COMMON MODELS FOR CONFIGURATION MANAGEMENT AND AUTHORIZATION MANAGEMENT IN SYSTEMS ENGINEERING ENVIRONMENTS

Biju Kalathil and John Welsh
Lockheed Martin Advanced Technology Laboratories
1 Federal Street, A&E Building
Camden, NJ 08102
bkalathi@atl.lmco.com and jwelsh@atl.lmco.com

Mary Catherine Tuck, Mark Bailey, and Auti Zielhke
Intergraph
One Madison Industrial Park, MS GD3005
Huntsville, AL 35894
mctuck@ingr.com

Abstract. In an integrated product development environment that includes several vendor tools, diverse and incompatible configuration management mechanisms and authorization management mechanisms across tools can lead to inefficiencies in the design process. We describe in this document a common model of configuration management and authorization that may be adopted for an integrated product development environment. We specified a common minimal set of configuration management mechanisms and authorization management mechanisms that must be provided by the tools to support the proposed configuration and authorization models. We implemented a pilot program of the configuration management and authorization management models using Intergraph Corporation's Document Manager 2.0 (DM2) software system. This pilot implementation proves the plausability and usability of the model.

1. INTRODUCTION

Configuration management means managing different versions of design objects. It includes creating, approving, and releasing a new version of a design object; organizing the versions of a design object; and assembling compatible configurations of versions of design objects to form a release of a product. *Authorization management* supports the granting and revoking of access to data by users of a system, and it ensures that user authorization is checked before a user is allowed access to a data object. Different CAD vendor tools provide different mechanisms to support configuration management and authorization management. A *mechanism* is an individual function of a system. The mechanisms provided by the tools not only vary in scope, but also in their semantics. In an integrated product development environment (IPDE) that includes several vendor tools, diverse and incompatible configuration management mechanisms and authorization management mechanisms across tools can lead to the following inefficiencies in the design process:

- There is no common way to handle the configuration management and authorization management of a product throughout its life cycle
- The design engineers working on a project have to learn several different paradigms of configuration management and authorization management.
- The configuration management data on a product generated by one tool cannot be used by another tool
- The authorization information cannot be shared among the various tools used in the IPDE. Authorizations have to be specified separately in the various tools and managing the consistency of the authorization information among the tools in the IPDE is a management nightmare and a significant cost overhead.

We describe in this document a common model of configuration management and authorization that may be adopted for an IPDE. We specified a common minimal set of configuration management mechanisms that must be provided by the tools to support the proposed configuration and authorization models. An important criterion we followed in developing the model and the mechanisms is that they should be generic enough to allow an organization to adopt any configuration management process or authorization management policy it chooses. We implemented the configuration management and authorization management models as part of the design environment developed for the Rapid Prototyping of Application-Specific Signal Processors (RASSP) program.

The RASSP program is an Advanced Research Projects Agency/Tri-Service program to dramatically improve the way digital signal processors are designed, manufactured, tested, and procured. The RASSP program will deliver an integrated system that integrates the CAD tools used in its design process, which is known as the enterprise framework. An *enterprise framework* provides the facilities and services to integrate the automated processes of an enterprise. In the RASSP system, the enterprise framework supports workflow management, design data management, library management, computer-supported collaborative work, and remote tool access.

In sections 2 and 3 we describe the RASSP configuration management model and list a set of mechanisms specified by the model. In sections 4 and 5 we describe the RASSP authorization model and list the mechanisms specified by the model. In section 6 we detail how we implemented the models under the RASSP design environment. In section 7 we summarize our findings.

2. CONFIGURATION MANAGEMENT IN THE RASSP SYSTEM

2.1 Shared and Private Workspaces. *Workspaces* are partitions of the design object space that allow designers to selectively make their design objects visible to others in the project [Cattell, 1991]. Workspaces are organized hierarchically, as shown in Figure 1. There is a *global workspace* at the root of the hierarchy, *shared workspaces* as the intermediate nodes in the hierarchy, and *private workspaces* as the leaves in the hierarchy. The links in a workspace hierarchy represent a *parent-child* relationship between the linked workspaces.

Workspaces provide varying levels of sharing of data objects. A workspace user has visibility to all the

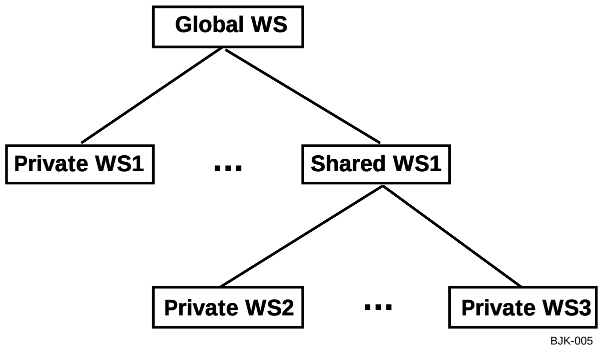

Figure 1. A workspace hierarchy

objects in the workspace and the objects in the ancestor workspaces of the workspace. All database users have visibility to data objects in the global workspace.

