11

A STRATEGY FOR AN MLS WORKFLOW MANAGEMENT SYSTEM

Myong H. Kang, Judith N. Froscher, Brian J. Eppinger and Ira S. Moskowitz

Abstract Current DoD information systems need to support many different missions through cooperation with different organizations and allies. In todays fast paced and dynamic environment, it is almost impossible to design and implement a different information system for each mission. Therefore, DoD needs MLS workflow management systems (WFMS) to enable globally distributed users and existing applications to cooperate across classification domains to achieve mission critical goals . An MLS WFMS that allows users to program multilevel mission logic, securely coordinate widely distributed tasks, and monitor the progress of the workflow across classification domains is required. In this paper, we present requirements for MLS workflow and a strategy for implementing it, especially the method for decomposing an MLS workflow into multiple single-level workflows.

Keywords: multilevel security, workflow management systems

1. INTRODUCTION

The streamlining of todays business processes has brought about significant increases in productivity and the ability for US companies to compete in the global market place. These re-engineering activities have as their basis an even greater reliance on information technology and automation. As a result, software developers have been challenged to streamline the software development process and to produce software that can be reused, that separates concerns, that supports autonomy and heterogeneity in a distributed computing environment, that allows extensibility, etc. The information technology (IT) community has developed distributed object computing standards, like CORBA and DCOM , that provide a basic level of interoperability among distributed applications. The next step is to build application specific software

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35508-5_22

V. Atluri et al. (eds.), *Research Advances in Database and Information Systems Security*

architectures that encode business logic for coordinating widely distributed manual and automated tasks in achieving enterprise level objectives. To assist business process modeling, vendors have developed several automated tools that help manage dependencies among activities and users. A WFMS makes these tools available to users and allows them to monitor the business processes. This technology manages activities within a distributed computing environment comprising loosely coupled, heterogeneous, autonomous, new (and legacy) components which may include transactional systems (i.e., DBMS). It provides a capability for defining the (1) business logic, (2) tasks that make up business processes, and (3) control flow and data dependencies among those tasks.

The potential benefits of this technology are enormous because of its broad reach to manage business process productivity. Industry is beginning to turn to workflow technology in order to minimize its manpower needs, optimize IT investment, achieve good performance, use legacy systems, gain flexibility for supporting evolving business goals, as well as to capitalize on advanced technology. However, commercial WFMS do not support distributed mission critical applications. They do not ensure mission critical properties, such as recoverability nor do they enforce access control policies, and certainly not multilevel secure (MLS) access control policies. Even though there is a great need for this technology in DoD, DoD cannot rely on commercial WFMS to protect national security information and perform mission critical business processes.

To address those problems, the Naval Research Laboratory (NRL) has embarked upon a research project to build an MLS WFMS . The goal of the project is to develop tools and security critical components that allow enterprises to harvest emerging commercial off-the-shelf (COTS) technology and still rely on legacy resources with reduced risk. In short, MLS WFMS should support (1) secure interoperability among tasks that reside in different classification domains, and (2) maximum use of commercial software/hardware.

The need for a WFMS that can manage MLS workflows is immediate and imposes an implementation strategy that allows operational users to exploit advances in COTS technology and also to satisfy their mission critical requirements. The multiple single-level architecture, described in [5, 6, 13], provides the foundation for enforcing information flow requirements. A WFMS comprises several tools and runtime enactment services. This paper examines what properties these components must satisfy in a multiple single-level distributed computing environment. The MLS workflow design tool is of particular interest because the commercial tool must be significantly changed to represent classification domains. While this does not fit the paradigm of using unmodified COTS products with high assurance security devices, finding a research

team [8, 9, 10] that is developing a WFMS that supports mission critical workflows makes it possible to develop an MLS WFMS.

In this paper, we present requirements for MLS workflows, tools for supporting MLS workflows, and a strategy for implementing them. We also describe the decomposition of an MLS workflow into multiple single-level workflows that will work together to accomplish the mission.

2. TOOLS TO SUPPORT MLS WORKFLOW

A WFMS consists of, in general, three components (1) workflow design (build-time) tool, (2) workflow enactment (runtime) service, and (3) monitoring tool.

