ADST-II-CDRL-HLAIDE-9800083
March 31, 1998

ADVANCED DISTRIBUTED SIMULATION TECHNOLOGY II (ADST II)

HIGH LEVEL ARCHITECTURE TOOLS ANALYSIS AND INTEGRATION SUPPORT

(DO #0061)

CDRL AB01

FINAL REPORT

FOR: NAWCTSD/STRICOM
12350 Research Parkway
Orlando, FL 32826-3224
N61339-96-D-0002
DI-MISC-80711

BY: Lockheed Martin Corporation
Lockheed Martin Information Systems
ADST II
P.O. Box 780217
Orlando, FL 32878-0217

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	31 MAR 1998	final

4. TITLE AND SUBTITLE
Advanced Distributed Simulation Technology II (ADST-II) High Level Architecture Tools Analysis And Integration Support Final Report

5. FUNDING NUMBERS
N61339-96-D-0002

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Lockheed Martin Information Systems
ADST-II
P.O. Box 780217
Orlando Fl 32878-0217

8. PERFORMING ORGANIZATION REPORT NUMBER
ADST-II-CDRL-HLAIDE-9800083

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

NAWCTSD/STRICOM
12350 Research Parkway
Orlando, FL 32328-3224

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for Public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

The HLA Integrated Development Environment (HIDE) delivery order was to provide a focus for the analysis, evaluation, implementation, and integration of many tools and toolkits required by the High Level Architecture (HLA). This effort was intended to expand and enhance the initial prototyping effort discussed in the Advanced Distributed Simulation Technology (ADST) II HLA Integrated Development Environment (HIDE) Results of Analysis Report, Contract Data Requirements List (CDRL) AB01, 31 March 1998. That initial prototyping effort focused on the design and initial implementation of an integrated development environment to support High Level Architecture development and the integration of tools from the Defense Modeling Simulation Office (DMSO), Simulation, Training, and Instrumentation Command (STRICOM), Defense Advanced Research Projects Agency (DARPA) and others.

14. SUBJECT TERMS
STRICOM, ADST-II, HLA, simulation, Tools, DARPA, DISMO

15. NUMBER OF PAGES
57

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

Document Control Information

Revision	Revision History	Date
	Initial Release	03/31/98

EXECUTIVE SUMMARY 3

1. INTRODUCTION 5

1.1 PURPOSE 5
1.2 CONTRACT OVERVIEW 5
1.3 EFFORT OVERVIEW. 5
1.4 TECHNICAL OVERVIEW 5
1.4.1 Implementation of the HLA tools integration framework 6
1.4.2 Integration of selected object modeling tools into the framework 6
1.4.3 Integration of selected software engineering tools into the framework 6
1.4.4 Implement a new tool using the framework 7

2. APPLICABLE DOCUMENTS 7

2.1 GOVERNMENT 7
2.2 NON-GOVERNMENT 7

3. RESULTS AND ANALYSIS 7

3.1 IMPLEMENTATION OF HIDE 7
3.1.1 Catalyst Browser Issues 8
3.1.2 Catalyst Administration Tool Issues 9
3.1.3 Catalyst Support Tool Issues 9
3.1.4 Impact Analysis Issues 10
3.1.5 Process Definition Issues 10
3.1.6 Process Enactment Issues 12
3.1.7 Loose Integration Issues 13
3.1.8 Tight Integration Issues 14
3.1.9 Levels of Integration 15
3.1.9.1 Level 1: Access to Tools and Files within a Process 16
3.1.9.2 Level 2: Bi-directional Translation of Data into CORBA/Catalyst 16
3.1.9.3 Level 3: Linking CORBA Objects from Different Tools/Databases 17
3.1.9.4 Level 4: Completeness and Consistency of Integrated Data 17
3.1.10 Common Object Model 18
3.1.11 Semantic Consistency and Completeness 19
3.1.12 Automatic Linking of Data 20
3.1.13 FOM Generation Issues 20
3.1.14 Documentation Generation and Navigation 22
3.1.15 FEDEP Process Enactment 25
3.2 INTEGRATION OF CURRENT HLA TOOLS 27
3.2.1 CAFDE 27
3.2.2 Federation Management Tool (FMT) 30
3.2.3 Federation Data Collection Tool 30
3.2.4 TASCVision 30
3.2.5 TASC OMDT 30
3.2.6 Visual OMT 31
3.2.7 OMDT Pro (HLA Lab Works) 31
3.2.8 FedProxy (HLA Lab Works) 31
3.2.9 FedDirector (HLA Lab Works) 32
3.2.10 Conceptual Modeler (HLA Lab Works) 32
3.2.11 Federation Composer (HLA Lab Works) 33
3.2.12 Scenario Generation Tool (SGT) (HLA Lab Works) 33
3.2.13 Scenario Execution Planning Tool (SEPT) (HLA Lab Works) 33
3.2.14 Scenario Monitor (HLA Lab Works) 34
3.2.15 Federation Test Suite (FTS) 34

3.2.16 OSim *35*
3.2.17 ModOIS *35*
3.2.18 ModSAF scenarios *35*
3.2.19 RTIME *35*
3.2.20 Warfighting Analysis and Integration Center Java PVD *35*
3.2.21 Federation Execution Planner's Workbook Tool *36*
3.2.22 Federation Verification Tool *36*
3.2.23 XGen *36*
3.2.24 DEM *36*
3.2.25 TASC HLA Construction Kit *36*
3.2.26 Modus Operandi Scenario Generation Tool *37*
3.3 INTEGRATION OF SOFTWARE ENGINEERING TOOLS 37
3.3.1 Chimera *38*
3.3.2 OzWeb *38*
3.3.3 WinWin *39*
3.3.4 EMMA *40*
3.3.5 Sybil *41*
3.3.6 Together/J *41*
3.4 IMPLEMENTATION OF HLA ARBITRATION TOOL 41
3.4.1 Comparing SOMs with other SOMs and FOMs *42*
3.4.1.1 Object Correlation Metrics 42
3.4.1.2 Attribute Correlation Metrics 45
3.4.1.3 Incremental Decomposition and Abstraction 46
3.4.2 SOM-FOM translation *47*

4. SCENARIO **49**

5. CONCLUSION **49**

6. POINTS OF CONTACT **52**

7. ACRONYMS **53**

EXECUTIVE SUMMARY

The HLA Integrated Development Environment (HIDE) delivery order was a study and prototyping effort focused on the design and initial implementation of an integrated development environment to support High Level Architecture (HLA) development and the integration of tools from the Defense Modeling Simulation Office (DMSO), STRICOM, DARPA and others. The focus of the effort was on the analysis, evaluation, implementation, and integration of the tools required to support the Federation Development and Execution Process (FEDEP) of the HLA. HLA tools currently being developed are not interoperable and most are being developed in a stovepipe fashion or with the expectation that data interchange formats (DIF) will solve all the interoperability needs. A common integration framework is needed that supports a common object representation for domain information, allowing data from various tools to be shared and semantically linked, thus giving an integrated view of related HLA FEDEP data. The framework should also support the continuous evolution of tools, standards, and applications.

The objectives of the HIDE effort were:

1. **Implement an initial version of HIDE**. This objective involved using Catalyst as a basis for HIDE. Catalyst is a CORBA-based (Common Object Request Broker Architecture) integration framework developed by Modus Operandi under sponsorship from Rome Labs and DARPA's Evolution Design of Complex Software (EDCS) program. Catalyst meets the goals specified above by providing common CORBA object models for data and providing mechanisms importing tool data into the common object representation and exporting it out of the same representation. Catalyst also supports the semantic linking of these objects so that integrated information can be browsed in an appropriate fashion. A basic functionality was to have been demonstrated by developing a prototype ModSAF Simulation Object Model (SOM) in the common object representation. However, the focus of the effort was changed to Federation development and a ModSAF-based Federation Object Model (FOM) was developed instead.

2. **Integrate current HLA tools**. This objective was to integrate robust HLA tools that were available, low cost, useful, and met the customer's needs. At the time, the STRICOM Object Model Development Tool (OMDT) developed by TASC was the only UNIX tool that met these requirements. It was "loosely" integrated using integration scripts that parse OMDT DIF files, creating CORBA objects and vice-versa. As part of this objective, other candidate HLA tools were also to be identified for future integration.

3. **Integrate selected software engineering tools**. This objective was to identify and integrate appropriate tools from the software engineering domain. Such tools include information web/browsers, information linking tools, process enactment tools, impact analysis tools, requirements negotiation tools, design management tools, code generation tools, and reengineering tools. The DARPA EDCS program was the primary source of these tools. Catalyst itself already provided tools in the area of process enactment, impact analysis, information browsing, information linking, and requirements negotiation and

thus the goal was to demonstrate their functionality in an HLA federation development environment.

4. **Implement an initial HLA arbitration tool**. A need was identified for a new HLA tool able to support a developer in creating new Federation Object Models (FOMs) by providing the capability to compare Simulation Object Models (SOMs) with FOMs. This comparison allows a developer to determine where the SOM from a new federate should be placed in the FOM and also allows the developer to find the best match of an existing FOM with the candidate federate SOMs. A prototype of this kind of tool was to be developed. Because of competing efforts by Aegis Research, the development of this tool was put on hold. Only requirements and design information was gathered for this tool.

5. **Develop a detailed plan for the implementation of a robust version of HIDE and the tools within it**. This plan became the proposal for the follow-on work and thus is not presented in this document. Refer to the *High Level Architecture Tools Analysis and Integration Support* proposal for more information.

The following report contains the results of these objectives. The issues associated with the use of Catalyst as the basis for HIDE are discussed as well as changes deemed necessary to the Catalyst tool suite. Also presented is a brief analysis of the various HLA, EDCS, and other tools available with emphasis on their capabilities and their appropriateness to the HLA. Tools that were integrated are also discussed, along with any appropriate issues.

Through the course of this work the effort shifted focus to the HLA FEDEP model. As a result, there is some discussion concerning the development of the FEDEP in HIDE and its enactment, along with issues concerning tools and the FEDEP. As this work progressed, other topics were discovered, such as the automatic generation of FOMs, loose versus tight integration of tools in HIDE, the automatic linking of integrated data in the FEDEP process, automatic documentation generation and navigation, and the semantic consistency and completeness of integrated FEDEP data. These are all discussed in this report.

1. INTRODUCTION

1.1 Purpose

The purpose of this final report is to document the ADST II effort which created the initial version of HIDE. This report includes a full description of each effort pursued under each initial objective, any lessons learned, and the results of analysis.

1.2 Contract Overview

HIDE was performed as DO #0061 under the Lockheed Martin Corporation (LMC) ADST II contract with STRICOM. The contract required LMC to create a prototype HLA development environment and perform a study (much like a Mini-FAS) for a more robust environment to be developed under follow-on work. The effort was successfully completed on 31 March 1998. The results are documented in this final report (CDRL AB01).

1.3 Effort Overview.

The HLA Integrated Development Environment (HIDE) delivery order was a study and prototyping effort to implement an integrated development environment to support High Level Architecture (HLA) development, and it included the integration of tools from the Defense Modeling Simulation Office (DMSO), STRICOM, DARPA and others. The focus of the effort was on the analysis, evaluation, implementation, and integration of the tools required to support the Federation Development and Execution Process (FEDEP) of the HLA. HIDE was used as the foundation of this effort. Based upon the Catalyst framework, it provided a common object representation for the integration and semantic linking of the data from various HLA and software engineering tools.

1.4 Technical Overview

The technical approach to the HIDE effort involved the data collection, evaluation, analysis, implementation, integration, and demonstration support of HIDE and its corresponding tools. The effort leveraged existing freeware toolsets, DMSO tools, and DARPA tools to provide a foundational framework for the tool suite. Specifically, the technical approach involved four tasks:

1. Implementation of the HLA tools integration framework;
2. Integration of selected object modeling HLA tools into the framework;
3. Integration of selected software engineering tools into the framework;
4. Implement a new tool using the framework.

The following is a short synopsis of each technical effort.

1.4.1 Implementation of the HLA tools integration framework

Catalyst was used as a basis for developing HIDE. Catalyst is a CORBA-based (Common Object Request Broker Architecture) integration framework developed by Modus Operandi under sponsorship from Rome Labs and DARPA's Evolution Design of Complex Software (EDCS) program.. Catalyst provides common object models for data and provides mechanisms for the integration of data from tools into and out of the common object representation. Catalyst also supports the semantic linking of this data so integrated information can be browsed in an appropriate fashion. A basic functionality was to have been demonstrated by developing a prototype ModSAF Simulation Object Model (SOM) in the common object representation. However, the focus of the effort was changed to Federation development and a ModSAF-based Federation Object Model (FOM) was developed instead.

Through the course of this task, the effort shifted focus to the HLA FEDEP model. As a result, issues concerning the development of the FEDEP in HIDE and its enactment in the Catalyst process enactment tool were discovered. As this work progressed, it became clear that other important issues needed to be addressed, such as the automatic generation of FOMs, loose versus tight integration of HLA tools, the automatic linking of integrated HLA data, automatic documentation generation and navigation, and the semantic consistency and completeness of integrated FEDEP data.