2.2 Data Object Versioning. We propose a data object versioning scheme where related data objects that evolve at the same time are grouped together as configurations, and versioning is managed at the level of configurations[1]. New configuration objects are typically created in a private workspace, at which point the configuration is considered a *transient version* of the configuration. A transient version may be updated or deleted. Once the transient version of a configuration reaches a state of maturity suitable for sharing with other designers in a project, it is promoted to a *working version* of the configuration, by checking in the configuration from the private workspace where it resides, to its parent workspace. A working version may not be updated, but it may be deleted. Working versions of configurations that represent the final state of design are promoted to *released versions* by checking into the global workspace. A released version may not be updated or deleted. We use the notation $state_i > state_j$ to denote that $state_i$ is a higher state that $state_j$. Thus *released* > *working* > *transient*

The versions of a configuration are organized as a directed acyclic graph, as shown in Figure 2, which is commonly referred to as a version tree. The following rule applies to a version tree: VT-Rule1: Given two versions c_i and c_j of a configuration c, such that $c_i \rightarrow c_j$, then the state of c_j should be less than or equal to the state of c_i.

3. RASSP CONFIGURATION MANAGEMENT MECHANISMS

The RASSP configuration management model specifies the syntax and semantics for the following mechanisms:

- Creating a workspace

[1] An individual RASSP enterprise framework tool may support versioning at the level of data objects also.

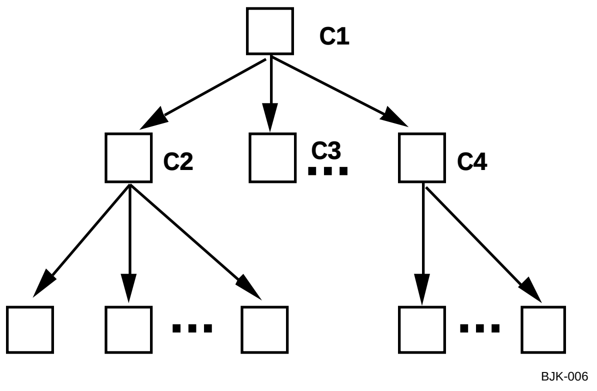

Figure 2. A version tree.

- Accessing an arbitrary workspace
- Accessing child workspaces
- Accessing the parent workspace
- Making a workspace visible
- Creating a configuration
- Inserting data objects into a configuration
- Checking out a configuration
- Checking in a configuration
- Accessing child versions
- Accessing the parent version
- Naming versions
- Retrieving a named version.

4. THE RASSP AUTHORIZATION MODEL

An authorization is a triplet $\{o_i, r_j, t_k\}$ where o_i is an authorization object in an authorization object hierarchy, r_j is a authorization role in an authorization role hierarchy, and t_k is an authorization type in an authorization type hierarchy [Rabitti, 1991]. An *authorization object* is a data object on which an authorization may be specified. Authorization objects in a database are organized as a directed acyclic graph, as shown in Figure 3. An *authorization role* is a collection of users who have the same set of authorizations on the same set of objects. The authorization roles in an organization are also organized as a directed acyclic graph, as shown in Figure 3. An *authorization type* is a type of operation that may be performed on a data object. The authorization types for a database are also organized as directed acyclic graphs, as shown in Figure 4. In Figure 5, the "Grant" authorizations are authorizations to grant an authorization to another role in the role hierarchy. The authorization type hierarchy for projects also contains the "Grant" authorizations. The directed links between two nodes in a hierarchy represent an *implication relationship* between the nodes.

An authorization may be *positive*, granting an authorization, or *negative*, revoking an authorization. An explicit or an implicit positive authorization $\{o_i, r_j, t_k\}$ has to exist for users to perform an operation of type t_k belonging to role r_j on a data object belonging to the authorization object o_i.

5. RASSP AUTHORIZATION MECHANISMS

The RASSP authorization management model specifies the syntax and semantics for the following mechanisms:

- Creating an authorization object
- Deleting an authorization object
- Adding a child to an authorization object
- Associating data files with authorization objects
- Retrieving an authorization object
- Retrieving the children of an authorization object
- Creating an authorization role
- Deleting an authorization role
- Adding a child to an authorization role
- Associating users with authorization roles
- Retrieving an authorization role
- Retrieving the children of an authorization role
- Retrieving an authorization type
- Retrieving the children of a node in the authorization type hierarchy
- Granting authorizations
- Revoking authorizations.