A workflow design tool is a distributed programming tool with a graphical user interface. Users can express data dependencies and control flow among tasks using this tool. Once a user specifies the mission logic (i.e., distributed programming logic), code for workflow enactment can be generated. A workflow enactment service is responsible for task scheduling, enforcing dependencies among tasks, passing data from one task to another, and error recovery based on the generated code. The workflow monitoring tool, in general, tracks and monitors the progress of execution.

DoD needs all the tools that single-level WFMS provides. However, DoD requires extra capabilities in those tools to support MLS workflow. We will examine the extra requirements for each tool.

2.1 DESIGN TOOL FOR MLS WORKFLOW

A workflow design tool is a distributed programming tool with a graphical user interface that provides the global picture of the whole mission. MLS workflow designers should be able to specify (1) tasks in different classification domains and (2) data and control dependencies (flow) among them. Based on this workflow design, a specification for workflow runtime can be generated. Final runtime code that will be securely executed on an MLS distributed architecture is generated, based on this specification. The runtime specification and code generation processes, in general, depend on the underlying MLS distributed architecture.

In MLS applications, each subtask may be located in a different classification domain. The design tools for single-level workflow do not provide a capability to specify classification domains or compartments. In other words, the whole drawing area of the workflow design tool belongs to one classification domain. It also does not generate runtime code that can be executed on the underlying MLS distributed architecture. What is needed is a design

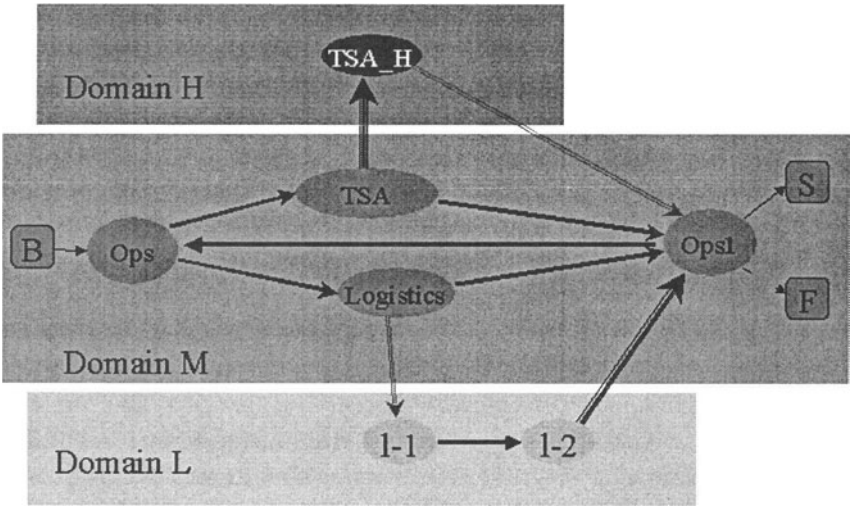

Figure 11.1 An example of an MLS workflow design.

tool for MLS workflow that(1) allows MLS workflow designers to divide a design area into multiple domains,(2) allows MLS workflow designers to specify information flow and dependencies among tasks that are in the same or different domains, and (3) allows MLS workflow designers to specify dominance relationships among domains (e.g., Top Secret > Secret > Unclassified).

For example, a workflow designer should be able to specify an MLS workflow as in Figure 11.1 where ovals represent tasks and arrows represent control and data flow. In the figure, B (begin), S (success), and F (failure) represent special tasks.

Even though this tool allows workflow designers to specify information and control flow among tasks in different domains, the operational environment of the tool will be system-high (i.e., the workflow design tool neither accesses sensitive data in multiple domains nor passes it around). Hence, although this tool has to be trusted in the sense that it does what it is supposed to do, it can be run on a single-level system.