1.4.2 Integration of selected object modeling tools into the framework

Under this task, the STRICOM Object Model Development Tool (OMDT) developed by TASC was "loosely" integrated into Catalyst using integration scripts that could parse OMDT DIF files, creating CORBA objects, and vice-versa. Also under this task, other candidate HLA tools were identified for future integration. Special attention was given to tools that met the following criteria: (1) robustness, (2) availability, (3) inexpensive, (4) useful, (5) meets customer needs.

1.4.3 Integration of selected software engineering tools into the framework

Under this task, WinWin, an EDCS requirements capture and negotiation tool developed by the University of Southern California Center for Software Engineering, was integrated into HIDE. WinWin was already loosely integrated into Catalyst but WinWin did not work under the current HIDE environment nor were there any HLA artifacts. Also performed under this task was the identification of potentially applicable tools from the software engineering domain. Such tools included information web/browsers, information linking tools, process enactment tools, impact analysis tools, requirements negotiation tools, design management tools, code generation tools, and reengineering tools. The DARPA EDCS program was the primary source of these tools. Catalyst itself already provided tools in the area of process enactment, impact analysis, information browsing, information linking, and requirements negotiation and thus the goal was to demonstrate their functionality in an HLA federation development environment.

1.4.4 Implement a new tool using the framework

A need was identified for a new HLA tool that to support a developer in creating new FOMs by providing the capability to compare SOMs with FOMs. This comparison allows the developer to determine where the SOM from a new federate should be placed in the FOM and also allows the developer to find the best match of an existing FOM with the candidate federate SOMs. Under this task, a prototype of this tool was to be developed. However, because of competing efforts by Aegis Research, the development of this tool was put on hold. Only requirements and design information was gathered for this tool.

2. Applicable Documents

2.1 Government

ADST II Work Statement for the High Level Architecture Tools Analysis and Integration Support, September 3, 1997, AMSTI-97-WO84.

2.2 Non-Government

ADST II Technical Approach for the High Level Architecture Tools Analysis and Support Delivery Order (HIDE98), February 13, 1998, ADST II-TAPP-HIDE98-9800046

3. Results and Analysis

3.1 Implementation of HIDE

The Catalyst framework developed by Modus Operandi serves as the framework for HIDE. It is a CORBA framework that provides a set of tools to support the integration of data as CORBA objects and navigation among objects within the resultant knowledge base. In order to met the needs of HIDE, several problems have been identified a that should be corrected, complemented by recommended enhancement to make HIDE a more powerful environment, such as automatic FOM generation, automatic linking of data, loose versus tight integration of tools, the levels of tool integration, semantic consistency and completeness, automatic document generation and navigation, a common domain object model, and FEDEP enactment; all of these issues need to be explored in follow-on work to create a HIDE that is useful for large federations Each of these problems and recommended enhancements is discussed within the context of the tool to which it relates.

3.1.1 Catalyst Browser Issues

Cosmetic Changes.

1. Change the Title bar of the browser and the popup selector to *HIDE Information Browser*.
2. The object locator should be changed to something like *HIDE Information Locator*. This would have to change on the Browser Window menu, the popup selector, and the title bar of the object locator itself.
3. The machine and workgroup contexts need to also be removed from the viewer and only displayed with some command line option. These just confuse things and allow the user to do potentially harmful things.

Saving.

It is unnatural to require the user to save objects before they can be copied and pasted. If this is a CORBA requirement, then the objects should be saved automatically if needed as part of the copy process.

Orphaned Objects.

There needs to be a way to get a list of all available object servers so that objects that are orphaned can be deleted. Perhaps a better way to accomplish this would be to prompt the user for confirmation when removing an unreferenced object from the name service If the user then "ok's" removal from the name server, the object may be deleted as well.

Multiple Object Servers.

There needs to be an easy way within Catalyst to specify different object servers for the same class. This will account for the case of more than one tightly integrated tool providing the same class data.

GUI.

1. There needs to be a horizontal scrollbar on the *edit attributes* window.
2. The user should be able to click on the name of the attribute to edit, in addition to the field area.

Tool Invocations.

There needs to be a quicker way to invocate a tool from the annotation object. Perhaps the user could double right click on an annotation object and it would automatically call the display function to invoke the tool.

Schema Browser.

There needs to be a Catalyst schema browser. Links to schemas and instances can provide many different uses and would help in querying of data. As part of an evolvable framework, we may want to think more about schema maintenance.

OCI scripting.
A TCL command is needed to refresh objects automatically. For this effort, TCL scripts have been written that execute when the user clicks on "display" in the *edit attributes* window. The script takes all the Catalyst data and exports it to an HLA DIF/FED file format, then automatically executes the appropriate tool with that file. Next, upon closing the tool, another script is run that automatically converts the newly modified data back to Catalyst (using mapping files to reuse existing objects and delete ones that were deleted in the tool). Since the data has changed in Catalyst, the display in the Catalyst browser needs to be refreshed and started over from the root object. Refreshing the root object doesn't seem to work: the root has to be deleted and retrieved from the locator again. A TCL command that would do this automatically would go a long way towards making the loose integrations act more like a tight integration, while also enforcing consistency among the tool and Catalyst data.

3.1.2 Catalyst Administration Tool Issues

Clear and import.
The Administration tool needs a mechanism to clear all process data before re-importing it. In the current version, each process/role/work product has to be individually deleted from a scrolling list. There needs to be a capability to allow the user to select a range of items (or all) from the scrolling list, mark them, and delete them. Users should also be warned, however, that they may be deleting common objects on a global edit.

Saving .ver files.
For export into Catalyst, the user should be able to save to any filename with a *.ver* extension and load it, rather than having the tool automatically save it as *CPD.ver* file. Consequently, the Administration tool should allow the user to enter the name of the export file.

3.1.3 Catalyst Support Tool Issues

Deleting Relations.
Currently, relations can be added to the system but never deleted. This is a problem if a relation is accidentally created.

Creating Relations.
A GUI interface is needed for the definition of relations. The current approach is via a text file which is inadequate for most users.

3.1.4 Impact Analysis Issues

Window resizing.
When the *Impact Definition* window is resized, the *Available Relations* scrolling list covers the [move >>] button. The window has to be closed and reopened to get it back.

Defaults.
The "Open" selection did not provide the default name "new impact analysis"; it was blank at first. This should not be the case.

Label area.
The return command does not work. The user must click the mouse instead.

Analysis results.
For some reason, only partial data was copied into the solution columns from Impact Analysis (2 in column 1 and none in the rest.)

Impact definition.
The issue here is why do selected relations have to be in order?

Not equal does not work.
In Impact Definition you cannot say [state != "initial"] and have it test correctly.

3.1.5 Process Definition Issues

Because the process definition tool is written using InSight, a majority of the issues with the tool result from the interface of InSight. The interface does not work the way other mouse interfaces work (using different buttons, double clicking required in some places, menu bar toggles) making it difficult to use and in some cases causing work to be lost. This tool should be ported to Java so that a consistent interface can be used across platforms.

Quick delete.
When the user selects the File menu bar with the right button, it toggles the option to close instead of bringing down the menu. Since there also is no confirmation of closing, work can, and was, lost numerous times.

The meaning of "work product."
It is unclear that a "work product" is an object in Catalyst/InSight that is a symbolic reference to the "actual work product" that stakeholders might be working on. This needs to be pointed out, maybe with a picture. Also, it would be nice to have some easy way of linking the Catalyst/CORBA object to the actual artifact with an established semantic relation. [Note: this may be done through Annotations.] In that way, instances of objects could be related to instances of artifacts. It would be a much easier thing to do version control and CM using

scripts/rules to monitor changes to the document and compare it to the last edit date, for example.

InSight work product attribute definition and display.
It would be nice to be able to define attributes of objects (especially work products) in InSight as you are setting up your entry and exit conditions, or at least have some easy way to look at what attributes are available to use in the conditions. With the current mechanism, the user has no way to verify which attributes could be tested against or what they were from this window.

Window back tracking.
If the top level window is accidentally closed, the current work can never be saved since the save option is only present in this topmost window. Closing this window should not only require confirmation but should close all the child windows as well, essentially closing the application.

Selecting text to edit.
The InSight tool makes it very hard to highlight the text fields for editing. Apparently, the mouse has to be in an exact pixel location at the end of the field. This makes editing very frustrating.

User defined roles.
The tool's predefined roles are too limiting. The capability needs to be added to allow the user to define their own roles such as "customer" or "database administrator".

Macro editing functions.
The user needs to be able to cut and paste "composite data" defined, i.e. steps, methods, tasks, activities, work products to quickly evolve the process. It is tedious to define, for example, all the steps for all the methods for all the tasks for all the activities from scratch, especially since the same basic approach is reused for all activities. If one activity is defined to the step level and then copied with changes such as its name, the work product it inputs, and the tools its steps invoke, it would have taken 1/10 the time to develop the FEDEP process. This is difficult due to the creation of many objects with unique ID's and due to some of the semantic conflicts and ambiguities, but some such functions need to be there to make it easier to use.

Undo deletion.
In many places, items (e.g. an activity and all its children) can accidentally be deleted. The tool needs to either warn the user or allow the deletion to be undone.

Save on exit.
The tool should warn the user to save on exit.

Reorder steps.
The tool provides no mechanism to reorder steps. If a step is added later that needs to be placed between two others, the user must delete the activity and redo all its steps in the proper order.

Scrolling in steps window.
After the user defines five steps, new steps cannot be scrolled. A dummy step must be defined to get the new step to scroll up.

Reentrant activities.
The capability needs to be added to allow the user to create a work product, exit the activity, and reenter the activity later to edit the document. Currently, this cannot be done within the same activity. This is critical for the FEDEP, which can be used iteratively or will reuse previous federation artifacts.

Optional Work Products.
There needs to be a way to specify that a work product is optional and allow the activity to be entered without it. This is necessary to support different, tailorable paths through the same process, where certain steps can use a previous work product as a starting point or start from scratch (e.g. FOM development).

Saving .ver files.
For export into Catalyst, the user should be able to save to any filename with a *.ver* extension and load it, rather than having the tool automatically save it as *CPD.ver* file.

Tight Integration.
The tool should be more tightly integrated with Catalyst so the user does not have to go through the Administration tool to export the data into Catalyst. The tool would have be modified to allow the user to specify the project.

Launching of tools.
Some modifications may need to be made to support the launching of tools from machines other than the server and differentiating between launching UNIX tools and Windows tools.

3.1.6 Process Enactment Issues

Roles and execution.
Roles need to be optional. Currently, in order for tasks to be run, the role:<none> must be specified at a minimum. If there is no particular person/role required for the task, there should be no role annotation. This is cleaner than specifying "none".

Error 2.
In the "No WP for WPI (error 2)" error, you cannot get out of the error condition. (WpiSet::PrivateAddWpiList 2 ERROR).

Not equal test.
In the test conditions, the parser does not seem to like the logical statement [x.state != "initial"]. The logic has to be changed to [x.state = "draft" | x.state = "final"] for it to work. This is cumbersome if there are many states.

Clear and reload process definition.
The tool needs to provide the user with an option to clear a process and reload it without exiting the tool and re-launching it.

Process Enactment.
It would be very helpful to provide an easy way to semantically link work product instances to actual file artifacts in a way that allows users to get access to the artifact from the WPI, whether that is in the browser or the enactment tool or elsewhere. That is, a "problem statement" work product is linked through a "path" attribute to the actual file and through an "application" attribute to its application. A script is used to append these into a command line command.

3.1.7 Loose Integration Issues

The loose integration is the simplest and lowest cost method of integration. In a loose integration, scripts or programs are written to read data files produced by the tool and then store the information the files contain in Catalyst as CORBA objects. Scripts or programs can also be written which take information stored in Catalyst and write it to files formatted for input back to the integrated tool. This approach uses data files as the method for passing information between the integrated tool and Catalyst. This strategy works very well for tools which do not provide callable Application Programmer Interfaces (APIs) or for tools intended to be single user tools. Examples of such tools are spreadsheets, word processors, scheduling tools, graphing programs, etc.

Catalyst contains high level scripting commands to make loose integrations easy to create. A script for transferring data to and from an Excel spreadsheet would take roughly 8 hours to create. Bi-directional scripts for the FED and OMT DIF files were created with the effort specified below. Some time was spent on the OMT DIF because of the large amount of parsing required. Effort for the FED scripts is skewed since part of the time was used learning TCL, the scripting language. Initially, new objects could not be created since HIDE did not contain the development toolkit that allows users to create their own servers. Thus, the predefined Component class had to be used, which is why FOM attributes were expressed as relationships instead of object attributes. The listing of the number of objects, attributes, and relationships was an attempt to get an idea of the complexity of the objects being used by the script, since this has an important bearing of the level of effort.

HIDE METRICS						
Integration	SLOC	Effort (manhours)	SLOC/manhour	# of Catalyst Objects	Avg # of attributes per object	Avg # of source relationships per object
Catalyst->FED	275	4	68.75	10	0	1
FED->Catalyst	640	24	26.66666667	10	0	1
Catalyst->DIF	600	8	75	26	0	5
DIF->Catalyst	2051	40	51.275	26	0	5
TOTALS:	3566	76	46.92105263	36	0	6
AVERAGES:	891.5	19	46.92105263	18	0	3

An important issue for loose integrations is keeping the data consistent between Catalyst and the tool. If the tool is used outside of Catalyst, this cannot be guaranteed, unless the user executes the import/export scripts. However, if the tool is launched from the process enactment tool some steps can be taken to ensure cross-tool consistency Under this effort, shell scripts were developed that can be launched from the FEDEP to convert the current Catalyst FOM data into a DIF file which was then launched as a command line argument to the OMDT (Unfortunately, TASC's OMDT never did process the command line argument correctly). Upon exit from the OMDT, the edited DIF file was converted back to Catalyst (with existing CORBA objects reused and deleted DIF artifacts removed as well). This helps insure consistency between loosely integrated tools and Catalyst.

