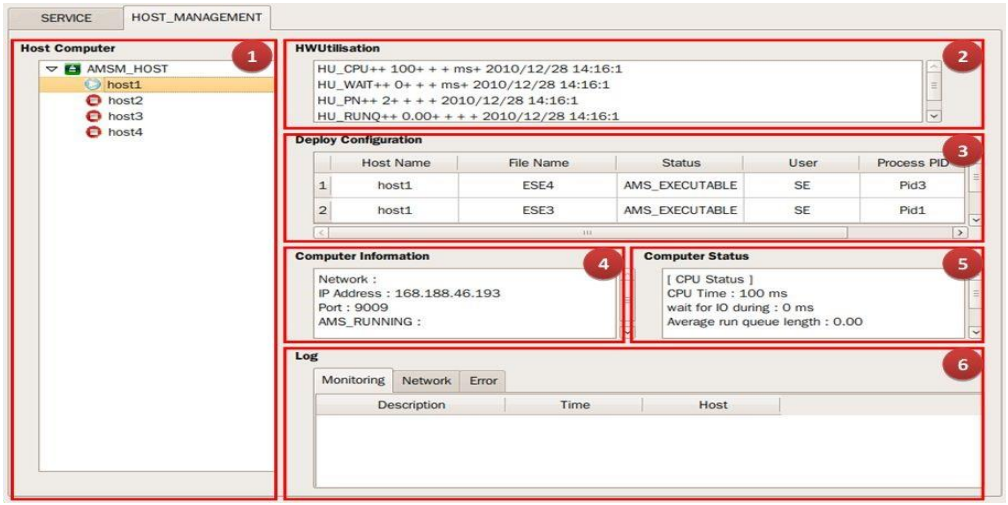

Figure 7. Host Management UI

1. Host Computer View: Shows a list of all hosts connected to the IMMS-CMS.
2. HWUtilisation View: Shows the hardware status of the host selected in the pane 1. It corresponds to the System Monitoring Service in the service layer. It provides information such as CPU load, network load, and memory usage.
3. Deploy Configuration View: Shows a list of applications running on the selected host. It corresponds to the System Topology Querying Service in the service layer.
4. Computer Information View: Provides detailed information of the selected host including IP address, port number, and execution information. It corresponds to the System Monitoring Service in the service layer.
5. Computer Status View: Provides the CPU status of the current host, network status, memory status, and hard disk status. It corresponds to the System Monitoring Service in the service layer.
6. Log View: Logs of events generated during the course of execution can be viewed.