MLS workflow has another functional requirement. When an MLS workflow is designed, it must often interact with tasks in other domains about which the designer is not allowed to know the details. For example, when a secret workflow designer designs a workflow, the secret workflow may need to send a request to a top secret task. The secret designer is not allowed to know how the top secret task gets the answer, but he knows how to send a request and how to receive an answer. Hence, the top-secret task, in this case, is a task foreign to the secret designer (i.e., the top secret task does not belong to his work-

flow). In fact, the H workflow in figure 11.1 may be designed in a completely different organization from the M workflow and the L workflow. A workflow design tool for MLS workflow should be equipped with the capability to model interactions with a task for which only its interface specification is known. We believe this capability to model foreign tasks has broader application, even for single-level workflows (e.g., two cooperative workflows run by two different organizations).

2.2 ENACTMENT SERVICE FOR MLS WORKFLOW

An MLS workflow enactment service is responsible for executing a workflow in a correct and secure manner. Hence, it depends on services in the underlying MLS distributed architecture to coordinate information and control flow among tasks in different classification domains. What we need is to make secure use of single-level COTS workflow enactment services with or without modifications. As we will explain in section 3, we plan to use multiple single-level workflow enactment services to execute an MLS workflow. Since there will be no direct communication among workflow enactment services at different classification domains, there are no special MLS requirements for a workflow enactment service itself. On the other hand, the underlying MLS distributed architecture and its security devices must provide the necessary assurance for multilevel security.

2.3 MONITORING TOOL FOR MLS WORKFLOW

When MLS workflow is executed, there are many automatic and human computer tasks that are executed in different classification domains. Workflow managers in different classification domains (there may be a workflow manager per classification domain) may have knowledge about tasks in their classification domain and other domains they are authorized to access. In other words, users of MLS workflow in different classification domains may have different views of the workflow they are running. Hence, an MLS WFMS should provide the ability to monitor activities in all domains the workflow manager is authorized to access. Monitoring may include when, where, and who performs the tasks in the case of human tasks. Because the expected, legal behavior of a workflow is specified, the workflow monitor can be designed to detect security critical events as well as unexpected behavior. Additionally, responses for security exceptions can be specified as part of the workflow design.

3. A STRATEGY FOR MLS WORKFLOW

An MLS WFMS should support functionality equivalent to a single-level WFMS from the perspective of MLS users who design and use multilevel workflows. Tasks that may be single-level individually but located in different classification domains, have to cooperate to achieve a higher level MLS mission.

To provide MLS services in a distributed and heterogeneous computing environment, a few information flow requirements must be enforced. Those are (1) high users must have access to low data and low resources, (2) high processes must have access to low data, and (3) high data must not leak to low systems or users.

An MLS WFMS should obey this MLS policy. Atluri et. al. investigated MLS workflow in general [2]. There are two basic ways to enforce the MLS policy in MLS workflow systems. (1) Build high-assurance MLS WFMS that will run on an MLS platform. (2) Build an MLS workflow by integrating multiple single-level workflows with an MLS distributed architecture.

The development of high-assurance software, necessary to provide separation between unclassified and TS/SCI information, such as MLS workflow systems, has proven to be both technically challenging and expensive. Todays fast paced advances in technology and the need to use COTS products make the traditional MLS approach untenable. Therefore, we have chosen the second approach for building MLS WFMS. It is more in line with the modern distributed computing paradigm than the first approach in terms of supporting autonomy and heterogeneity.

To implement an MLS WFMS using the architectural method, the following technical approach has been established:

- Choose an MLS distributed architecture where multiple single-level workflows can be executed.
- Choose a strategy for dividing an MLS workflow into multiple single-level workflows.
- Choose a single-level WFMS to execute single-level workflow in each classification domain.
- Implement the necessary tools for supporting MLS workflow.
- Extend the workflow interoperability model to accommodate the communication among workflows at different classification domains.
- Extend the single-level workflow enactment service to accommodate communication among tasks in different classification domains.

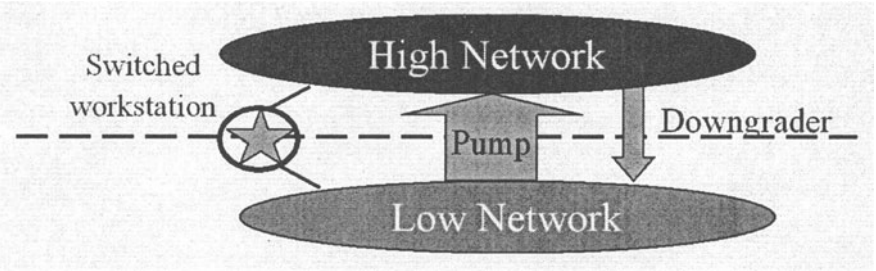

Figure 11.2 An MLS distributed architecture.