The disadvantage of loose integrations is that source data from a tool needs to be dealt with in batch mode, as entire file; access to data on an object-by-object basis is not possible using loose integration scripts.. For example, if one FOM class was to be changed with the OMDT, the entire FOM would be written to a DIF file, edited within the OMDT, and converted back into Catalyst objects, all for a single change to one object Depending upon the tool, there may be possibilities for working on partial files, but that will have to be investigated on a case-by-case basis.

3.1.8 Tight Integration Issues

The tight integration strategy is more complex than loose integration but provides real time access to the integrated tool's data. In a tight integration, a server program is created which interfaces with the tool by calling its API. Requests for data from Catalyst are translated by the server into calls to the tool's API to retrieve the data. Only the data retrieved by the API is converted into Catalyst objects.

The structure of all the server programs is defined by Catalyst. A toolkit is provided with Catalyst for generating the standard server structure code; the tool integrator merely has to fill in the parts which are unique to the tool being integrated. The ORACLE database was integrated into Catalyst to provide read-only access to ORACLE data in 160 hours. Write access could be provided in another 160 hours of effort. ORACLE is a very complex tool to program; simpler tools could be integrated in less time.

One feature of Catalyst tight integration is that it does not require changes to the tool being integrated (source tools), unless the server needs to be linked in. Therefore, the owners of a

source tool do not need any development tools and do not need to make any changes to their tool, provided the tool already has an API. If source tools do not produce data files and do not have APIs, then some work may have to be done on the part of the tool provider to support one of these features.

Unfortunately, tight integration is only valuable for tools that support a client-server architecture, such as databases. Since Catalyst gets its information from the tool's repository, the tool needs to have a back end server running that can be queried through an API. Tight integration should not require a tool's front end to be running. For tools that store their information in files, loose integration is the preferred integration technique; In loose integration situations, data must be kept consistent.

Another problem with tight integration is that the tool interface is linked in with the Catalyst object server. Typically, only one tool can supply data for a given class of object. Conflicts arise if two tools provide the same kind of data from different sources. It may also be advantageous for one server to be run on multiple machines to distribute the load. With the loose integration, this is not a problem. If the object factory for a given class from a given tool can be specified via the name service for the server to use then the tightly integrated tool to use can be specified. Currently there is no way to do this in Catalyst.

3.1.9 Levels of Integration

Integration can be subcategorized into four levels in increasing level of difficulty.

Level 1, the simplest level of integration, involves "process integration," in which a tool is integrated into an executable process model, from which it can be invoked. Level 1 involves no data integration, but rather provides users with access to tools within the process context in which they are to be used. For example, ModSAF has been linked into the Scenario Development activity of the FEDEP executable process model to support federation developers in laying out battlefield scenarios. In this case, ModSAF can be invoked from within the executable process model and used in context for scenario development.

Level 2 integration is the "bi-directional translation of data" into Catalyst, which is where the traditional loose and tight integrations come into play. In level 2 integration, source data are converted into CORBA objects and CORBA objects back into source data. For example, "M1A2 tank" data from the ModSAF scenario files may be translated into CORBA "M1A2 tank" objects in Catalyst. These objects might then be translated into DIF formats for OMDT tools.

Level 3 integration involves the "automatic linking of integrated data" with other data from other tools. At this level, CORBA objects are linked to each other by semantic relations defined for the domain. For example, CORBA M1A2 tank objects originating from ModSAF may be linked by a "has-requirement" semantic relation to a "night vision" requirement object originating from the WinWin requirements tool.

Level 4 involves keeping this data complete and consistent as the related data changes. If the "night vision" requirement is deleted, for example, which related objects will be affected? Clearly, in this example, the "M1A2" tank behaviors will be affected.

Below, we discuss tool integration by level. The characteristics of each level are discussed first, followed by a discussion of work performed at each level. Future integration work will be scoped in terms of these levels of integration. .

3.1.9.1 Level 1: Access to Tools and Files within a Process

Level 1 is identified by the following characteristics:

- Organizes tool/file access for users within the FEDEP Process
- Serves to guide users through the process
- Ensures common tools/files are used across the process
- Helps to structure thinking about what tools are appropriate for what activities
- Ensures conditions are met to start/perform/complete work on work products (CM)
    - Input status checked
    - Authorized personnel checked
    - Output status checked

Under the FEDEP executable process work several tools were integrated and demonstrated at level 1:

1. WinWin: this requirements development and rationale capture tool is integrated into the *Conceptual Model Development* and *FOM Development* activities of the FEDEP.
2. OMDT: the Object Modeling and Development tool is also integrated into the *Conceptual Model Development and FOM Development* activities.
3. ModSAF: this simulation system has been integrated to support the *Scenario Development* activity. ModSAF allows developers to lay down forces on a battlefield and to use the resultant objects as the basis for a Federation Object Model (FOM), also known as an Ideal FOM.
4. Xemacs: is available in various FEDEP activities for documentation.
5. Netscape: is also available at any time to access HLA standards documents, as well as the MSRR, the OML, and so forth, as referenced in the FEDEP model.

3.1.9.2 Level 2: Bi-directional Translation of Data into CORBA/Catalyst

Level 2 integration is characterized as follows:

- Data structures represented as common CORBA objects
- Common GUI for browsing objects and object networks
- Data may be edited in Catalyst and exported to a tool, or *vice versa*

- Impact analysis (limited) may be performed on CORBA objects/network
- Serves as common knowledge representation required for Level 3 integration

As mentioned previously, OMT DIF data was represented in Catalyst as a CORBA object network. The DIF data can be changed via the Catalyst browser itself and exported to the OMDT or *vice versa*. Similarly, the FEDEP model is available within Catalyst as well and can be changed either within the browser tool or within the process definition tool.

3.1.9.3 Level 3: Linking CORBA Objects from Different Tools/Databases

Level 3 is where the power of the integrated data model comes into play. This level is identified by the following characteristics:

- Links related objects from multiple tools in meaningful ways
- Supports Catalyst browsing of integrated data as networks of objects
- Supports data transfer from tool to tool (import from one, export to another):
    - Promotes data sharing across tools
    - Facilitates use of different (similar) tools within common environment
    - Supports evolutionary migration from legacy tools (databases) to new tools (databases) with no impact
- Supports analysis of integrated data through:
    - Browsing
    - Impact analysis
    - Custom analysis scripts
    - Data exported to analysis tools
- Serves as common knowledge representation required for Level 4 integration

Using Level 3 integration, Federation "objectives objects" can be semantically linked to Conceptual Model "requirements objects" which can be linked to "scenario objects" which can be then be semantically linked to FOM classes and interactions. Furthermore, impact analysis can identify what FOM objects are affected when requirements change. Research is required to determine how this linking can be done automatically (see Automatic Linking of Data).

3.1.9.4 Level 4: Completeness and Consistency of Integrated Data

Level 4 represents the knowledge management of enterprise information, in this case all the integrated data associated with the FEDEP and its tools. In order for a framework like HIDE to be flexible over time it must have the following level 4 characteristics:

- Maintain semantic correlation of data/objects as they change
- Understand impacts of change (robust) to correlated data/knowledge base across projects and processes

- Intelligent support to team members for system design, development, and maintenance
- Automated/semi-automated updating of information/knowledge base across tools
- Ensured accuracy of analyses based upon ensured consistency and completeness of data/knowledge base

As an example, if a FOM requirement is deleted, added, or modified, semantic consistency and completeness can help answer the following questions:

- How do I know when the FOM is consistent with this change?
- What products/objects will be affected and how will they be altered?
- How can the integrated data/knowledge base be automatically transformed to maintain consistency?
- What will the change affect cost?
- How will the change impact schedule?

Of these four levels of integration, the most important are levels 3 and 4, but the most visible are levels 1 and 2. Future work will focus considerable effort on levels 3 and 4.

3.1.10 Common Object Model

To achieve the benefits of level 3 and level 4 integration, a common object model will need to be developed for all HIDE data/knowledge. This common model will allow tools to more seamlessly share data. The common model is also important for allowing complementary tools to each produce part of the final product, such as a FOM. The common modeling representation will also provide the meta-data necessary to support automatic linking and semantic consistency and completeness maintenance.

As a starting point for this domain model, common classes have been created in Catalyst to represent OMT DIF information. The tight integration mechanism is the same mechanism used to create custom object servers only no tool access code is required (unless tightly integrated of course) . The following object servers were created for HLA FOM-specific classes:

- OMTAffectedAttribute
- OMTAssociation
- OMTAssociationMember
- OMTAttribute
- OMTClass
- OMTComplexComponent
- OMTComplexDataType
- OMTComponent
- OMTDIF
- OMTEnumeratedDataType
- OMTEnumeration

- OMTInteraction
- OMTInteractionMember
- OMTNote
- OMTObjectModel
- OMTParameter
- OMTParticipatingClass

However, the loose integration scripts developed for this UDO used the provided Component classes and thus will need to be modified to use the new OMT classes in follow-on work. Originally, each attribute, parameter, characteristic, etc. of the FOM was a separate instance of a component class. These are now members of the new OMT classes and thus not represented as individual Component objects as viewed in the browser. The only disadvantage of this approach is that some of the relations must have the attribute name in the relation, such as *datatype_note_reference,* since individual attributes cannot be related directly to, only objects can. Using the Component class approach, a datatype attribute would be a separate Component object that would have a note_reference relation to some note object.

3.1.11 Semantic Consistency and Completeness

As mentioned previously, the semantic consistency and completeness of integrated data is critical for a robust development environment , in which project-wide information is continually changing. As changes are made, the impacts to existing data need to be identified and, where possible, automatically corrected. In order for this to be possible a domain meta-model and ontology/lexicon needs to be created. This meta-model is a semantic network that includes the domain classes, as well as required and optional relations that define the linkages among the object classes. Such relationships would include those that link *federation objectives* to *federation requirements* , and *federation requirements* to *FOM objects*.

The meta-model should also include categories of artifacts. Requirements, for example, should be subclassified as *performance requirements, interface requirements, entity requirements*, and so forth. This subclassification scheme will help reduce the search space when working with specific kinds of artifacts.

In knowledge-based systems terminology, the ontology is a typical "is-a" heirarchy of terms from the domain. For example, "tank ***is-a*** platform". Frequently, reasoning needs to be performed on abstract object classes, such as a "platform" class, rather than on a specific object instance, such as a particular M1 tank..

Using the domain model and ontology, inconsistencies are much easier to identify. For example, it is a relatively easy matter to answer such queries as: "What objects are inconsistent if requirement R is deleted?".. This domain model can be stored in one central location and can be represented in a Resource Definition Format (RDF), which is a semantic network representing meta-knowledge about data sources at a given site. XML is one

method used to represent a RDF, since XML provides services for navigation and update (see Documentation Generation and Navigation).

Inconsistency analysis can be accomplished in one of three ways:

1. External impact analysis can be executed by the user when desired;
2. Attributes can be added to all objects to identify inconsistencies with a set of defined inconsistency relations;
3. A method can be added to all objects that allows them to perform self tests for consistency. The object can run a standard test (present in the Catalyst core code) when it is created, modified, or deleted. Catalyst servers can also capture deletion events of related objects so the objects can then determine if they have been rendered inconsistent by an external event.

To facilitate the automatic assurance of semantic consistency and completeness, a rule-based system, such as CLIPS, should be integrated into Catalyst. Simple consistency mechanisms can be implemented without a rule-based system, but as the domain model and complexity grows, a rule-based system would be more maintainable and efficient. Also, as the user updates the domain model, it also must be checked for consistency and completeness.

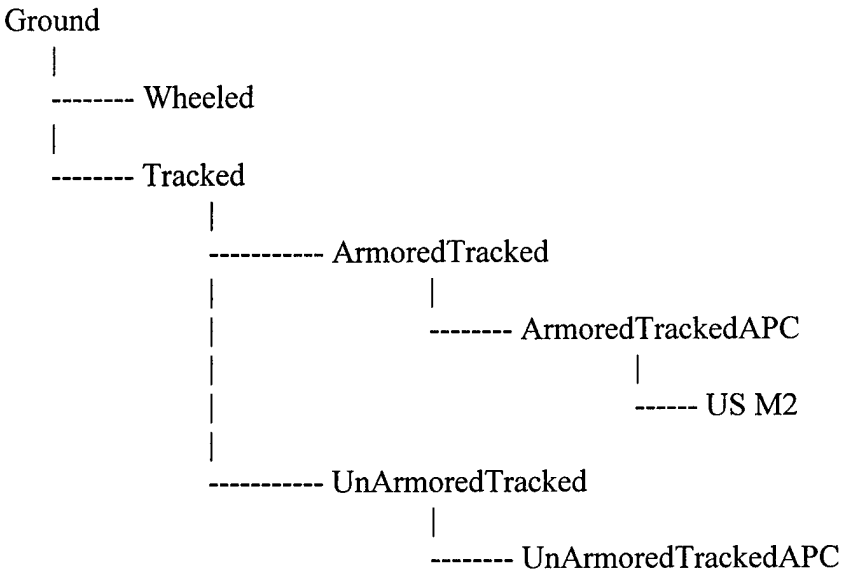
3.1.12 Automatic Linking of Data

The automatic linking of data is actually one facet of semantic consistency and completeness. The automatic linking of new objects to existing artifacts is similar to a repair situation when things have changed. The domain meta-model describes what should be linked to what (e.g. requirements should be linked to design artifacts in the following way: "requirement-object ***is-a-requirement-for*** design-object".) and the ontology/lexicon describes the common language for reasoning about the data. For example, in defining requirements, the user can designate that a requirement pertains to tanks (using the ontology/lexicon). When a new tank is added to the FOM, the systems knows that requirements are to be linked to FOM classes and it examines the current requirements and links the new tank class to any requirements containing the tank keyword. The system should present the linking options to users for verification. As a second step, natural language parsing techniques can be added so that the keywords can be extracted from the requirements without the user having to specify them. The latter is a future and longer-term effort.