6. IMPLEMENTING THE RASSP CONFIGURATION AND AUTHORIZATION MANAGEMENT MODELS

We implemented a pilot program of the configuration management and authorization management models using Intergraph Corporation's Document Manager 2.0 (DM2) software system.

DM2 is an enterprise-wide electronic object management system that provides an object-oriented framework with functionality for storage, query, security, and usage control. DM2's architecture provides a graphical environment to assist users in quickly locating and using objects. DM2 manages information by ensuring that access is controlled and integrity is preserved throughout the life cycle of an object. Users can tailor DM2 to provide the following functions: administration of user, groups, and hosts; creation of a generic set generic set of object classes and relationships; object creation, storage, vaults, and queries; and rule-driven security.

Users implement the workspace hierarchy in DM2 using the features of *users* and *vaults*. Relationships

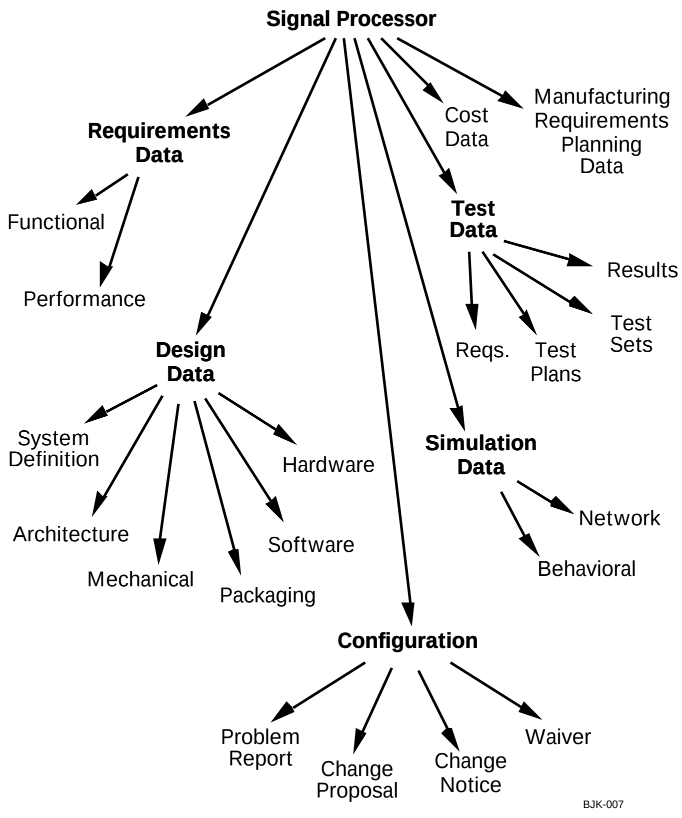

Figure 3. An example authorization object hierarchy.

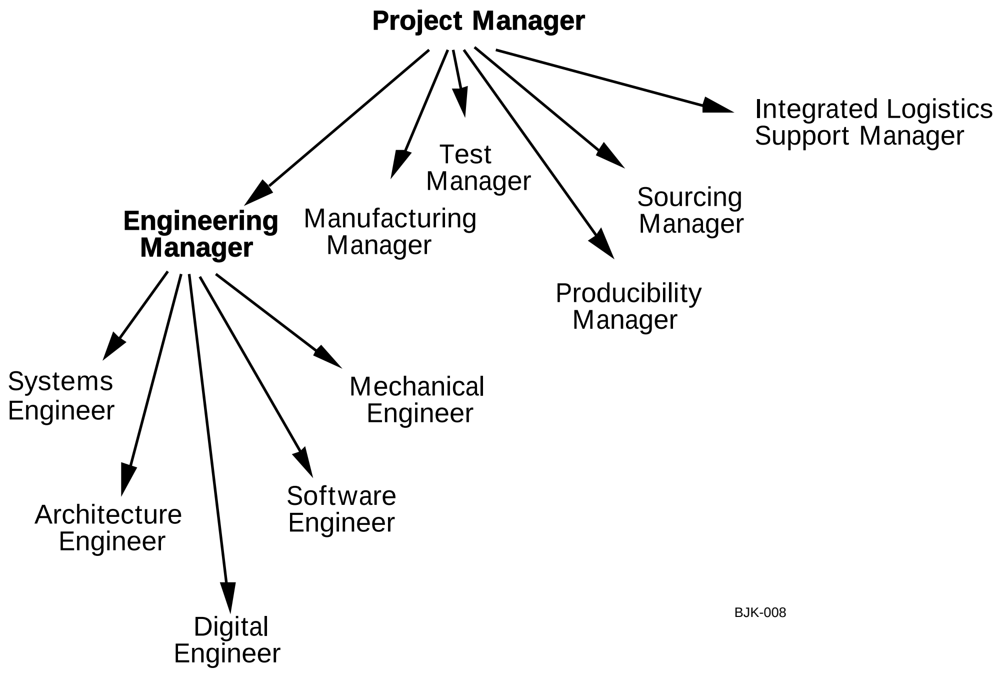

Figure 4. An example authorization role hierarchy.