Design and Implementation of the Host Manager

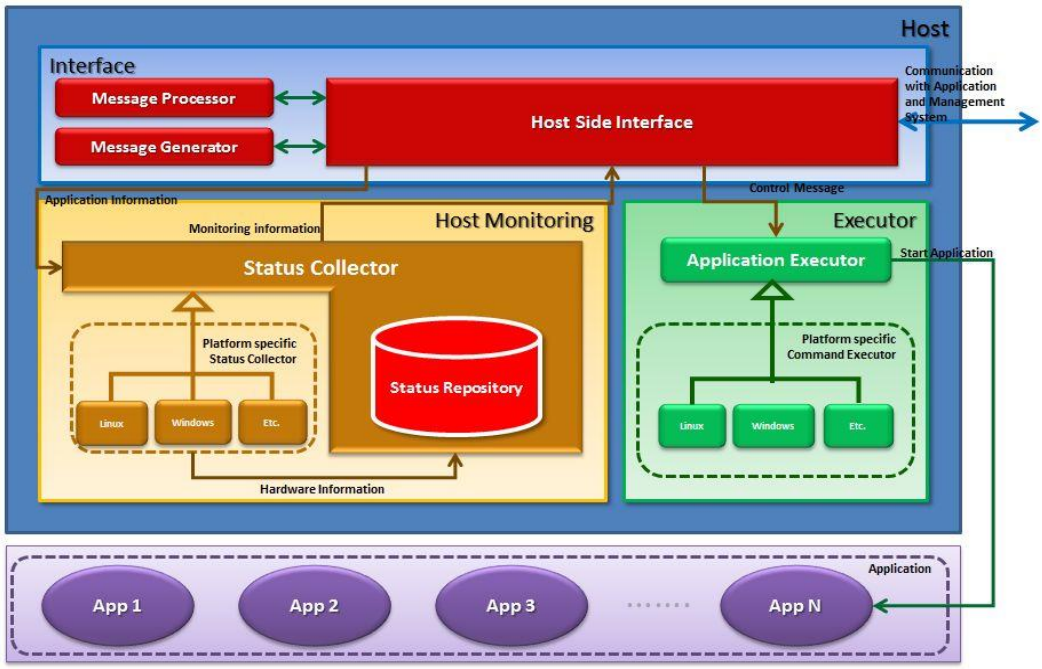

Figure 8. Host Architecture of IMMS-CMS

Figure 8 shows the detailed architecture of the host manager. The host manager is responsible for communications between applications running in the IMMS-CMS and hosts. It also collects monitoring information in the host. Therefore it is made up of the following: Host Side Interface, which is responsible for enabling communications; Status Collector, which collects host status information and status information of the application running in the host;

and Application Executor, which runs the application in the host. Because for applications the communication interface does not get initialized until executed, the execution must be done in the host. Therefore, when an application execution command has been received, the control command is directly executed by the host manager. After the execution of the application, the following are done: receive the application status from the application and send it to the IMMS-CMS; and send the control command to the application.

Dynamic Reconfiguration for Information Model

The purpose of dynamic reconfiguration for the information model is to match the information model to the managed system's changed configuration without stopping management works. This is an approach to be able to guarantee the availability and synchronization of the integrated management/monitoring system by forwarding changes that occurred on the managed system. Dynamic reconfiguration for the information model is triggered by changes to system configuration such as addition/deletion of hardware, addition/deletion of applications, or changes of information about applications. If it is needed to change the configuration, the host administrator and application administrators rewrite the host's specification files and application's specification files, respectively. Rewritten specifications are passed to the integrated management/monitoring system, in turn the configuration manager reconfigures the information model to be synchronized the configuration of the managed system.

Hardware Configuration Information

Host administrator writes the configuration information about hardware. He/she precisely determines the information about hardware before writing the configuration information as in Table 1.

Table 1. Hardware Information for Dynamic Reconfiguration

Name	Description
Name	Host name. It must be a unique identifiable name.
Operating System	Information about OS on the Host
Warning Load	Warning-level load value set on the Host for load distribution
Danger Load	Danger-level load vale set on the Host for load distribution
CPU Performance Score	CPU performance score of the Host
Memory Performance Score	Memory performance score of the Host

Application Configuration Information

Application administrator writes the application-related configuration information as in Table 2.

Table 2. Application Information for Dynamic Reconfiguration

Name	Description
Name	Application name. It must be a unique identifiable name.
Command	Describes commands the application can perform.
Redundancy Group	Application describes information about fault-tolerant group for fault tolerance.
Operating System	OS information for running the application.
OS Maximum Version	Information about the highest version of OS for running the application.
OS Minimum Version	Information about the lowest version of OS for running the application.
Startup Dependency	Information about dependency for starting up the application.
Shutdown Dependency	Information about dependency for shutting down the application.

Procedures for Dynamic Reconfiguration

Followings are the dynamic reconfiguration procedures for the addition of hardware and the addition of an application.