3.1.13 FOM Generation Issues

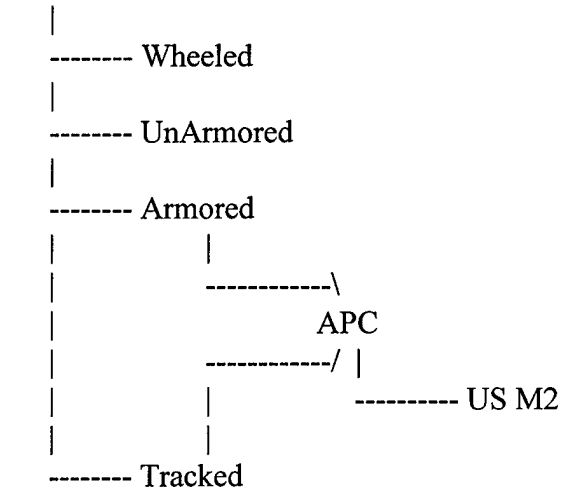
During the process of creating the FEDEP in Catalyst, the potential for automatically creating a FOM from the integrated data arose. For example, integrated data from a requirements tool or scenario generation tool could be used to automatically create a FOM in Catalyst. This FOM could then be accessed from other tools that use the OMT DIF format, such as the OMDT.

Unfortunately, automatic FOM generation has its issues. It can be especially difficult because of single inheritance, the only kind supported within the HLA. Single inheritance forces developers to create less efficient and flexible class hierarchies, such as:



Note the necessary redundancy of information under Tracked Ground vehicles.

A better, multiple inheritance object hierarchy would look as follows:



Using single inheritance can cause a combinatorial explosion, since there needs to be classes to represent each possible combination of class types. Alternatively, some of the abstract classes (such as "unarmored" or "armored") might be better "flattened" and represented as

attributes, rather than classes. The usual rule of thumb is if a class represents a different behavior that can be inherited, then it should be represented as a class; if it is simply a conceptual difference (armored vs. wheeled), then these might be better served as attributes. In any case, the HLA only supports single inheritance in its FOM, so the FOM developer is already limited in the development of class hierarchies.

FOM development includes identifying the interactions among objects that will occur during federation execution. Traditional scenario tools, such as the ModSAF PVD, provide no declarative information on object interactions; only classes can be derived from the scenario file. HLA specific scenario tools need to be developed that can elicit the interactions from the federation developer.

Determining FOM attributes is also difficult. In the ModSAF scenario case, vehicles have tasks, some of which are physical models and some of which are behaviors. Although physical models have useful attributes, some may be internal while others may be external, with no automated way to tell one type from the other. Without meta-knowledge about the system (in this case ModSAF) it is unclear what attributes should be represented within the FOM. Thus, most of the FOM comes from an *a priori* domain model. With the ModSAF scenario integration into Catalyst (see ModSAF PVD scenario under HLA tools), 90% of the generated FOM is accomplished independent of the actual scenario .The scenario only provides the leaf classes, such as vehicles; information about these vehicles (armored, tracked, etc.) determines where in the domain model they should be placed.

In addition, many abstract classes are not used in the FOM. These classes are defined as placeholders for future FOM. The unused classes could be removed from the FOM based on the classes used in the federation scenarios. This highlights another FOM creation issue: "What is the proper mix between domain and scenario-specific FOM classes and interactions?" Too much domain information can make the FOM large and unwieldy but too little makes the FOM less flexible for future federation use.

3.1.14 Documentation Generation and Navigation

A key problem in any large system development effort is the generation of accurate documentation. Since the objective of HIDE is to integrate multiple tools across different phases of the FEDEP process, and since each tool contributes to the evolving knowledge base of artifacts, the problem of generating documentation is compounded. Some tools may define similar objects, which must be coalesced into a single view, other tools may contribute new objects that must be related to others in the evolving knowledge base. In this integration framework, therefore, it is desirable to be able to view documentation and navigate the various semantic relationships between artifacts in an easy and well-understood (i.e. web browser) manner.

The Extensible Markup Language (XML) may provide the basis of a solution to this challenge. XML is the successor to HTML, the Hyper-Text Markup Language. XML allows schemas to be associated with documents, allows custom tags to index document parts, and

provides a rich set of linkages that may be used to relate document parts to each other in meaningful ways. To create current, consistent documentation of FEDEP artifacts, we intend to use XML to generate documents from the CORBA objects in the HIDE knowledge base. By generating documentation directly from the FEDEP knowledge base, we accomplish two key objectives in one approach: (1) the generation of current, user-friendly web documentation from the FEDEP artifacts and (2) a web-based browsing environment that allows users to access and navigate through FEDEP artifacts in a format that they understand. Just as the FEDEP provides a process-centric view of the enterprise, XML browsers can provide a document-centric view of the world. XML advantages include:

1. A mechanism for aggregating enterprise objects into custom views that can be navigated in a hyperlink fashion using standard XML parsers and browsers.

2. A Document Type Definition (DTD) that can be associated with the document. This DTD acts like a schema, providing the potential for schemas to be stored with the data and the documentation.

3. XML tags that allow different views of the same data. For example, a FOM class tag <OMTClass> might be used to display the class in a traditional form or a <OMTDIFclass> tab might be used to display the information in a DIF format. The use of XML style language (XSL) sheets also allow the documentation presentation format to be changed using a simple style language similar to that in use today for HTML.

4. A common presentation format (like HTML) for enterprise information, such as integrated data, web documents, and others, using plug-ins (Word, postscript, audio, video, etc.)—all in a web browser-like environment.

5. Robust XML links and custom tag attributes (to store the Catalyst unique ID, for example) can be used to represent all of the expressive power of Catalyst's relationships

Semantic links represent one of the most powerful features of XML and distinguish it from HTML. XML links can be *simple,* like HTML links, *extended,* to incorporate multiple link destinations within the same links, *groups,* to group links together, or *document.* Links contain many attributes that can be of use to Catalyst data:

ROLE: This represents the role of the link and can be used to capture Catalyst's semantic linkages.

TITLE: The title of link for navigational purposes.

CONTENT-TITLE: Specific information about the resource being linked to. The Catalyst instance name can be placed here.

CONTENT-ROLE: The role of the information being linked to. The Catalyst class type can be placed here.

SHOW: This attribute can hold three values: NEW, REPLACE, or EMBED. NEW follows the link in a new browser, REPLACE replaces the current page with the link resource, and EMBED embeds the linked resource *within* the current document. As mentioned later, EMBED can allow objects of different types to be embedded in the same document.

ACTIVATE: This attribute can hold two values: AUTO or USER. USER follows the link when the user clicks on the link. An AUTO value instructs the browser to follow the link as soon as it hits the link when parsing the document. In combination with the EMBED SHOW attribute, this can allow any kind of embedded resource to automatically be displayed.

BEHAVIOR: This attribute allows an implementation-dependent description of the link traversal behavior to be stored with the link. This can be used to provide tool display information such as that used by Catalyst annotation objects.

XML links also have flexibility in defining the destination of the linked resource. The destination is given using an URL but can be augmented using XPointers. XPointers allow the link to go to a specific location within a document instead of to the top. They allow the destination to be specified in terms of the ID of a specific tag (Absolute), by traversing the document tree (Relative), or by string matching. Relative terms allow you to specify the relationship (CHILD, ANCESTOR, PRECEDING, FOLLOWING, PSIBLING, FSIBLING, ROOT, DITTO, or STRING), the index (e.g. the 5th child of some parent) and the type of the element. A span can also be defined so only a portion of the document is displayed. For example, the XPointer (starting with the pound sign):

http://hideserver.com/tanks.xml#ROOT()M1..DITTO()FSIBLING(1,..)

specifies the destination of the link as starting from the root of the document, find the tag with ID equal to M1 and display starting there ending with the following tag of any type at the same level. The use of links and spans can be very useful for presenting custom views of the integrated data as the result of queries.

There are two alternatives for integrating XML with Catalyst:

The Catalyst core IDL can be modified to generate an XML version of each object. This is not desired because not only does it require changes to the core IDL but restricts the potential presentations of the object.

XML documents can be maintained on a server that contain all the objects of the same class type. The XML API allows individual elements to added, deleted, or changed, since the document is represented internally as a tree. The only disadvantage is the potential bottleneck as many objects of the same type are changed at the same time. Updates would be

deadlocked as the XML file is being updated. The advantage of this approach include easy querying of the data and existing documentation. This advantage has the following features:

- Ad-hoc queries can reference internal components within these documents.
- Using the capability of XML Xpointers, which can provide a span, we may selectively display only appropriate portions of a document In other words, document displays based on specific queries present only what was desired by the user.
- Using the auto traverse and embedding feature of links, objects of dissimilar type can all be aggregated into the same document as a result of queries.
- These XML documents can also be edited and transformed back into the Catalyst objects.
- XML templates can be set up for the various kinds of HLA artifacts (objectives, requirements, FOMs, etc.) so that users can enter their information without other tools.
- The XML browser can be set up to view the documents based upon the FEDEP process. A selection of Work Products can be presented to users, for example, allowing them to choose what they wants to view based on artifact types(requirements, design artifacts, etc.)

Although XML is not yet mature, it shows promise for combining the Catalyst browser with the HTML documentation and referenced artifacts. XML paves the way for automatic generation of HTML documents representing the FEDEP data in HIDE. A Java prototype using JACL (Java/TCL linkage) should be developed to explore the potential for document navigation.

3.1.15 FEDEP Process Enactment

The FEDEP model v1.1 was created and enacted using Catalyst's process definition and enactment tool. Available tools were integrated with this executable process model so that tools could be invoked from activities within the process.. For example, a requirements tool, called WinWin, is one tool that can be invoked by users from the "Conceptual Analysis" activity. In some cases, more than one tool is available to users to accomplish an activity. This provides them with multiple approaches to creating the same work product.

Roles and products were also defined that correspond to the activities in the FEDEP. Future activities include making the process tailorable to facilitate federation reuse and modifying the tool to support the concept of "optional work products".

Several issues concerning the process enactment tool and the FEDEP itself have arisen out of the process definition and enactment work:

Process Brittleness. The brittleness of the process is a concern. At the recent SIW conference, the conclusion among the FEDEP designers was that the current level of detail does not cover all the uses of the FEDEP so a 5-step abstract model was included as well.

This is not sufficient for enactment, but it does beg the question of how brittle a defined, executable process model will be.

Early Identification of Federates. In requirements development, it is possible to identify federates early in this phase. How is this done in the current FEDEP and what is the work product?

Ideal FOMs. In the Conceptual Model development phase, the concept of an "ideal FOM" in OMDT format was added to the enacted process. ModSAF was used as a proof-of-principle to generate an initial FOM of objects based upon a federation scenario. This is an opportunity for other scenario tools, such as Modus Operandi's tool, to collaborate and add their own information to the ideal FOM. The reuse of ideal FOMs also needs to be supported.

Alternative FOM Development Strategies. In federation design there are alternative FOM development strategies, such as using a bottoms-up approach, using a combination of the SOMs, using a previous FOM, or using a reference FOM. This may require alternative paths through an enacted process that reflect these alternative approaches; this multithreaded approach is not currently supported in the automated FEDEP process model developed under this UDO. The tool may need to be modified to support multithreaded process paths, or separate versions of the process could be developed. References to documentation discussing the FOM development process should also be accessible from the process. FOM development should also take into account changes to federates necessary to conform to the FOM; these changes may be captured as "issues" within a rationale capture tool, such as WinWin.

FEDEP Roles. Using the defined FEDEP roles and products the proper roles need to be used where applicable in the process. Currently there are no roles defined. One problem is that some of the roles are supervisory and not the actual "doers". Some products are also collaborative such as Requirements negotiation and thus are not assigned to any one individual. How can this be supported?

