Framework Encapsulations: A New Approach to CAD Tool Interoperability

Peter R. Sutton*
Cooperative Research Centre for Satellite Systems
Queensland University of Technology
2 George St.   Brisbane QLD 4001   Australia
Phone: +61-7-3864-2458
p.r.sutton@ieee.org

Stephen W. Director*
College of Engineering, University of Michigan
1221 Beal Avenue
Ann Arbor MI 48109
Phone: +1-734-647-7010
director@umich.edu

ABSTRACT

Today's complex leading-edge design processes require the use of multiple CAD tools that operate in multiple frameworks making management of the complete design process difficult. This paper introduces the concept of framework encapsulations: software wrappers around complete CAD frameworks that allow the design data and flow management services of a framework to be utilized by a common design process management tool. This concept has been applied to the Minerva II Design Process Manager, enabling Minerva II to manage the design process across multiple CAD frameworks, and potentially multiple design disciplines.

1. INTRODUCTION

In today's complex design processes, designers require multiple CAD point tools in order to perform the various tasks that make up a design process. In "leading-edge" design processes, where "best-of-class" point tools are required, CAD tools may be obtained from several different sources: commercial CAD tool vendors, in-house development groups and academic research groups. Due to the different sources, these tools often operate within disparate CAD framework environments, making it difficult to manage the entire design process. This problem is exacerbated by the trend towards multidisciplinary concurrent design, where different parts of a system (e.g., hardware and software, or product and manufacturing processes) are designed in parallel by different groups of designers using different sets of CAD tools and environments. The aim of this work is to realize *unified design process management*, that is, simplified management of the design process across multiple CAD frameworks and potentially across multiple design disciplines.

CAD frameworks can be thought of as design environments which provide *design management services*. Early frameworks provided *design data management* and *design tool management* services - managing data format translations, tool command lines, and then on to design database services including the management of configurations, libraries and versions. More recent CAD framework systems, both commercial and research (e.g., [14],[16] and [17]), have provided further design management services which can be called *design flow management* or *design methodology management* [10]. These services provide for the automation of sequences of tool executions and for the specification of, and enforcement of, repeatable design steps. These sequences are usually represented by directed graphs called *flows* to show the movement of data between tools. CAD frameworks which provide

services at this level, will be referred to in this paper as *framework executive systems*, due to their ability to manage the execution of sequences of tasks.

Design process management [7] is a level of design management support above that provided by today's CAD frameworks. Design process management involves support for, and control of, all aspects of the solution of design problems - including conceptual (or exploratory) design, problem decomposition, backtracking, constraint propagation and management, and design history management. Design process management can be considered a new level of abstraction of design management services [1]. Just as the increasing complexity of VLSI design is driving design tools to operate at higher levels of abstraction (e.g., at the behavioral level instead of at the RTL or gate levels), the increasing complexity of design management means design management tools must evolve to operate at a higher level of abstraction: abstracting away some of the details of design flow and data management.

A design process manager (i.e., a design process management tool) providing such services must be both *discipline and framework independent*. Such a design process manager is not intended to replace existing CAD frameworks - there is no need to reimplement the lower level design management services. A design process manager should instead *manage* and *supplement* those lower level services. These requirements have led to the development of the concept of a *framework encapsulation*: a software wrapper around a complete CAD framework that enables the framework's design management services to be used and managed by a design process management tool. Just as tool encapsulations allow CAD point tools to be used within a CAD framework, framework encapsulations allow complete frameworks to be used from within a design process management tool.

Framework encapsulations have been implemented in a prototype tool called the *Minerva II Design Process Manager*. Minerva II is based on the Minerva Design Process Planner and Manager, originally developed by Jacome [7],[8]. Minerva II is a discipline-independent tool which implements some of the design process management features described above and, through framework encapsulations, can use the design flow and data management services provided by various CAD frameworks.

The remainder of this paper is organized as follows. Section 2 describes existing tool interoperability approaches. Section 3 describes the concept of framework encapsulations and issues regarding their implementation. Section 4 describes the use of framework encapsulations within Minerva II. Finally, Section 5 summarizes the paper and draws some conclusions.

2. EXISTING TOOL INTEROPERABILITY APPROACHES

The most closely related previous work in this area is that concerning *tool encapsulations* or, more generally, *tool interoperability* - the ability of CAD point tools to work together.

There are two important aspects to tool interoperability: the ability of tools to *share* design data and the ability to *be controlled* by a framework executive system. This section considers existing approaches to these aspects of tool interoperability in more detail.

2.1 Design Data Sharing

The tasks that CAD point tools perform are not independent, meaning that design data must be able to be shared between the tools. There are three common methods for sharing data: common file formats, procedural interfaces and messaging interfaces.