Algorithm – Add Hardware

In : hardware h, set of application A, set of dependency Dep

Out : void

```
1   function AddHardware(h, A, Dep){
2       A_stop = ∅ ;
3       for each a_i ∈ A do {
4           D = ∪_i (h, a_i) ;
5           for each (a_n, a_m) ∈ Dep h do {
6               if (a_i = a_n) {
7                   stopRunning (a_m) ;
8                   A_stop = A_stop ∪ a_m ;
9               }
10              if (a_i = a_m) {
11                  stopRunning (a_n) ;
12                  A_stop = A_stop ∪ a_n ;
13              }
14          }
15      }
16      IM = IM ∪ (h, A, D, Dep) ;
17      startRunning(A_stop) ;
18      startRunning(h) ;
19  }
```

Algorithm – Add Application

In : hardware h, application a, set of dependency Dep_a

Out : void

```
1  function AddApplication(h, a, Dep_a){
2      A_stop = ∅ ;
3      for each (a_i, a_j) ∈ Dep do {
4          if (a = a_i){
5              stopRunning(a_j) ;
6              A_stop = A_stop ∪ a_j ;
7          }
8          if (a = a_j){
9              stopRunning(a_i) ;
10             A_stop = A_stop ∪ a_i ;
11         }
12     }
13     D_a = (h, a) ;
14     IM = IM ∪ (∅, {a}, D_a, Dep_a) ;
15     Notify(A_stop) ;
16     startRunning(A_stop) ;
17 }
```

- $IM = (H,\ A,\ D,\ Dep)$: information model
- H: set of the information models for hardware
- A: set of the information models for applications
- $D \subseteq H \times A$: set of the deployment information models. If an application ($a \in A$) is deployed on some hardware ($h \in H$), this fact is represented by deployment information model, $(h, a) \in D$.
- $Dep \subseteq A \times A$: set of the dependency information models. If, when starting or finishing an application ($a_i \in A$), it is influenced from the status of another application ($a_j \in A$), the dependency relationship between them is represented as $(a_i, a_j) \in Dep$.
- stopRunning(): function that stops execution of hardware or application in the managed system.
- startRunning(): function that starts execution of hardware or application in the managed system.
- Notify(): function that informs hardware of the change of dependency. The hardware contains applications in which some dependency relationships are changed.

Execution Scenario for IMMS-CMS

Running and Monitoring Applications

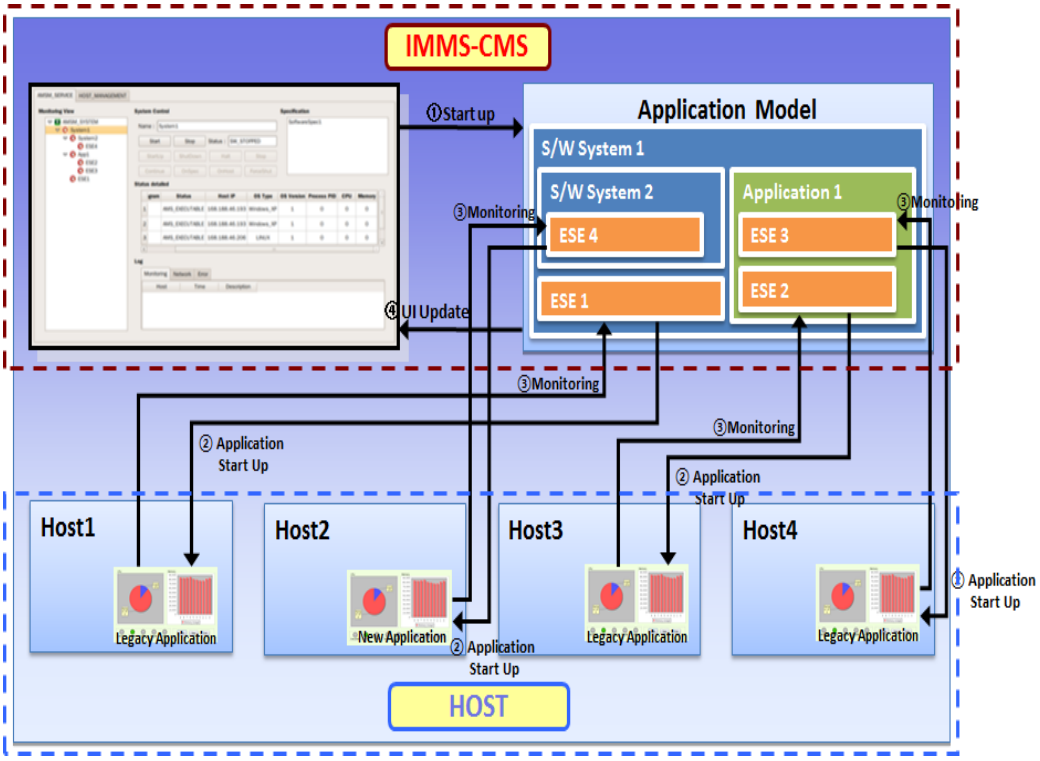

Figure 9. Application Execution Method

The IMMS-CMS runs applications as shown in Figure 9.