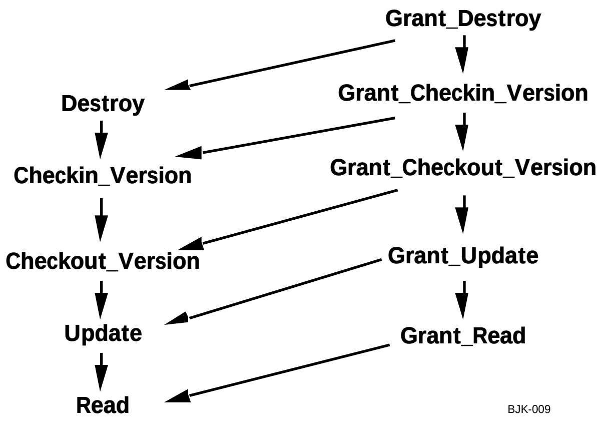

Figure 5. The authorization type hierarchy for design data objects.

between workspaces are enforced by defining *groups*, which contain related users, and limiting the access of these groups through the use of *rules*.

In DM2, each user has a private workspace. That is, there is a one-to-one mapping between a user and a private workspace. Based on projects, a parent-child relationship can be established between private and shared workspaces. Rules enforce the privacy of individual workspaces. Shared workspaces are implemented through vaults, which are logical collections of shared objects. Rules control access to a vault. User-to-vault relationships can be established to allow visibility of ancestor workspaces. The global workspace consists of selected data from all shared workspaces (vaults) obtained through the DM2 *query* capability. The DM2 implementation of the RASSP workspace hierarchy is shown in Figure 6.

Users map a *global workspace* in DM2 to all baselined objects in all vaults within a database. Items are visible in a global workspace through the DM2 Saved Query Object Class. Users map *shared workspaces* in DM2 to a *vault/vault location* that they can access by performing transfer, checkin, and checkout operations. A vault may contain data objects or actual file system items. A vault location provides a file system location for storing physical files owned by the vault. A private workspace may have more than one *work location.* Similar to vault locations, a work location provides actual file system space for objects residing in a private workspace.

DM2 provides out-of-the-box implementation of the RASSP authorization model by providing functionality equivalent to that of the triplet described in section 4. In addition, DM2 extends the definition of the triplet by adding a *condition* that defines the circumstances for an authorization role to perform an operation on an object. An authorization in DM2 may be described as a quadruplet $\{o_i, r_j, t_k, c_n\}$ where c_n is the *authorization condition.* In DM2, the quadruplet is a *message access rule.* In a message access rule, an authorization object is an object *class* on which an operation may be specified, an authorization role is a defined *group* or *user* for which the authorization is valid, and an authorization type is a *message group* that defines operations which may be performed on the object class.

7. CONCLUSION

The RASSP configuration management and authorization management model provides the core set of mechanisms necessary to support any policy or methodology an organization may choose to adopt in the respective areas. It is based upon mainstream approaches established in the database community and it provides a common approach that may be adopted by systems engineering environments. We implemented the models in the Intergraph DM2 product data management system to support systems engineering, using its extensibility and customization facilities without requiring changes to the source code. This pilot implementation proves the plausability of the model. We are exercising the implementation on the RASSP Design Process Benchmarking project to further study the model's usability.

REFERENCES

[Cattell, 1991] Cattell, R.G.G., Object Data Management, Massachusetts: Addison Wesley, 1991.

[Rabitti, 1991] Rabitti, F., Bertino, E., Kim, W., and Woelk, D., "A Model of Authorization for Next-Generation Database Systems," ACM Transactions on Database Systems 16(1): 88-131, March, 1991.

{ EMBED Word.Picture.6 }

Figure 6. DM2 workspace.

Legend:

⛁	=	Represents the physical file system location for the items residing in the associated vault (shared workspace) or user (private workspace). VL = Vault Location: Each vault location will be associated with only 1 vault. WL = Work Location: Each work location will be associated with only 1 user.
○	=	Represents a DM2 saved query. SQ = Saved Query: Every user will have access to the Global Workspace through a predefined query.
▭	=	Represents a logical collection of objects based on ownership.