Common data formats, such as CIF for VLSI layouts, allowed tools to interoperate prior to the existence of CAD frameworks. More recent standards, such as EDIF 3 0 0 [6] continue to allow tools to work together, now usually within or between CAD frameworks.

CAD tools may also share data via a procedural or programming interface (PI) to an information provider (or server) process. An example is the CAD Framework Initiative Design Representation (CFI-DR) standard [2]. A PI standard allows design tools to interact with any information provider that implements the standard interface routines. Procedural interfaces are also used whenever design data is stored in a proprietary database, e.g., that of a commercial CAD framework. The interface may be provided in more than one language: typically a C interface is provided, along with an extension language interface for scripted access.

A similar approach to the PI is a messaging interface. In this case, a tool communicates with a whole suite of tools via a standard messaging architecture (such as ToolTalk from SunSoft [12]) using a predefined message dictionary (such as the CFI message set [3]). Messages allow separate processes to work together, but instead of a tool working with a static information provider (as in the PI case), the tool works with a dynamic suite of tools: any tool may answer requests, and new tools may be added to the system without any of the other tools being modified.

Standards (common file formats, standard procedural interfaces, and message dictionaries) may be defined to allow tools to work together. On the "leading edge", however, standards are not necessarily helpful: technology and tool capabilities are advancing more rapidly than the standardization process; tool vendors perceive no commercial advantage in fully supporting interoperability; and some "standards", such as Verilog and VHDL, are not fully portable due to incompatible implementations.

2.2 Design Tool Control

In order for a CAD framework executive system to control tool invocation, a standard interface to the tool must be provided. There are two methods by which this interface can be provided: *tool integration* and *tool encapsulation*. These approaches can together be referred to as *tool incorporation* [5]. Tool incorporation allows CAD frameworks to manage the execution of the tools, abstracting away the tool invocation details (e.g., filenames and command line arguments) and simplifying usability by the designer, possibly including the tool in a design flow. Data sharing, as discussed in Section 2.1 above, is one aspect of controlling this invocation - an executive system must be able to manage tool input and output. The two tool incorporation approaches are discussed below.

2.2.1 Tool Integration

Tool integration describes the process whereby a CAD point tool is tightly connected to a CAD framework. The source code of the point tool must be modified so as to implement communication with the CAD framework. The tool can directly obtain data to operate on, indicate completed results and, if desired, signal any "interesting" events during tool execution. Tool integration is often not possible due to source code unavailability and, even if possible,

is likely to be an expensive process, requiring knowledge of the tool implementation. Tool integration provides great flexibility, however, in that integrated tools and frameworks can communicate with each other at will (via procedural or messaging interfaces). This can lead to improved functionality over encapsulated tools, which may only be able to communicate with a framework at the start and finish of execution. Tool integration, however, usually restricts the tool to working within a particular framework.

2.2.2 Tool Encapsulation

Tool encapsulation describes the process whereby a CAD point tool is loosely interfaced with a CAD framework. A layer (or wrapper) of software (also called a tool encapsulation) between the framework and the tool is developed, perhaps automatically [5]. The software wrapper treats the tool as a black box - working only through the tool's defined interfaces. These interfaces will include the tool input (e.g., data files, command line arguments, environment variables and standard input), the tool output (e.g., data files, return codes and standard output and error), and sometimes intermediate information. On the framework side, the software wrapper communicates with the framework using a predefined procedural or messaging interface. At a minimum, this interface will provide hooks so that the tool encapsulation may obtain the data to operate on and return the result to the framework. The framework may also restrict the language(s) in which the software wrapper is written by only providing a communication interface for a limited set of languages.

2.3 Design Process Management Implications

The ideas of the tool interoperability approaches discussed above can be applied to the problem of making a design process management tool work with several, independent CAD frameworks. Whole frameworks could be incorporated into a design process management tool via integration or encapsulation. Most CAD frameworks are commercial products without readily available source code, meaning that the only practical approach is encapsulation, i.e., *framework encapsulation*. Just as tool encapsulations allow a CAD framework to communicate with a tool and utilize the tool's functionality, framework encapsulations allow a design process management tool to communicate with a CAD framework and utilize its functionality. The concept of framework encapsulations is described in the following section.

3. FRAMEWORK ENCAPSULATIONS

Definition 1: A *framework encapsulation* is a software wrapper around a CAD framework executive system that provides a common interface to the design management services provided by the CAD framework for use by a design process manager.

Framework encapsulations allow design process management software to be independent of CAD frameworks, yet utilize the design data and flow management services of the frameworks. Fig. 1 shows possible interaction between a designer, design process manager, framework encapsulations, framework executive systems and CAD point tools.

Section 3.1 describes the specific CAD framework design management services which need to be provided (through a framework encapsulation) to a design process manager. Section 3.2 describes some issues regarding the implementation of framework encapsulations and Section 3.3 describes the framework encapsulations which have been implemented.