3.1 MLS DISTRIBUTED ARCHITECTURE

Composing an MLS workflow from multiple single-level workflows is the only practical way to construct a high-assurance MLS WFMS today. In this approach, the multilevel security of our MLS workflow does not depend on single-level WFMS but rather on the underlying MLS distributed architecture. Thomas and Sandhu have proposed task-based authorization for single-level workflows [14]. The MLS distributed architecture will(1) host multiple single-level workflows to be executed and (2) provide conduits for passing information among tasks in different classification domains.

Our MLS distributed architecture is based on a security engineering philosophy: a few trusted devices in conjunction with information release and receive policy servers enforce the information flow policy of the classification domains, and single-level systems and single-level engineering solutions provide other functionality. A generic MLS distributed architecture is shown in Figure 11.2.

In this architecture, switched workstations (e.g., Starlight [1]) enable a user to access resources in multiple classification domains and create information in domains that the user is authorized to access. One-way devices (e.g., a flow controller such as A Network Pump [4]) together with information release and receive policy servers provide a secure way to pass information from one classification domain to another. An information release policy server resides in a classification domain where the information is released, and an information receive policy server provides a secure way to pass information from one classification domain to another as shown in Figure 11.3.

In general, when COTS software passes data to a flow controller a wrapper translates the protocol of COTS software to that of the flow controller because COTS software and flow controllers communicate with other software through their own protocols. Hence, a wrapper can be considered a protocol translator. The detailed description of the architecture and the MLS services are in [6].

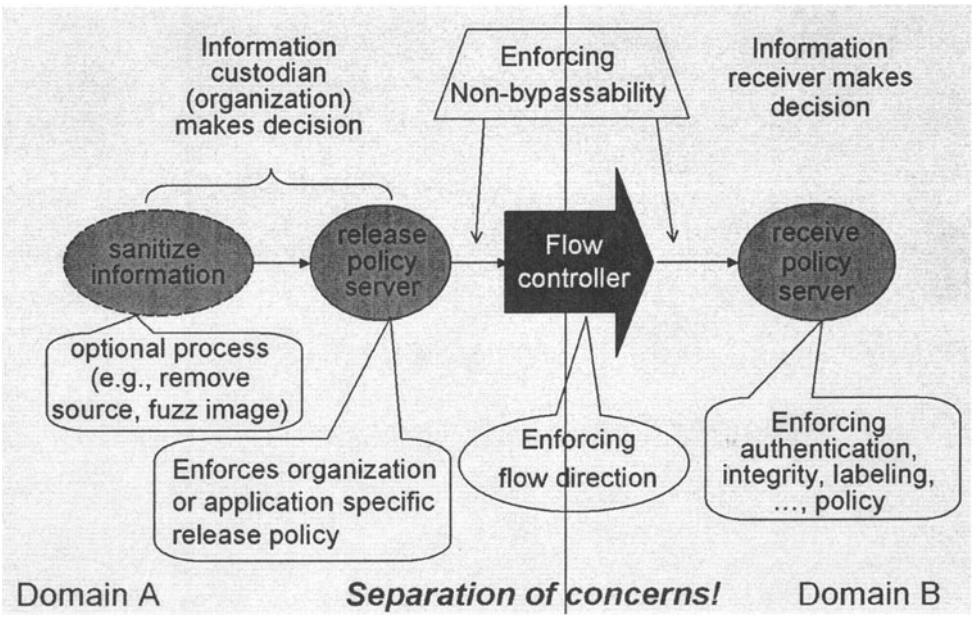

Figure 11.3 Information release and receive policies.