Federation Execution. The execution phase of the FEDEP raises some questions about its enactment within HIDE. Should HIDE launch the RTI, the federates, and data collection applications? Although this is defined as an "activity" within the FEDEP model, it appears to be quite different and much more complex in nature than other activities within the process. In particular, federates are usually distributed over various machines, require special initialization, and are often managed by controller applications. It is not clear that these variable and complex functions should be handled within HIDE. Secondly, should results of the federation execution be fed back into the HIDE knowledge base and linked to other data/objects? How the "Federation Execution" activity within the FEDEP fits into HIDE is a concern and must be addressed, since many HLA tools focus on the execution phase. Recommendations for execution phase tools will be withheld until decisions concerning the execution phase can be reached.

3.2 Integration of Current HLA Tools

The HLA tools developed by DMSO and others are the best candidates for integration into HIDE, since they are already support specific activities within the FEDEP. However, issues that must be considered include the platform of the tool, its robustness, its capability, availability and cost.

The platform issue is an important one. In order for HIDE to be successful, it needs to operate across heterogeneous platforms in as a homogeneous a manner as possible. This topic will be explored in depth in follow-on work. The conversion of the Catalyst toolset to Java is currently underway and will help with this problem.

The current state of the art in HLA tools was analyzed under this effort. The issues mentioned previously were taken into consideration for each tool and any potential tools for integration into HIDE were identified.

3.2.1 CAFDE

The Computer Aided Federation Development Environment (CAFDE) is an effort whose goals and objectives seem similar to those of HIDE, but whose approach is quite different. CAFDE's main goals are logical flow through the FEDEP, a standard GUI interface to all tools, internet access to repositories, coordination of multiple development sites, HLA compliance, feedback into process, and interoperability (tools, platforms, and languages). The development approach to CAFDE includes using a standard shell for connecting tools, relying on next generation tools to be developed using CAFDE-specific APIs, and the relying on DIFs formatting standards for interoperability among tools. CAFDE expects future tools to provide the automation needed to capture sponsor objectives, provide requirements traceability, and to support the development of object models that conform to sponsor objectives and needs. Some of the problems associated with this approach are discussed below.

The HIDE environment provides for similar functionality as CAFDE, but uses CORBA/Catalyst as the integration framework. There is no reliance on DIFs; traceability and consistency can be maintained via the Catalyst semantic relations manager capability. Collaboration can not only be provided by collaborative tools, such as WinWin, but by the distributed nature of CORBA itself, which provides implicit collaboration and coordination across separate development sites.

In the CAFDE approach, all tools are intended to have the same look-and-feel, because this makes training and integration easier. This is reasonable in theory but impractical in reality; in general, it requires starting from scratch in the development of what is essentially an object-based software development environment. HIDE can provide a consistent interface to the integrated FEDEP data via the Catalyst tools, but the tools themselves will have different interfaces. Requiring tools to conform to a single kind of interface is impractical unless you

develop all the tools yourself (e.g. Microsoft Office) and won't work for all the legacy tools. Unlike CAFDE, HIDE cannot rely on next generation tools to implemented, but rather provides an agile, CORBA-based framework that can take advantage of existing and developing tools through seamless integration of the underlying knowledge base. This is a fundamentally different approach based on open interoperability standards and high leverage integration of evolving technologies as they become available. The notion here is that many tools may be used to support a specific activity, with the appropriate integration of the tool into the HIDE framework; users are free to select what works best for them, whether that be using a familiar tool to do the job or using a number of tools that each contribute a part of the solution. Were a set of tools to be developed in CAFDE, HIDE would treat them as alternative tools that could be used in different phases of the FEDEP process to produce and use FEDEP data, like any other tools.

CAFDE is built on the notion of a Federation Engineering Framework (FEF). The FEF is an abstraction of the FEDEP process (or any development process, for that matter) and gives CAFDE its commercial bent needed since it is a SBIR. The FEF consists of 3 steps: requirements, construction, and transition. The FEF is not practical for process enactment but the FEDEP can be mapped to the FEF.

CAFDE plans to use "Use-Cases" to represent federation requirements. This is a promising notion, provided tools exists that allow developers to create Use-Cases (see Together/J Commercial Edition). In HIDE, use-cases would be linked to requirements and other related artifacts. CAFDE will allow the steps of the FEF to be performed incrementally and iteratively; this is one required extension to the process enactment tool (as previously discussed).

CAFDE also supports the use of automated tools to perform computer-aided compliance checking of the simulation components against the use-cases defined. This has not actually been done, but APIs exist that would allow the creation of semantic consistency tools.

CAFDE-based tools (that have yet to be created) will document the construction of the simulation components in the OMT tables. Using the OMDT, these tables may be printed. In HIDE, another simple Java tool may be desirable that can print these and other FEDEP data from Catalyst.

CAFDE "embraces" the use of the OMT Base Object Models (BOMs) for reuse. These can be considered portions of FOMs that identify a "single aspect of federation interplay". These BOMs can be represented as a palette of components in a CAFDE tool. This seems to a be a function of the tool itself, like OMDT, rather than one that is a function of the HIDE integration environment. Under HIDE perhaps potential BOMs could be identified from requirements or scenario information and brought into Catalyst. CAFDE can also use the requirement information to generate the meta-data for the newly created BOM, but only if the requirements were generated by a CAFDE tool. HIDE does not restrict tools to a single format; any may be used and integrated into Catalyst.

CAFDE plans to use the RTI itself as the mechanism for distributing information. While this sounds like a promising synergy, "using the HLA to develop an HLA federation!", the RTI was not designed for this task. More robust and proven distributed information technologies, such as the Web and CORBA, are better suited to this task.

CAFDE plans to develop a wizard control to guide users through federation development. This is a good idea, and should be incorporated into HIDE. CAFDE has a scripting language for creating Wizard Controls. HIDE could do something similar with some Java classes provided.

CAFDE relies on DIFs for integration of information (OMT DIFs, FED file DIFs, UML DIFs, etc.) This is a danger for a number of important reasons. (1) DIF is yet another data interchange format. As such, it lends itself to a proliferation of interchange formats. (2) DIF is not a universal standard. This means that other (COTS) tools being developed cannot be easily integrated with DIF-based tools. This is an unfortunate consequence of using yet another "ad hoc" interchange format. (3) There are many different DIF formats; it is unstable and changing. Tools using the DIF "formats" must deal with this instability. (4) Support tools for DIF are virtually non-existent. For example, users cannot "browse" DIF files, perform impact analysis on changes to objects within DIF files, export DIF file data easily to other tools, or analyze integrated DIF data using simple scripting languages, rule bases or other methods. HIDE's approach of using CORBA, a widely accepted standard for common distributed object representation and distribution, offers a more scaleable approach.

The CAFDE API is not discussed in detail. The future goal of this API is to provide OML interfaces, access to the web, and the RTI collaboration interfaces. No other detail is provided. With the API approach, each tool will be integrated differently, using different interfaces and functions from the API. In Catalyst, each tool is effectively integrated the same way - by creating a server that implements the small number of core IDL interfaces. This is a much cleaner approach to tool/data integration.

A few questions should be answered with regard to the API:

1. Does the API support distribution? That is, can the tools be running on a different machine and still exchange data with CAFDE?
2. When data changes, is it up to the user or to CAFDE to propagate those changes to other tools within the CAFDE toolset? For example, if a requirement changes, will all affected design artifacts be "notified" of the changes and appropriately "updated" to maintain semantic consistency across the DIF files?
3. If the API is responsible for change management, then CAFDE itself would have to be modified as new tools are added so that it knows what needs to be updated. This is not a very evolvable approach.

In summary, the CAFDE framework has some interesting ideas that should be incorporated into any "HLA environment" approach. However, the design of CAFDE does not go far enough to provide a truly agile and evolvable integration framework. CAFDE has been

envisioned as a comprehensive environment, but with the contractor providing most of the tools; true integration has not been seriously considered. The detail in the CAFDE paper suggests a lack of integration experience and the problems associated with it. For example, client-server technology is lacking (2 tier or 3 tier), a widely accepted technology for tool and data integration. Thus, there is no common data/knowledge representation and no integrated view of the data via semantic relationships (thus no impact analysis can be performed). Integration is done entirely through various DIF formats and APIs similar to the loose integration of Catalyst. In conclusion, HIDE is farther along as an integration framework, since the Catalyst infrastructure has been developed over years and has already dealt with some of the hard issues of tool integration.

3.2.2 Federation Management Tool (FMT)

This tool is written in Java and records and displays the MOM data about the federation and federations. There was no mention of support for the pausing/starting of federations. It does support extra capabilities by extending the MOM data; it is important to note that federates must be modified to supply this data. Since the FMT fits under the execution phase of the FEDEP, and since it is unclear how the execution phase fits into HIDE, it is also unclear how to integrate the FMT (Note: There similar tools are being developed by MITRE/MaK technologies and Aegis).

3.2.3 Federation Data Collection Tool

This tool is being developed by Virtual Technologies Corporation (VTC) and fits in the execution phase of the FEDEP. Nothing further is known about this tool at this time.

3.2.4 TASCVision

TASCVision is not an HLA-specific tool but is a simulation visualization tool. TASCVision was investigated for possible integration into HIDE and it was determined that it would not be of use to be *directly* integrated. The data it produces is low level, consisting of information on light sources, polygons, etc. instead of domain objects such as vehicles. However, it can be linked to a scenario tool to give a nice visualization environment and the scenario tool can be integrated. TASC is in the process of integrating TASCVision with its scenario tools and once completed, TASCVision will be indirectly integrated when the scenario tool is integrated.

3.2.5 TASC OMDT

The TASC OMDT was loosely integrated into HIDE as mentioned previously. To be fully useful, a newer version that supports command line arguments for file access and access to

the OMDD is needed. TASC did create a version with command line argument support but it never worked properly.

The OMDT was loosely integrated using a OMT DIF file translator. Any tool that uses the DIF format can now be integrated into HIDE. The loose integration scripts also keep the data consistent between HIDE and the OMDT. When the tool is launched from HIDE, HIDE FOM data is converted into a DIF file read into the OMDT and when the OMDT is exited, the DIF file is converted back into HIDE reusing CORBA objects that did not change. Work needs to be done to remove the hard-coding of the files and to allow the specification of the FOM for HIDE to use. This integration, however, demonstrates a proof-of-principle.

3.2.6 Visual OMT

This is another OMDT tool developed by PITCH that is currently in the test and evaluation phase. It provides all the expected OMDT capabilities. It is of limited use to HIDE at present because it is a COTS tool and only runs on Windows NT/95.

3.2.7 OMDT Pro (HLA Lab Works)

This is the Aegis version of the OMDT. The latest version supports access to the OMDD. It allows the user to import one or more data dictionary files and then pick from the contents of the data dictionary files when populating or modifying an object model. The user interface looks much cleaner than the TASC OMDT. Unfortunately, OMDT is a Windows 95 tool, but a Java version is forthcoming sometime in the future. It interesting to note that OMDT Pro is an OLE server which may provide some integration flexibility through the use of some OLE/CORBA bridges. A Sun and SGI version is due out the 2nd Qtr of 1998. This version should be integrated into HIDE when available, since it has more capabilities than the TASC OMDT that is currently integrated into HIDE.

3.2.8 FedProxy (HLA Lab Works)

The Aegis FedProxy is a federate that acts like a tool. It can be used to prototype early design concepts and test them in a realistic environment. It can also model incomplete or external functionality. As part of federate development it can aid in compliance testing by serving as an auxiliary federate to test with the federate under test. As part of the FEDEP, the FedProxy can be used during federation design to serve as a stand-in for real federates and can generate realistic data traffic to study network traffic under various hardware, software and network configurations. FedProxy's event queue and clock can be used to schedule events. Also, the user can watch external events affect FedProxy's objects and view record logs of all subscribed activity. During federation integration and testing, the FedProxy could stand-in for missing federates. There is some overlap here with the Federation Test Suite that will have to be investigated.

The FedProxy is FOM/SOM independent, has a way to represent a federate's published and subscribed entities, represents the state and *behavior* (with user intervention) of proxy entities, and supports all the HLA time management schemes. It has a set of Java classes that allow users to add automated responses to external events or to create any custom behavior desired.

FedProxy is being developed under a SBIR and it is currently planned for a release for Sun and SGI in the 2nd Qtr 1998. Again, since this is an execution tool, integration depends upon the federation execution/HIDE issue. From an integration standpoint, it could use the OMT object model data in HIDE but it may not contribute any data back to HIDE. Integration of this type of tool may not make much sense and it needs further exploration.

3.2.9 FedDirector (HLA Lab Works)

The FedDirector is a federation management tool that ties into the federation management RTI services and subscribes to the MOM and FOM. FedDirector subscribes to everything in the FOM so it can display various views centered around the HLA functional areas. FedDirector has a main view that displays information on all federates such as status and time. There is a Federation Management view that supports the pausing, restarting, saving, restoring, etc. of federations. The Declaration Management view provides publish and subscribe information on a federate-by-federate basis and allows modification of these interests on behalf of the federates (RTI 1.3). The Object Management view displays instance information of all objects by federate with values, delivery order, and transport type information. The Ownership Management view describes what objects and attributes are owned by which federates and allow changes given a list of possible owning federates. The Time Management view displays each federates time, lookahead, and whether it is time constrained or time regulated. Finally, the tool can also send and receive interactions to/from any federate. This tool is only available for Windows NT in March 1998 (beta) and will be available for use with the RTI 1.3 in June (version 1.0). It seems to be more robust than any of the other management tools but its platform will be an issue.