3.1 Required Framework Services

A design process manager has specific requirements of a CAD framework executive system, the most important of which is the

ability to perform tasks (i.e., execute tools and flows) in order to solve problems. These tasks may be formalized in a design plan [7]:

Definition 2: A *design plan* is a partially ordered sequence of tasks which can be executed to solve a design problem.

Definition 3: A *partially ordered sequence of tasks* is a set of tasks in which conditions *may* be imposed on the order in which (some of) the tasks must be carried out.

The following sections enumerate the specific services which a framework encapsulation should make available to a design process manager. The services are considered in two categories, corresponding to the lower level framework services. The specifics of how these requirements are met are discussed in Section 3.2.

3.1.1 Design Flow Management

In order to solve design problems, a design process management tool requires knowledge of what problems a framework executive system is capable of solving and how (in terms of flows or tool executions) they will be solved. This is resolved on a per-problem basis through the process of design plan generation:
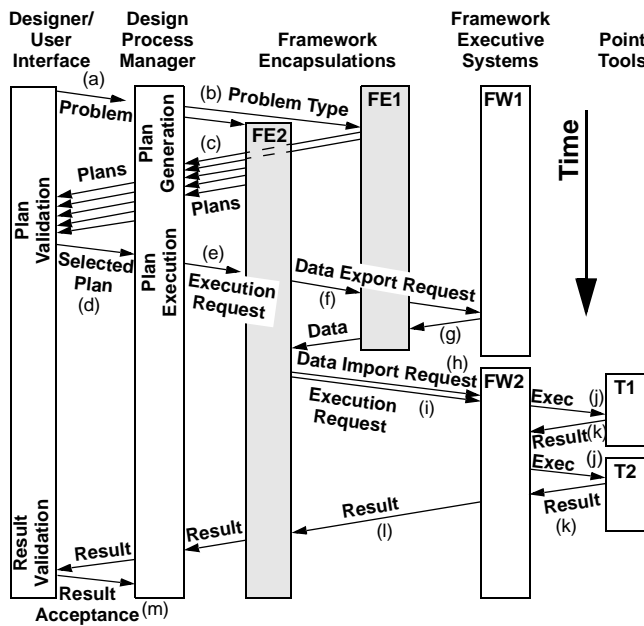

Figure 1. Possible communications over time between a designer, design process manager (DPM), framework encapsulations (FE1 and FE2), framework executive systems (FW1 and FW2) and CAD point tools (T1 and T2). (a) The designer selects a problem to solve. (b) The problem type is passed to the framework encapsulations. (c) FE1 and FE2 return possible plans to the DPM which passes them back to the user. (d) The user selects a plan. (e) The DPM sends an execution request to the relevant FE (FE2). (f) Assuming this execution requires data stored in another framework (FW1), FE2 generates a data export request of FW1 via FE1. (g) FW1 returns the data to FE2 via FE1. (h) FE2 imports the data into FW2. (i) FE2 can then pass an execution request to FW2. (j) FW2 executes the appropriate tools which make up the design plan. (k) FW2 records the results of their execution. (l) The overall plan result is passed back to FE2, the DPM and the designer. (m) The designer may accept or reject the plan result.

Definition 4: Design plan generation is the determination of a design plan to solve a design problem within a particular framework executive system, i.e., which tasks can be performed by which tools in what order so as to solve the design problem.

Along with the ability to generate possible plans within a framework executive system, a framework encapsulation must also be able to perform design plan execution, using appropriate data values as arguments. When a framework executive system completes execution of a plan, this must be detected by the framework encapsulation, and the execution result returned to the design process manager.

3.1.2 Design Data Management

Whereas a framework executive system can manage design flows independently of the data *values* being operated on[1], a design process manager may require data values to help control the design process, for example, initiating iteration if a constraint is not met. For this reason, framework encapsulations need to be able to extract data values from the encapsulated framework for use by the design process manager.

Because tools from multiple CAD frameworks often need to be used in the solution of a design problem, data transfer between CAD frameworks must be supported. This necessitates a data import/export mechanism.

3.2 Implementation Issues

There are a number of issues regarding the implementation of the above services within framework encapsulations. These issues are discussed below.

3.2.1 Design Plan Generation

For a framework encapsulation to determine whether a framework executive system can solve a certain problem it must determine whether it can generate (or has stored) a design plan that solves that problem. To do this, a framework encapsulation needs specific information about the problem to be solved, namely, the *objective* of the problem (e.g., placement at the layout level of abstraction), the *type of artifact* being operated on (e.g., operational amplifier), the *input data types* (e.g., transistor level description, process technology) and the *required output data types* (e.g., layout). In the parlance of Minerva II, this information is considered a "problem type" and is described in detail in [13].