1. The user selects a software system via the user interface and presses the execute button. The execution control message will be sent to the application model.
2. In the application model, applications of software systems to be executed are found and an execution command is sent to each of the hosts.
3. Each of the hosts run the application and collects status information and sends it to the application model. Each of the application models is updated based on the collected information and changes are notified via the user interface.
4. In the user interface, information to be shown to the user is changed using information of updated application models.

Monitoring System Error Status

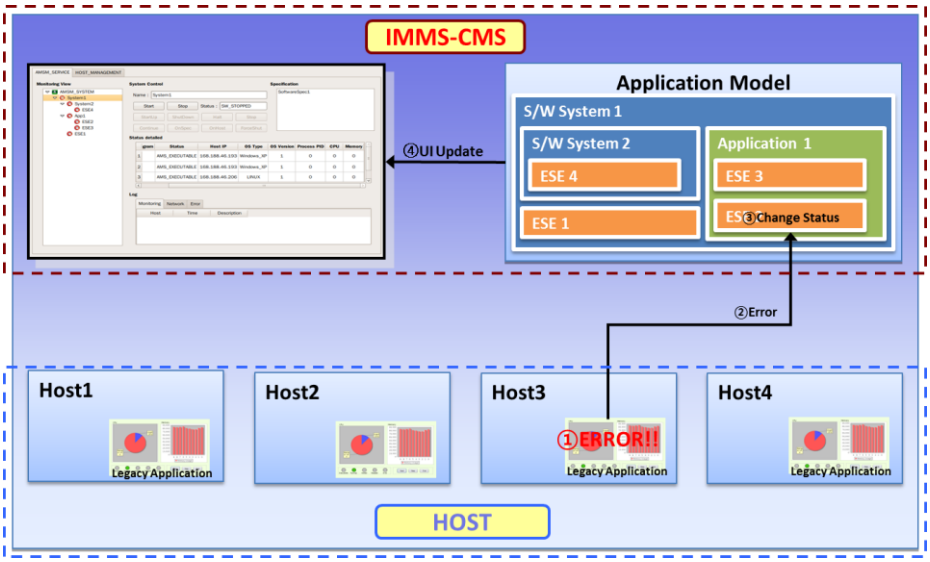

Figure 10. Operation in Case of Errors

The integrated management system works as in Figure 10 in case errors are detected during monitoring.

1. An application in operation makes an error.
2. The corresponding host gives information about the erroneous application and its 1:1 mapping application model.
3. The application model changes the status information and informs the user interface of the change.
4. The user interface uses the changed information to update the application's model. Then, the user interface indicates the application is in error so that the user can check out the information.

6. Conclusions

As the size of the large scale weapon system gets bigger and as greater number of subsystems are installed, it is increasingly becoming more important to manage each of the subsystems in a an integrated way. Such a large scale weapon system is not made up of a single system but rather of various heterogeneous subsystems. Therefore, a standardized integrated management system is needed to effectively manage subsystems of the large scale weapon system. In this paper, an integrated management system for the large scale weapon system is implemented, called IMMS-CMS, which uses information models for hardware and software being managed in the large scale weapon system. As the information models are managed in a way independent of the actual system, they are not affected by heterogeneous environments and an integrated management can be done for all subsystems in the large scale weapon system.

Integrated management system architecture was designed and implemented which can effectively use information models. Using the architecture, services can be formed in various combinations and provided to the user via the user interface.