3.2.10 Conceptual Modeler (HLA Lab Works)

This tool constructs and records the static and dynamic portions of the conceptual model using an object-oriented approach. It uses UML as its symbology. It will import entities, actions, tasks, and interactions from the CMMS and convert it to the UML representation. The UML can also be annotated with information such as requirements and constraints to support VV&A. Other annotations, such as specific simulations for modeling particular objects, security requirements, or scheduling, can be captured and attached to the specifications. This tool is being developed under a SBIR and is believed to be in the very early stages, but should be integrated into HIDE when available.

3.2.11 Federation Composer (HLA Lab Works)

This tool records federation composition, performs comparisons among object models (conceptual model to SOM, conceptual model to SOM, SOM to FOM, etc.) The similarity comparison is done in the same way as the arbitration tool and relies on the HLA Data Dictionary to aide in the comparison. The comparison is used to find candidate SOMs for a federation. This tool is exactly what the arbitration tool is supposed to be, although from discussions with Aegis it does not have the complexity in terms of its comparison heuristics. Since this tool is a year from deployment, work on the arbitration tool may need to be continued. However, Federation Composer should be integrated into HIDE when available.

3.2.12 Scenario Generation Tool (SGT) (HLA Lab Works)

This is a 2D/3D tool that uses the CMMS to drive the definition of the scenario laydown. Its primary purpose is to map the conceptual object representations to the FOM representation and assign responsibility of each scenario object instance to a federation member capable of modeling the scenario instance. Since some of this mapping is what HIDE is supposed to do and it will be interesting to analyze the overlap.

The SGT will follow these steps:

1. Capture the identification of theater, geographical specification, scenario time frame and study resolution.
2. Determine scenario laydown classifications (land, sea, air), resolution (aggregation level), terrain models, and coordinate systems.
3. Define assets and initial locations.
4. Determine associated components of assets (radars, weapons, sensors, etc.).
5. Define the scenario timeline and events that will occur.
6. Determine routes of assets and times.
7. Determine command and control relationships.
8. Define areas of interest, such as flight corridors, engagement areas, etc.

The SGT plans to interoperate with other tools using DIF formats and SQL interfaces. This tool will be available within about three months. Other scenario tools will be integrated into HIDE, as should this tool. The overlap with other scenario tools needs to be explored. If Aegis can be part of the HIDE team, then perhaps the proof-of-principle version can be integrated.

3.2.13 Scenario Execution Planning Tool (SEPT) (HLA Lab Works)

The primary responsibility of this tool is to bring together the federation composition, the FOM, and the scenario laydown, each from a separate planning tool, and plan how the

scenario will be executed within the identified federated environment. Its requirements include:

1. Associating the conceptual models used during scenario laydown with the objects defined in the selected FOM. Since associations are made at an attribute level, this can be a difficult task. Under HIDE, some or all of this may be able to be done automatically (see Automatic Linking of Data).
2. Identifying the FOM object publishing responsibilities of each federate.
3. Assign ownership responsibility of each object instance.
4. Develop a translator file for federate initialization.

This kind of tool would be of use in the FEDEP of HIDE.

3.2.14 Scenario Monitor (HLA Lab Works)

The Scenario Monitor acts like a PVD for the federation execution and also serves as a logger for scenario playback.

3.2.15 Federation Test Suite (FTS)

While it is still unclear how federation execution will operate under HIDE, the data used by the FTS can still be integrated for linkages to other products. The FTS scripts were developed by consulting various HLA resources such as the FOM, Fedex Performance Workbook, Interaction protocols, and federation agreements. The test procedures are written in an HTML-like (looks like XML) format that should pose no problems for parsing. Test procedures contains tags for initial conditions, requirements, capabilities, scenario information, pass-fail criteria, and script filenames or instructions for the test and analysis federates. The only problem is in linking the test procedures to the HLA artifacts in HIDE, given the text in the appropriate section of the test procedure. The text in the test procedure would have to be written conforming to certain rules so the appropriate artifacts could be found and linked within HIDE, otherwise these would have to be done manually. This is another example of the automatic linking problem mentioned earlier.

It is interesting to note that while the FTS and other testing tools are referred to as "federation testing tools," they really fit more in the federate development process, not the FEDEP process. To the extent they are federation testing tools seems to be limited to the fact that the RTI calls given to the federates under test involve data from the FOM as opposed to the SOM. How to best integrate testing tools needs to be investigated.

3.2.16 OSim

OSim has promise for integration as part of the *Federate* development process. It is a complete simulation framework that generates federate code. It stores its classes and instances in Object Store, which is also used as Catalyst's persistent object archive, and thus can be tightly integrated through its own APIs. OriginalSim, Inc. also plans to integrate OSim with the Rational Rose CASE tool. OSim can also can store its data in text files, which can be parsed, but this is inefficient since the data is quite complicated. OSim needs to be explored further and integrated into the federate development process.

3.2.17 ModOIS

The CDF Upgrade delivery order has been modifying Motorola's ModOIS for exercise setup (specifically for simulators). Motorola has developed an HLA RPR FOM version that may play a role in HIDE, once the execution phase issue has been addressed.

3.2.18 ModSAF scenarios

ModSAF is not an HLA tool but was integrated with Catalyst to provide a proof-of-principle scenario generation capability for HIDE. The ModSAF PVD is not really meant for this task (nor is it removed from ModSAF, so the entire simulation must be run to use it) but it illustrates how some initial FOM objects can be derived from scenario files. The ModSAF rearchitecture task is currently developing a stand-alone scenario editor that the HIDE team may be able to employ. In any case the problems mentioned under FOM generation still exist.

3.2.19 RTIME

RTIME is a graphical CASE tool that has been augmented to support SOM annotations to the Shlaer-Mellor Object-Oriented Analysis Methodology. It can generate RTI code for federates and exports the SOM in the DIF format. This tool really has no place in the FEDEP process but could be used as part of a federate development process.

3.2.20 Warfighting Analysis and Integration Center Java PVD

Booz-Allen & Hamilton, Inc. (BAH)has developed an initial version of a web-based PVD that can be used for simulations. Special server processes tap into the DIS network (they use VR-Link) and format this information for display on the web via a Java applet. Portions of the server processes are written in Java as well. HLA support is planned for the future. There is really nothing to integrate for FEDEP activities. However, during federation execution it might be useful to launch this from HIDE to watch the exercise. Since BAH is an ADST-II team member, there may be a possibility of using their tool as HIDE matures.

3.2.21 Federation Execution Planner's Workbook Tool

This tool is being developed by VTC and proposes to automate the creation of the Federation Execution Planner's Workbook. It is unclear why a tool is needed to fill out a workbook, but perhaps it will use artifacts produced by other HLA tools through DIF. Once more information becomes available, this tool can more be more properly analyzed.

3.2.22 Federation Verification Tool

This tool is being developed by GTRI and will most likely not be available for alpha test until August or September. This tool will use yet another DIF format. No further details are known at this time.

3.2.23 XGen

XGen is a ModSAF scenario generation tool designed to create scenarios and initialize vehicles for a ModSAF-only exercise (with possibly more than one ModSAF). It uses the CATT task database and TOE data to do this. XGen can fit into the FEDEP in the same manner as the ModSAF PVD proof-of-principle. Since it creates ModSAF scenario files that can already be translated into HIDE, it is in effect already integrated, albeit loosely. The modifications it makes to ModSAF vehicle reader files are part of federate initialization; how this fits into the FEDEP needs to be explored.

3.2.24 DEM

TASC's DEM is an execution monitor that monitors network-specific properties in addition to access to the MOM data. It also initializes federates, the communications network, and the RTI. The actual execution of the federation from the HIDE perspective is nebulous, as mentioned earlier, so DEM may or may not have a place within HIDE.

3.2.25 TASC HLA Construction Kit

The HLA Construction Kit is a TASC IR&D project that plans to integrate primarily OMDT, DEM, and XGen in some fashion. The HLA construction toolkit also plans to provide support for entering exercise objectives using keywords and keyboard input. It also plans to provide a data mining engine that can retrieve and display from the appropriate authoritative database(s) a scenario laydown and object selections in textual and visual formats. When this capability is available, it may provide functionality for the beginning phases of the FEDEP process. This is especially important, since most of the tools that can be used in these phases are Windows-specific and thus cannot be run in the UNIX environment. Communication

needs to be kept with TASC to keep abreast of their progress. This tool may possibly be integrated into HIDE.

3.2.26 Modus Operandi Scenario Generation Tool

The Modus Operandi Scenario Generation tool allows users to capture descriptions of federation functionality and use. Scenario Generation provides a graphical means, called an outliner, for a user to enter a scenario. The outliner is a visualization of the underlying schema used to capture scenario information. The underlying schema forms a template for the kinds of information that are required to generate a complete, valid scenario. Scenarios are driven by goals or requirements. The goal describes the intended outcome of the scenario. This goal-driven approach ensures that a user is able to articulate what he/she is trying to accomplish in a given scenario.

A scenario comprises a context and a narrative. The context includes the scenario name and type, any background information, any related goals, and any related scenarios. The scenario narrative includes the steps which are performed to achieve the scenario goals. Each scenario step includes the role of the entity performing the step, the actor who performs the step, the step inputs, the step outputs, any obstacles to performing the step (exceptional conditions), any optional extensions associated with a time reference that specify step sequence (perform before or perform after), and attributes such as postconditions (completed/not completed).

The entire structure is recursive, allowing steps to have subgoals and substeps. Quality attributes, such as cost, performance, schedule, stability, availability, etc., can be associated with a narrative to allow trade-off analyses during scenario generation. The actors identified in scenario steps will be linked to model entities in the simulation federates. Using this approach, a scenario can be used to select the appropriate model entities for a federation.

While not a graphical laydown tool, this tool does allow the user to specify the kinds of information necessary for a federation. It is expected that this tool will be integrated into Catalyst by Modus Operandi themselves, but if it is not, it should definitely be integrated into HIDE. It may also collaborate with other scenario tools that are not specific to the HLA to provide 2D/3D visualization.

3.3 Integration of Software Engineering Tools

Most of the software engineering tools investigated are part of the DARPA Evolutionary Design of Complex Software (EDCS) program, since these tools are more easily available and the HIDE team is involved in other EDCS-related efforts. Unfortunately, most of the tools apply to the federate development process and do not really apply at the federation level. Most are concerned with problems at the code level. These tools should be investigated more closely when the federate development process is investigated.

3.3.1 Chimera

Chimera is a prototype for a software development environment based upon the idea of hyperlinks. It is similar to Catalyst in the use of relations between data but does contain any common format and is only a 2-tier client server model (not as scaleable). The basic idea is that all the tools that work on the data support the use of hyperlinks so as links are followed, tools are automatically brought up with the appropriate document and the user is placed at the proper position within the document. Actions can be taken upon link traversal, determined as a function of who clicked on the link or how the traversal was requested. Chimera supports composable links, n-ary links, and multiple context links. The links themselves are objects as well, which is a useful feature. The disadvantage is that all tools have to be modified to support the Chimera API and hyperlinks within their native formats (there are exceptions, such as Framemaker, which has macro support and supports hyperlinks of its own). This is impractical for third party tools, such as those from Microsoft and Adobe. There is really nothing to be gained by trying to work with Chimera.

3.3.2 OzWeb

OzWeb is similar to Chimera in that its foundation is hyperlinks. It stores all artifacts and corresponding links in an object-oriented referential hyperbase. It also supports associative and navigational queries over the hyperbase objects.

OzWeb also contains some kind of process enactment. Depending upon process definition, OzWeb enforces task prerequisites and implications, including constraints on when artifacts can be viewed or updated; automates the invocation of tasks at the appropriate time; notifies appropriate supervisors under specified conditions; and collects metrics and maintains a complete audit trail of user actions. Using the same process information, OzWeb can automatically infer some linkages based upon inputs of one task being the output of another. HIDE will do this using a domain model, but the process objects could also be traversed to give other forms of linkages as well.

Like Catalyst, OzWeb supports the remote launching of tools and redirection of X Windows and Windows NT/95 GUI's to the user's screen. More specifics on how OzWeb redirects Windows NT/95 GUIs are needed, since this has a direct bearing on architecting HIDE for a multi-platform environment. Peer tool servers communicate with each other across a WAN to determine the best place to launch a tool, and automatically set up the local environment and invoke the tool on the user's machine. This most likely means that OzWeb has copies of the tool on various machines and uses load balancing to determine which one to run. Currently in Catalyst, tools can be run remotely by using a "rsh" (remote shell) command.

OzWeb provides Java, HTML, and X window client interfaces. The server can run on Solaris or Windows NT/95. The major components, notably the object management system, the process engine, the transaction manager, and the tool service can be used separately and their APIs accessed via direct links in code, TCP/IP sockets, HTTP or CORBA. It is

available for free download from www.psl.cs.columbia.edu/software/download.html. OzWeb does overlap Catalyst capabilities, but it should be explored further because some of their components, if robust, could be reused in HIDE.

3.3.3 WinWin

WinWin is a requirements/rational capture, negotiation, and coordination tool. It supports:

- capturing the desires of stakeholders;
- organizing the terminology so that stakeholders are using the same terms in the same way;
- expressing disagreements or issues needing resolution;
- offering options as potential solutions;
- negotiating agreements which resolve the issues;
- using third party tools to enlighten or resolve issues;
- producing a requirements document that summarizes the current state of the proposed system;
- creating documents that support multimedia and hyperlinks;
- tracing the ways by which requirements decisions were reached;
- checking the completeness and consistency of requirements.