3.2 WORKFLOW INTEROPERABILITY

As we mentioned earlier, our strategy for implementing an MLS workflow is through combining single-level workflows on an MLS distributed architecture. Workflows in different domains may be heterogeneous and autonomous. Hence workflow interoperability is an important requirement for the approach that we have taken to implement an MLS workflow. Two important aspects of workflow interoperability are (1) interoperability protocol among independent WMFS and (2) the ability to model interoperability in a workflow process definition tool (i.e., workflow designer).

A standard body such as Object Management Group (OMG) can handle the first aspect (e.g., jFlow [12]). However, the second aspect should be handled by each WFMS.

OMGs jFlow introduces two models of interoperability. They are nested sub-process and chained processes [12]. These two models provide powerful mechanisms for interoperability. However, we would like to extend these models to support an additional interoperability model, cooperative processes. Consider two independent autonomous workflows that need to cooperate. Let us assume that there are two cooperating organizations. Organization A is in charge of workflow A and Organization B is in charge of workflow B. Tasks

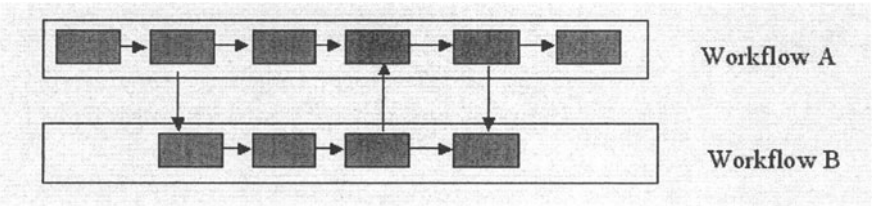

Figure 11.4 Cooperative processes.

in workflow A and workflow B can communicate and synchronize with each other as shown in Figure 11.4. In this example, two workflows may have independent starting and ending points.

There is another situation that we want to support in the context of cooperative processes. In general, the designer of workflow A does not need to know the structure of workflow B and vice versa. In conjunction with MLS principles, the designer of a workflow may not be allowed to know the detailed workflow structure of a higher level workflow. For example, in Figure 11.1, the designer of the workflow, whose classification domain is M, may not be allowed to know the workflow structure that contains the TSA_H task.

However, there is a minimal set of information that is required for communication and synchronization among tasks in cooperating autonomous workflows. This includes (1) where and how to send/receive requests (i.e., the location and invocation method of tasks) and (2) how and where to receive replies (i.e., expected output and the return location).

Therefore, the above specification has to appear in the workflow design so that the proper runtime code can be generated. Hence, we need a primitive that represents this situation in the design tool. We have implemented an MLS workflow design tool based on an MLS workflow model. Due to the limited space, we present those in a separeate paper [7].

4. MLS DEPENDENCIES AND MLS WORKFLOW DECOMPOSITION

As we mentioned briefly in section 2, an MLS workflow design tool allows MLS workflow designers to (1) divide a design area into multiple domains, (2) assign tasks in different domains, and(3) specify data and control flow among them.

Once the design of an MLS workflow is completed, the MLS workflow has to be divided into multiple single-level workflows to be executed on the underlying MLS distributed architecture that was described in section 3.

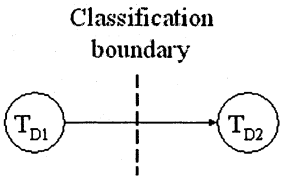

Figure 11.5 MLS transition across a classification boundary.

4.1 MLS DEPENDENCIES

An MLS workflow design tool should allow the same kind of intertask dependencies as a single-level workflow design tool (e.g., guards, input and output classes). However, some dependencies in an MLS workflow may be specified across classification boundaries (i.e., MLS dependencies). In other words, workflow state information and some values may have to move across classification boundaries during workflow execution. Hence, it is important to understand what the vulnerability is, whether it is easily exploitable and how to reduce it.

In our MLS WFMS, all information that has to move across classification domains must go though information release and receive policy servers and a high-assurance flow controller (e.g., Pump or downgrader [3, 11]). We divide a transition between two tasks in different classification domains into a series of transitions. For example, if there is a transition from a task in domain 1 (TD1) to a task in domain 2 (TD2) as in Figure 11.5, then

this transition will be divided into transitions from TD1 to PD1, PD1 to PD2, and PD2 to TD2 as shown in Figure 11.6. Note that there is the flow controller between PD1 and PD2 where PD1 and PD2 are proxies that combine flow controller wrappers and policy servers (i.e., PD1 combines a flow controller wrapper and information release policy server, and PD2 combines a flow controller wrapper and information receive policy server).