In addition, in order to solve the synchronization problems of the integrated management and monitoring systems due to the change to the configuration of a large weapon system, a way that reconfigures the information model dynamically is presented.

The aim of this paper was an integrated management of all subsystems of the large scale weapon system. As the focus of the implementation was on the integrated management of different subsystems, things such as overload or errors that may occur in any of the subsystems were not taken into consideration. However, load balancing and fault tolerance are planned to be implemented in the future. This would prevent any potential overload in the large scale weapon system and a subsystem would be able to be replaced with another even if an error occurred.

Acknowledgements

This research was supported by Agency for Defense Development. (Contract No. UD090017KD).

References

[1] C. Eryigit, "A. S Uyar, Integrating Agents into Data-Centric Naval Combat Management Systems", 23rd International Symposium on Computer and Information Sciences, **(2008)**, pp. 1-4.
[2] OMG, Application Management and System Monitoring for CMS Systems, http://www.omg.org/spec/AMSM/1.1(**2010**).
[3] DMFT, Common Information Model, http://www.dmtf.org/standards/cim, **(2011)**.
[4] OMG, MDA Guide Version 1.0.1, http://www.omg.org/mda, **(2000)**.
[5] E. H. Kim, M. J. Choi, W. K James, M. S. Cho, S. E Chun and Y. O. Lee, "Design and Implementation of an SNMP-based Integrated Management System for Efficient Management of Naval Combat System", Journal of the Korea Institute of Military Science and Technology, vol. 11, no. 2, **(2008)**, pp. 32-42.
[6] S. Abiteboul, B. Marinoiu and P. Bourhis, "Distributed Monitoring of Peer-to-Peer Systems", International Conference on Data Engineering, **(2008)**.
[7] N. H. Minsky and V. Ungureanu, "Law-Governed Interaction: A Coordination and Control Mechanism for Heterogeneous Distributed System", ACM Transactions on Software Engineering and Methodology, vol. 9, **(2000)**, pp. 273-305.
[8] M. Wegdam, "Dynamic Reconfiguration and Load Distribution in Component Middleware", PhD. Thesis, University of Twente, the Netherlands, **(2003)**.

Authors

Bup-Ki Minis a PhD student at Chungnam National University, Korea. He is a member of the Software Engineering and Application laboratory. His research interests include Software Engineering, Software Testing, Software Architecture and Smart Phone. He received the Bachelor (2009) degrees from Kongju National University, Korea and the Master (2012) degrees from Chungnam National University, Korea. He is a member of KIISE, KSII and KIMST.

Hyeon Soo Kim is a professor at Chungnam National University (CNU), Korea. He works for Department of Computer Science and Engineering at CNU. His current research areas include Software Engineering (Software Testing, Software Architecture, Software Maintenance, Software Reengineering), Distributed Computing (J2EE/EJB, .NET, Web service, Service-Oriented Architecture). He received the BS degree (1988) from Seoul National University (SNU), Korea, and the MS (1991) and the PhD degree (1995) from Korea Advanced Institute of Science and Technology (KAIST), Korea. He is a member of KIISE and KIPS.

Seunghak Kuk is a researcher at Agency for Defense Development(ADD), Korea. His research interests include Software Engineering, Software Testing, Software Architecture, Service-Oriented Architecture, Web Service and Smart Phone. He received the Bachelor (2009), the Master (2012) and the PhD degrees from Chungnam National University, Korea. He is a member of KIISE, KIPS.

Chumsu Kim is a senior researcher at Agency for Defense Development (ADD), Korea. He works for Naval Combat Systems PEO at ADD. His research areas include system/software engineering and combat system development (system architecture, analysis, design, and test). He received the BS degree (1996) from Inha university, Korea, and the MS (1996) from Gwangju Institute of Science Technology(GIST), Korea. He is a member of KIISE.

Woonggie Han is a senior researcher at Agency for Defense Development (ADD), Korea. He works for Naval Combat Systems PEO at ADD. His research areas include system/software engineering and combat system development (weapon control, M&S, and training). He received the BS degree (1996) and the MS (1996) from SungKyunKwan University.