To accomplish these capabilities, WinWin supports a set of artifacts including *win conditions* (requirements and rationale), *issues*, *options*, *agreements*, and *terms*. It also supports comments pertaining to these artifacts and the association of artifact files/tools with the WinWin artifacts. WinWin also supports relations among the artifacts. It has *relatesto* and *replaces* relations and links from issues to win conditions, options to issues, agreements to options, and agreements to win conditions. These relations are mirrored in Catalyst, allowing browsing of requirements by their relationships within Catalyst as well as within WinWin.

WinWin can also print out the artifacts in a text file (for loose integration), a HTML file or Framemaker file. WinWin also allows level 1 integration of third party tools (currently COCOMO and XEmacs are supported) which can be launched from the WinWin menu.

Currently, a loose integration for WinWin exists. However, WinWin should be tightly integrated into HIDE, since it has a client server architecture. A separate database server is run to coordinate among the various stakeholders. A tight integration would provide for immediate updates to Catalyst and impact analysis could be run when changes are detected.

WinWin should be used wherever negotiation exists. Currently, this is during the Conceptual Analysis and FOM Development activities within the FEDEP process. For FOM development, requirements capture and linkage is critical, since federation-wide decisions must be made concerning the object model, algorithms used, databases used, etc. One of the lessons learned from implementing the FEDEP is that frequently decisions remain undocumented.

3.3.4 EMMA

The Evolution-Memory Management Assistant (EMMA) is a collaborative development and evolution system developed by CoGenTex, Inc. under the EDCS program. It supports collaboration and evolution in the following ways:

- EMMA supports structured, goal-directed communication about the development of a system and can record information about the expectations and responsibilities of all parties involved. This is defined as the context of the development in which decisions are made (i.e. rationale). A system is decomposed into a series of goals, each with its own subgoals and context, and so on down to the primitive level. Plans for achieving those goals and corresponding assumptions are recorded.

- EMMA presents this information as a solution status in terms of uncompleted goals or inconsistent goals resulting from changes to requirements or assumptions. All goals will have interrelationships that can be traversed to anticipate the impacts of evolutionary changes.

- EMMA supports system evolution by recording anticipations about future changes in requirements, assumptions, and resources. A plan for responding to these changes can be recorded as well. EMMA also provides a mechanism for the high level goals (requirements) and assumptions to be changed as a result of changes to part of the system (architecture, COTS replacement, interface changes, etc.). This is known as top-down evolution. For example, when a system is placed into a new environment, assumptions under which the system was developed may no longer hold. Traces from the assumption in the top level context can be traced to all the lower level goals that depended upon the assumption. Given plans for likely evolution, EMMA develops anticipated responses to possible evolutionary changes. EMMA also allows changes as a result of problems encountered fulfilling lower level goals (bottom-up evolution). Finally, EMMA supports the impact analysis of changes and distribution of the results to interested stakeholders.

EMMA provides a Java browser that allows the user to navigate the knowledge base. Elements in the knowledge such as goals, plans, etc. can be linked to source documentation, much like the "annotation object" in Catalyst.

EMMA has several features in common with WinWin, namely support for collaboration (although no mention of any central database server was mentioned), recording of goals and assumptions, and recording possible directions for system evolution. However, EMMA is more suited for goal-directed development, after the initial requirements have stabilized, and is useful for planning future evolution. EMMA can complement WinWin and could potentially play a role in the later phases of the FEDEP, such as federation planning and future federation evolution. WinWin can provide requirements, assumptions, and evolution directives to EMMA and EMMA can provide back to WinWin exceptions and

requirement/assumption changes that have associated issues and/or require more negotiation. The use of EMMA should be explored to see how well it fits federation development.

3.3.5 Sybil

Sybil is a set of tools and browsers developed by the University of Colorado under the EDCS program to support the integration of heterogeneous databases. Its focus is the gradual migration of data from legacy databases to more modern ones and/or the interoperability among them. For example, as an application adds a new database, relationships among existing databases and the new database are built gradually over time. New data is placed into this new database and interconnections are built for keeping this data consistent with the legacy databases.

It is important to note that the legacy data is still accessed through the legacy applications (or through the modern application via the interconnections), there is no centralized view like the common objects of Catalyst. However, since there is no centralized view, the integrations are reduced to a set of point-to-point transformations which is not as scaleable as a centralized, common format. Every time a new database is added to the system, schema translations, data translations, and query translations need to be performed, as in Catalyst. However, interconnections to all the other appropriate databases must also be developed, which is where scaleability and adaptability becomes an issue. Based upon the information provided by the developers, there does not seem to be anything in Sybil that Catalyst cannot do or is not already doing.

3.3.6 Together/J

Together/J is Java-based tool for enterprise-wide software development, a product of Object International Corporation. It uses the Unified Modeling Language (UML) as the object paradigm and features simultaneous design-and-code editing, supporting either design first, code first, or both. The Whiteboard version is free. The commercial version also adds visual UML modeling, wall chart printing, automatic generation of HTML documentation, saving of class diagrams, use-case diagrams, sequence diagrams, and state diagrams.

This tool may be of use for federate development. The use of UML for specifying federation-specific elements, such as federation design and scenarios, should be investigated. If it proves worthwhile, this tool can be used during federation development.

3.4 Implementation of HLA Arbitration Tool

During FOM development, an arbitration tool can be used to compare federate SOM representations to identify similarities that can be used to arbitrate a common format required for the FOM. For new objects to be integrated into an existing FOM, the tool can examine the existing FOM and determine where the new object best fits. As FOMs become more object-

oriented this will be of more use. The Aegis FedComposer has this same goal. However, the heuristics behind the comparison are complex and the information needed may not be represented in the FOMs themselves. Also, once the FOM is decided upon and given a standard lexicon and a set of standard types, transformations can also be derived to transform federate SOMs into the FOM.

3.4.1 Comparing SOMs with other SOMs and FOMs

Using the OML, the user can query for existing SOMs and FOMs. However, there is currently no way for the user to determine which FOM best matches the SOMs that the federation is using. Also, once a base FOM is chosen, any SOM objects that do not exist in the FOM need to be added. Heuristics need to inform the user the best place for the object in the hierarchy (what it is most related to) and what the format needs to be.

The arbitration will be aided by a hierarchy of content-standard attributes, such as the OMDD, which will help determine the similarity among object models and will also represent the objects that will hold the transformation routines. The content standards also serve to filter what FOM capabilities are required so that some SOM information can be ignored. The impact of having no content standards or attribute hierarchy in the HLA is that more guidance is required from the user and there is less reuse of existing transformations.

3.4.1.1 Object Correlation Metrics

HLA objects are represented in terms of more general objects (inheritance) and in terms of aggregation of objects (components) and atomic attributes. This means the comparison of object models must be done in terms of their inheritance hierarchies and the composition of an object itself, which includes components and attributes. When trying to compare and correlate objects, several metrics can be used to determine how similar they are. These include the WHERE-IS, HAS-A, IS-A, SIBLING-OF metrics.

WHERE-IS Metric

A source object can be found in the FOM at a lower or higher level of decomposition than it is in its own SOM. This can be defined as the WHERE-IS metric and can be illustrated as follows:

SOM Object A	FOM Object B
Object Attribute A	Object Attribute D
Atomic Attribute B	*Object Attribute A*
Atomic Attribute C	Atomic Attribute B
Object Attribute D	Atomic Attribute C

The attribute A, which happens to be the class we are looking to compare, is a component of Object A but in the FOM, represented by Object B, it is a component of Object Attribute D which is a component of the main object. Object A and Object B are clearly similar in this case but are not exact. Comparing from Object B to Object A gives us the inverse metric. The WHERE-IS metric can be defined algorithmically as:

```
FOR I = 1 TO LEVEL_DIFFERENCE
        closeness = closeness - WHERE_IS_ADJUSTMENT*closeness
```

For each level of decomposition difference, the current closeness is reduced by the WHERE_IS_ADJUSTMENT percentage amount.

IS-A Metric

A source object can be related to a more general or more specific object present in the destination FOM object we are comparing against. This is defined as the IS-A metric. For example:

SOM Object A	FOM Object B
Tank	Tank
Main Gun	*M256*
Smoke Launcher	Smoke Launcher

In this case, Tank B specifies an M109, where Tank A specifies a more general Main Gun. This illustrates both the general-to-specific and specific-to-general IS-A metrics depending upon the direction of the comparison. The IS-A metric can be specified algorithmically as:

```
FOR I = 1 TO INFERENCE_DISTANCE
        closeness = closeness - IS_A_ADJUSTMENT*closeness
```

For each level of inheritance difference, the current closeness is reduced by the IS_A_ADJUSTMENT percentage amount.

SIBLING-OF Metric

A source object can be related to a similar object of the destination FOM via a common parent/ancestor. This is defined as the SIBLING-OF metric. For example:

SOM Object A	FOM Object B
Tank Platoon	Tank Platoon
M1	*M1A1*

Here, Object A and Object B are similar, since they are composed of similar subobjects (M1 and M1A1) that share a common parent in the domain (Tank). The SIBLING-OF metric can be represented algorithmically as:

```
FOR each sibling
        closeness = (WHERE-IS metric of sibling) *
                        Number_Of_Parents_In_Common / Number_Of_Parents
        if (closeness > max)
                max = closeness
closeness = max - SIBLING_OF_ADJUSTMENT*max
```

The SIBLING-OF metric tries to correlate a similar (sibling) object in the destination object with a source object. The closeness is adjusted based upon where the sibling object is found in the destination FOM and the number of parents shared by the source object and the sibling object. The best metric of all the siblings is used as the final metric result.

HAS-A Metric

A source object can be decomposed into its sub-objects, which can then be correlated. This is defined as the HAS-A metric. For example:

SOM Object A	FOM Object B
Object Attribute A	Atomic Attribute B
Atomic Attribute B	Atomic Attribute C
Atomic Attribute C	

Object Attribute A is composed of attributes B and C in the SOM; Object Attribute A does not exist in the FOM but the subobjects do. Objects A and B are clearly related. The HAS-A metric is represented algorithmically as:

```
closeness = 1.0 - HAS_A_ADJUSTMENT
contribution_percentage = 1.0/ Number_of_Subobjects
closeness_sum = 0
FOR each subobject
        closeness_sum = closeness_sum +
                                max_for_subobject(IS-A,WHERE_IS,SIBLING_OF)
closeness = closeness * closeness_sum
```

The HAS-A metric tries to correlate the source object's sub-objects in the destination object. The sub-objects are subjected to the previous metrics. The maximum for each of these metrics is used for each sub-object which are then combined together. The sum is then reduced in closeness by the HAS_A_ADJUSTMENT amount. Notice that in this implementation, the HAS_A_ADJUSTMENT is actually applied first, which the other metrics use as their initial value.

It is important to note that any combination of these metrics can be used at the various levels of decomposition and inheritance hierarchy to determine the closeness. Extra subobjects may also be present on either the source object or the destination object. Extra subobjects on the destination object do not affect the closeness as it has been defined. Extra subobjects only means that the destination object represents more than is necessary for the source object which is acceptable. However, there may be some ambiguity if more than one destination object share the same subset of subobjects that match the source object. As far as the closeness is concerned the objects are equal. A modification to the algorithm could be made that would choose the object with the least amount of extra subobjects but that is no guarantee that objects will not be ambiguous. Extra subobjects on the source object do decrease the closeness since the destination object may be missing important characteristics.

3.4.1.2 Attribute Correlation Metrics

In addition measuring the correlation among objects, metrics must also be calculated for the correlating the attributes associated with that object. Given a common lexicon, such as the OMDD, attribute values with the same name have an more similarity than those that do not. However, the name alone is not enough to guarantee identical matches, since often it is not guaranteed that attributes with the same name are identical. Metrics similar to the object correlation metrics must be used: IS-A, HAS-A, PARENT-OF, and WHERE-IS. In addition to the OMDD, what is also needed is a common hierarchy of domain attribute types, such as "ASSAULT POSITION ***is-a*** POSITION", "POSITION ***is-an*** AREA" and "AREA ***is-a*** LOCATION". This ontology allows similar semantic types to be compared and the closeness determined; it also provides a potential conversion path among types.

Using this ontology, the IS-A and PARENT-OF metrics both determine the closeness along an inference path between a source attribute and destination attribute. The IS-A metric determines if a destination attribute is a child of one of the source attributes. The metric determines the inferential distance between the two. Similarly, the PARENT-OF metric determines if a destination attribute is a parent of one of the source attributes. Unmatched (Additional) parents in a PARENT-OF metric also do not affect the closeness for the attribute. This just means that the attribute is more complex than the source attribute being correlated, which is satisfactory. These two metrics can be combined to generate a correlation path from a specific source attribute to a more general attribute and then back to a more specific destination attribute. For example, ASSAULT POSITION can be correlated to an OBJECTIVE by following the inference path from ASSAULT POSITION to POSITION to AREA to OBJECTIVE, where OBJECTIVE is a specific type of AREA.