Note that domain policies may not allow combining flow controller wrappers and information policy servers. In that case, we have to break down transitions further. Also note that since TD1 and TD2 belongs to two different workflows, the PDi serve as synchronization points.

4.2 DECOMPOSITION OF AN MLS WORKFLOW

Using the method that we described above, an MLS workflow will be separated into multiple single-level workflows. The single-level workflows neither

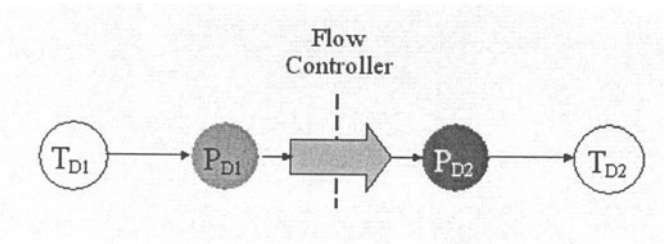

Figure 11.6 Indirect transition through a flow controller and policy servers.

communicate directly nor recognize single-level workflows from other classification domains. The number of single-level workflows that will be generated is equal to the number of classification domains in the MLS workflow (except the domains that contain only foreign tasks). When the breakdown is performed, direct transition (see Figure 11.5) becomes indirect transition as in Figure 11.6, and security depends on the underlying MLS architecture. Before we show an example of the division from an MLS workflow to multiple single-level workflows, we will formalize the MLS workflow model and the decomposition of an MLS workflow.

4.3 AN EXAMPLE DECOMPOSITION

Let us consider the simple MLS workflow shown in Figure 11.1 with classification domains L < M < H. Since this particular workflow is designed at domain M (special nodes, B, S, and F signify where the particular workflow was designed), all tasks in domain M and L, which is dominated by domain M, may be native tasks. Since the designer of the workflow in domain M may not know the detailed structure of the workflow in domain H, which dominates M, he can declare the TSA_H a foreign task. The specification of foreign task expresses interfaces (i.e., invocation methods and outputs) and where to send requests. The transitions, TSA to TSA_H, TSA_H to Ops1, Logistics to l-1, and l-2 to Ops1, and input and output classes that are associated with each transition define when and what kind of data will be passed to other workflows at different classification domains. After decomposing the MLS workflow, the runtime code generator generates the two workflows as shown in Figures 11.7 and 11.8. The shaded proxies in Figures 11.7 and 11.8 represent the combination of policy server, flow controller wrapper and synchronization nodes that represent external transitions.

Note that workflows in domains M and L are two independent workflows after the decomposition. The concept of cooperative processes and the synchronization node that we discussed in section 3 are useful for interoperability

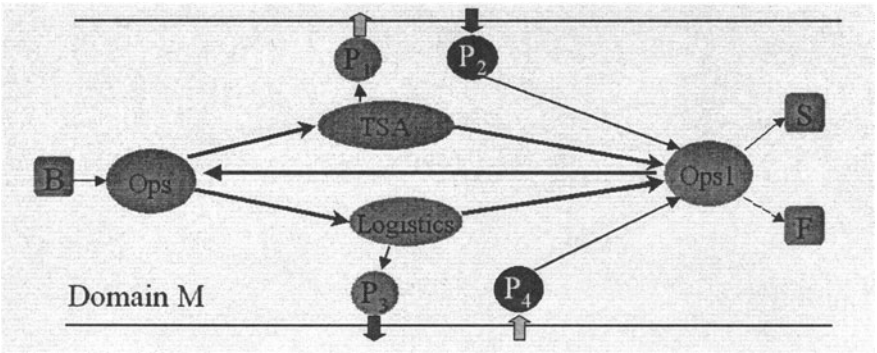

Figure 11.7 An outcome of the M workflow after decomposition.

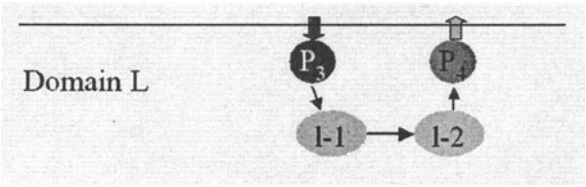

Figure 11.8 An outcome of the L workflow after decomposition.

between these two independent workflows. Also note that the decomposition of the original MLS workflow does not generate the workflow in domain H because there is only a foreign task in domain H.

5. SUMMARY

Todays military is required to respond to constantly changing threats and cooperate with allies and different organizations. This dynamic environment and the militarys dependence on IT systems require an MLS WFMS that allows

- Rapid specification and construction of mission specific IT systems,
- Maximum use of existing or commercial software/hardware, and
- Secure sharing and exchanging information among organizations in different classification domains.

In this paper, we presented requirements for an MLS workflow and tools needed to support an MLS WFMS. An MLS workflow designer should be able to specify different classification domains and tasks in those domains. He also should be able to specify control and data flows among tasks in different

classification domains. To accommodate this need, we introduced an MLS workflow model and an MLS workflow design tool.

MLS workflow may be realized in many different ways. However, composing an MLS workflow from multiple single-level workflows is the only practical way to construct a high-assurance MLS WFMS today. Hence, we presented a strategy for implementing an MLS workflow by composing multiple single-level workflows on a multiple single-level architecture. When independent multiple single-level workflows work together to achieve a higher mission, workflow interoperability is a vital element. We introduced an extended workflow interoperability model for that purpose. We then presented a method for decomposing an MLS workflow design into multiple single-level workflows for runtime.

References

[1] Anderson, M., North, C., Griffin, J., Milner, R., Yesberg, J. and Yiu, K. (1996). Starlight: Interactive link. *Proceedings of the Twelfth Annual Computer Security Applications Conference.*

[2] Atluri, V., Huang, W.K. and Bertino, E. (n.d.). A semantic based execution model for multilevel secure workflows. *Journal of Computer Security* (to appear).

[3] Chang, L.W. and Moskowitz, I.S. (1998). Bayesian methods applied to the database inference problem. *Proceedings of the Twelfth IFIP WG 11.3 Working Conference on Database Security.*

[4] Kang, M.H., Moskowitz, I.S. and Lee, D. (1996). A network pump. *IEEE Transactions on Software Engineering*, **22(5)**, pp. 329–338.

[5] Kang, M.H., Froscher, J. and Moskowitz, I.S. (1997). An architecture for multilevel secure interoperability. *Proceedings of the Thirteenth Annual Computer Security Applications Conference.*

[6] Kang, M.H., Froscher, J. and Eppinger, B. (1998). Toward an infrastructure for MLS distributed computing. *Proceedings of the Fourteenth Annual Computer Security Applications Conference.*

[7] Kang, M.H., Eppinger, B. and Froscher, J. (1999). Tools to support secure enterprise computing. *Proceedings of the Fifteenth Annual Computer Security Applications Conference.*

[8] Krishnakumar, N. and Sheth, A. (1995). Managing heterogeneous multisystem tasks to support enterprise-wide operations. *Distributed and Parallel Database Journal*, **3(2)**.

[9] METEOR home page, http://lsdis.cs.uga.edu/proj/meteor/meteor.html.

[10] Miller, J., Palaniswani, D., Sheth, A., Kochut, K. and Singh, H. (1998). WebWork: METEORs web-based workflow management system. *Journal of Intelligent Information Systems*, **10(2)**.

[11] Moskowitz, I.S. and Chang, L.W. (1999). The rational downgrader. *Proceedings of PADD'99*.

[12] OMG jFlow submission, ftp:///fpt.omg.org/pub/bom/98-06-07.pdf.

[13] Security Architecture for the AITS Reference Architecture, http://web-ext2.darpa.mil/iso/ia/Arch/SecArch1/SecArch1.htm

[14] Thomas, R. and Sandhu, R. (1997). Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management. *Proceedings of the Eleventh IFIP WG 11.3 Working Conference on Database Security*.