The HAS-A metric determines the closeness along a decomposition path between a source attribute and destination attribute. For example, suppose a ROUTE can be decomposed into a START POINT and END POINT. Then, a source ROUTE attribute can be correlated with START POINT and END POINT attributes of the destination object. The IS-A and

PARENT-OF metrics can be combined with the HAS-A metric so that the subattributes of attributes may also be matched with the destination attributes.

Finally, the WHERE-IS metric can be used to locate attributes present in more general or specific objects in the class hierarchy. The closer the attribute is found to the destination object, the more similar the source and destination objects are. This metric can also be combined with the previous metrics since, technically, all cases can apply to one attribute at the same time: it can be higher or lower in the tree, of a more general or specific type than the attribute being compared, and or decomposed into subattributes.

3.4.1.3 Incremental Decomposition and Abstraction

The correlation algorithm uses incremental decomposition and abstraction of objects to determine the closeness. Each source object is recursed into (its components which are subobjects) and is compared (via recursion again) to the levels of the destination object. Each object is decomposed into its sub-objects which are also correlated down to the primitive level. The correlation algorithm uses the following high level steps when correlating a source object:

1. Check for the presence of the source object at the given level of decomposition in the destination object.

2. If the object is not present, apply the WHERE-IS, IS-A, HAS-A, and SIBLING-OF metrics, using the maximum closeness result.

3. Recurse into the source behavior, performing these steps on each sub-object. Combine the results of the sub-object correlations and multiply the result by the closeness value determined in one of the two previous steps.

4. Repeat steps 1-3 on the next object at this same level of decomposition.

The attribute correlation algorithm follows the same basic steps, with the attribute metrics being applied instead. It is important to note again that objects can have an increased closeness if their names match, but objects that match in name are not necessarily equal. The closeness must be determined down to the primitive level to determine an accurate correlation (hence the presence of step 3 above). The correlation algorithm uses the semantic closeness metrics defined earlier to determine the object closeness value. This value is calculated using closeness factors (decreases in closeness) for each metric, along with a few others. These factors may need to be adjusted for a specific destination FOM to guarantee proper correlation.

As each object is correlated, the metric that produces the best closeness value is combined with the aggregate closeness value of its sub-objects. The value is then combined with the other objects at the same level of decomposition and filtered up to the upper levels of

decomposition. At the top level, the correlation of the objects is combined with the attribute correlation to obtain a final correlation for the object in the range between 0 and 1.

The attribute correlation uses a similar version of the correlation algorithm but is slightly modified, because it is also focused on a possible conversion path, as well as the closeness. This will aid in the actual transformation of the data mentioned below. These same algorithms and metrics also apply the interactions and their parameters as well; these are just more reduced cases.

The total closeness between a source object and a destination object in the found is defined by:

```
total_closeness = ATTRIBUTE_PERCENTAGE*attribute_closeness +
                  (1.0 - ATTRIBUTE_PERCENTAGE)*object_closeness
```

3.4.2 SOM-FOM translation

Once the FOM format and the relationships between the SOM objects and the FOM objects have been identified, transformation routines need to be developed for each federate to conform to the new FOM. As the SOM-FOM arbitration process is executed, transformations can be identified based upon how close the representations are. The transformations are built from primitive transformations ,such as those shown below. Since the "closeness" determination is done recursively up the class hierarchy and into aggregated structures, the transformation process is as well. In fact, the additional closeness metrics and heuristics described above are required for the transformation to be determined. Once the transformation has been identified, the transformations can be stored in a "case base" so that transformations can be used again, or used as the basis for new sets of transformations.

Since a transformation is built up from a set of primitive transformations, not all transformations will always be found in the case base. Heuristics will suggest either the closest transformation or suggest and entirely new one that has been built up from other transformations in the case (primitive or otherwise). The user can then alter these transformations if necessary and store them back into the case base. The transformations themselves will be developed using a high-level transformation grammar that will be parsed and stored in the case base. The following example illustrates this grammar:

```
SOM OBJECT A                    FOM OBJECT B

a1  float32                     a1 int32
a2 {                            a2 {
        x float64                       x float64
        y float64                       y float64
        z float64               }
}
```

a3 int32	a3 {
	pos1 float32
	pos2 float32
	pos3 float32
	}
a4 float64 (precision x)	a4 float64 (precision y)

Here there are two different representations (A and B) of the same object. A set of mapping operations can be created to transform A into B as follows:

```
trunc(a1,a1)
remove(a2,a2,z)
user-defined-expand(a3,a3)
adjust-precision(a4,a4,new_precision)

user-defined-expand(source,dest)
        dest.pos1 = toreal(source) + 6.759
        dest.pos2 = toreal(source)
        dest.pos3 = 0.0
```

Potential primitive operations include truncation of values, removal of attributes, precision adjustments, addition of default values, conversions between integral types, conversions between real and integral types, and user defined conversions between unlike types. Since attribute a1 is converted from a float to an int type, a truncation operation is used. In object B, attribute a2 does not have a z component so it is removed. Attribute a4 simply needs a precision adjustment. Attribute a3 is more complicated since it is converted from an atomic type to an aggregate structure. A user defined expansion is used in this case which is composed of more primitive transformations to create the mapping. Inverses can also defined as follows:

```
toreal(a1,a1)
add(a2,a2,z,default-function)
user-defined-collapse(a3,a3)

default-function
  return query("Default value for attribute z of member a2")
```

Note that a default function is required for adding attributes which in some cases may require input from the user. Also, a user defined collapse function is needed to transform an aggregate structure into an atomic one (it is not presented here).

Once the transformations have been determined they are stored in the case base as a mapping between SOM Object of type A and FOM Object of type B. The case base itself will be stored in an object database as a set of transformation services associated with each kind of object. Given a correct API, federates can create instances of these objects (attributes are

objects as well) at run-time allowing the dynamic loading of different SOM-FOM transformations without re-compile. This of course requires that legacy federates also have an internal-SOM transformation routine in place.

4. Scenario

Under this effort, a scenario was not required. However, for demonstration purposes a simple scenario had to be developed to exercise the tool integrations and the environment. The scenario consisted of objectives in XEmacs, a scenario in ModSAF, an ideal FOM created from this scenario, a full FOM created with the OMDT from the ideal FOM.

5. Conclusion

The prototype HLA Integrated Development Environment (HIDE)_ demonstrated that an HLA integration framework is not only feasible, but is highly desirable for the future evolution of HLA development. The results are promising. Tools not designed to work together have been integrated into an executable FEDEP process model, a level 1 integration. These tools include Netscape (web-browsing), XEmacs (word processing), WinWin (requirements definition), OMDT (object model development), and ModSAF (scenario-based object model development). In addition, many of these tools (OMDT, WinWin, ModSAF) were integrated at level 2, with the data they produce translated into standard CORBA object representations that have the potential of being semantically linked, resulting in a seamless FEDEP knowledge base. In short, we have surpassed our original objectives, which included:

1. **Implement an initial version of HIDE**: accomplished
2. **Integrate current HLA tools**.: accomplished
3. **Integrate selected software engineering tools**: accomplished
4. **Implement an initial HLA arbitration tool**: requirements accomplished; tool work discontinued based on similar Aegis work.
5. **Develop a detailed plan for the implementation of a robust version of HIDE and the tools within it**: accomplished (under a separate document).

In addition to these objectives, an executable FEDEP process model was developed, with tools integrated into it, that helps to structure the distributed, collaborative work of federation development. All of this was accomplished under cost.

The FEDEP process is complicated and requires integrated tools to support it. These tools come from many sources and produce data in many different formats, yet many of them are

very useful to HLA developers. Nevertheless, they were not developed to work together, producing stovepiped data. Therefore, these tools are not as useful in isolation as they are when integrated. Furthermore, as data from these tools are integrated as a seamless CORBA knowledge base, change is easier to manage within the FEDEP process, since the impacts of changes can be analyzed and made visible to all stakeholders. Ensuring a seamless, visible, consistent knowledge base produced by a wide variety of HLA support tools is the focus of follow-on work in HIDE.

HLA FEDEP ACTIVITIES	Outputs	Tool	Information Access
Sponsor needs identification	Formalized problem statement	XEmacs	
		Word proc (doc)	HTML/Acrobat (Doc linkages)
Objectives development	Detailed federation objectives	HLA Construction Kit	
		Proj mgmt tool	
		Spreadsheet	
Scenario development	Fed functional scenario spec (R)	Word processor	
	Ideal FOM	Event trace tool	
		Graphics tool	
		Scenario databases	
		Xgen	TOE, Equipment, Scenarios, Modsaf
		SGT	
		HLA Construction Kit	
		MO Scenario Tool	
Conceptual analysis	Federation conceptual model	Object modeling tools	OMMS
	Detailed federation requirements	OMDT, OM Template	OMDD
		Word processor	OMDT
		OMMS tools	
		OMDD tools	
		WinWin	
		Conceptual Modeler	
		Traceability tools	
Federation design	FOM reuse components	WinWin	FOM Library
(Federate design)	FOM member list	EMMA	SOM Library
	Fed development approach	SE process tools	
		Together/J	
		Rational Rose	
Federation development	HLA FOM	Web & tools	Reference FOM
	Fed commonality matrix	TASC OMDT	Similar FOMs
	Scenario data	HTML	HLA OM Development Process
		OMDT Pro	OML
		Visual OMT	OMDD
		FedComposer	
Execution planning	Fedex Workbook (5 tables)	FEPWT	
	RTI Initialization Data (RID) file	Word processor	
		SEPT	
		EMMA	
Federation integration & test	Formal test plan	FTS	HLA Management Object Model
	Test data collection files	FedProxy	
	Cost & schedule impacts of modificatio	Mgmt tool	
		FVT	
Federation execution	Test data collection files	Database tools	
	Playback data files	RTI analysis tools	
		FMT	
		DEM	
		FedDirector	
		ModOIS	
Federation AAR	Analyzed data	Statistical analysis tools	
		Postprocessing tools	
		FDCT	
Feedback	Action items (identify Fedep activity)		
	Reuseable federation products		MSRR
	FOM, SOM mods, OMDD, Scenario,		
	Conceptual model, etc		

In summary, many tools have been examined under this effort and we plan to analyze many more for integration into HIDE in follow-on work. In addition, we have analyzed what it means to integrate these tools into the FEDEP process. The above table lists the FEDEP activities explored, their output products, and the category of tools (or specific tools) that need to be part of HIDE to support the FEDEP. This provides the point of departure for future HIDE work.

6. Points of Contact

ADST II HIDE Team
Kent Bimson
Project Director
407-306-5631
Kent.D.Bimson@cpmx.saic.com

Christopher Dean
Lead Engineer
407-306-4038
Christopher.Dean@cpmx.saic.com

STRICOM
Rob Miller
Project Director
407-384-3685
MILLER1@stricom.army.mil

Bryant Lafoy
Project Engineer
407-384-3864
lafoyb@stricom.army.mil

Acronym List

ADST	Advanced Distributed Simulation Technology
API	Application Programmer Interface
BAH	Booz, Allen & Hamilton
CAFDE	Computer Aided Federation Development Environment
CATT	Combined Arms Tactical Trainer
CDF	Core DIS Facility
CDRL	Contract Data Requirements List
CLIPS	C Language Integrated Production System
CMMS	Conceptual Model of the Mission Space
COCOMO	Constructive Cost Model
CORBA	Common Object Request Broker Architecture
DARPA	Defense Advanced Research Projects Agency
DEM	Distributed Exercise Management
DIF	Data Interchange Format
DIS	Distributed Interactive Simulation
DMSO	Defense Modeling Simulation Office
DO	Delivery Order
DTD	Document Type Description
EDCS	Evolution Design of Complex Software
EMMA	Evolution-Memory Management Assistant
FAS	Feasibility and Analysis Study
FDCT	Federation Data Collection Tool
FEDEP	Federation Execution and Development Process
FEDEX	Federation Execution
FEF	Federation Execution Framework
FMT	Federation Management Tool

FOM	Federation Object Model
FVT	Federation Verification Tool
GRTI	Georgia Tech Research Institute
GUI	Graphical User Interface
HIDE	HLA Integration Development Environment
HLA	High Level Architecture
HTML	Hyper Text Markup Language
IDL	Interface Definition Language
IR&D	Internal Research and Development
LMC	Lockheed Martin Corporation
ModSAF	Modular Semi-Automated Forces
MOM	Management Object Model
MSRR	Modeling Simulation Resource Repository
OCI	Object Community Interchange
OLE	Object Linking and Embedding
OMDD	Object Model Data Dictionary
OMDT	Object Model Development Tool
OML	Object Model Library
OMT	Object Modeling Template
PVD	Plan View Display
RDF	Resource Definition Format
RTI	Run Time Infrastructure
SBIR	Small Business Innovative Research
SEPT	Scenario Execution Planning Tool
SIW	Simulation Interoperability Workshop
SGT	Scenario Generation Tool
SOM	Simulation Object Model
STRICOM	(US Army) Simulation Training and Instrumentation Command
TASC	The Analytical Sciences Corporation

TCL	Tool Command Language
TCP/IP	Terminal Control Program/Interconnect Protocol
TOE	Tables of Operation Equipment
UDO	Unilateral Delivery Order
UML	Unified Modeling Language
VTC	Virtual Technologies Corporation
WAN	Wide Area Network
WPI	Work Product Instance
XML	Extensible Markup Language
XSL	XML Style Language