There are several approaches a framework encapsulation could take to design plan generation. One would be for the framework encapsulation to contain complete knowledge of all of the plans that a framework executive system is capable of performing (or those that are supported) along with which tools can perform which tasks in each plan. Such a framework encapsulation would require maintenance to update possible plans every time a change is made to the CAD framework, e.g., a new tool is added.

Another design plan generation approach would be to dynamically build plans in response to information about the problem to be solved. Design plan generation in this context bears some relationship to planning in the artificial intelligence sense [9] - deriving a sequence of actions to change the state of a system. AI planning technology may serve as an implementation technology for plan generation within a framework encapsulation, although in most cases, simple algorithmic approaches would suffice.

---

[1] Executive systems must know meta-data about the data, e.g., the type and location of the data, and the name of the tool which created the data. It is the data (values) themselves that are not needed by the executive system.
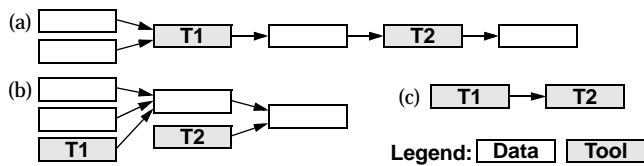
Figure 2. CAD framework flow representations. (a) Bipartite flow graph, showing data "flowing into" tools which produce data. (b) Task graph [14], showing tools and data as dependencies of data to be created. (c) Tool flow, where data is implicit, as used by some commercial frameworks.

Other approaches to design plan generation could fall between these two extremes. For example, a framework encapsulation may store a set of plan-templates, which are filled in by querying the framework for the tools that are available to perform the tasks that make up each plan. It has the advantage over the first approach that maintenance is only needed when the tasks that the framework can perform are changed and not when new tools are added.

The output from plan generation is a series of plans, each of which should be capable of solving the problem. (Multiple framework encapsulations may return plans, and each framework encapsulation may return more than one plan if desired.) Each plan is a sequence of tasks that can be represented as a flow or flows to capture the ordering constraints within the plan. CAD frameworks use many different flow representations - some examples are shown in Fig. 2. Framework encapsulations are responsible for mapping between the representation used by the framework and that used by the design process manager (which is a bipartite flow graph (Fig. 2(a)) in the case of Minerva II).

Each plan may also have choices associated with elements in the flow, for example, multiple tools may be available to perform some task, or multiple data instances may be available for some input (e.g., tool configuration options). Before plan execution, specific instances of such items would need to be chosen.

3.2.2 Design Plan Execution
Once design plans are generated by framework encapsulations, selected plans must then be executed. The design process manager will pass the selected design plans (and associated tool and data choices) to the relevant framework encapsulation for execution. Each unique combination of a design plan with selected tool and data items is considered by the design process manager to be a separate solution alternative; however, the underlying framework encapsulation need not execute them that way. Plans which have tasks in common may share the result of a single task execution. A framework encapsulation is also free to execute all tasks in all design plans, even if some tasks are shared between plans.

3.2.3 Execution Result Return
Upon the completion of plan execution, the result of the execution must be returned from the framework encapsulation to the design process manager. There are two aspects to the execution result: the completion status and the output data.

The completion status specifies whether the plan was executed successfully. "Success", in this case, does not necessarily mean that the output data is correct or useful; it means that CAD tools performing the tasks in the plan did not return errors[1]. In the case of an editing task, for example, it means the editor quit successfully - it does not mean that the edited data meets particular requirements, or is even syntactically correct[2]. Plan failure may occur for several

reasons, e.g., tool failure or data incompatibility. Minerva II returns notification of a failure but may not be able to provide the reason - this depends on the capabilities of the executive system. In some cases, a designer may have to investigate the failure at a lower level in the design management hierarchy, e.g., examining tool execution log files.

Output data from a successful design plan execution is typically stored in a file or a CAD framework database. The framework encapsulation is responsible for returning this data to the design process manager. In some cases the actual data may be returned (e.g., simple strings and numerical values), whereas in other cases, a pointer to the data (e.g., filename or database object-id) is returned.

3.2.4 Data Value Extraction
Data value extraction refers to a framework encapsulation obtaining data from a CAD framework for use by the design process manager in *controlling* the design process. As outlined above, execution result return is one form of data value extraction. Our approach will not consider any other form of data value extraction. This is based on the assumption that data values within a CAD framework are created as the result of some task, and that if this task (or a data value it created) was of importance to the design process it would be managed by the design process manager.

Data value extraction should not be confused with data import and export, which refer to data transfer for the purpose of making data available for use by other CAD frameworks. Data import and export are discussed in the following section.

3.2.5 Data Import and Export
If a tool from one framework executive system is to access the data contained within a different framework executive system, then the owning framework must be able to export data (via its framework encapsulation) to the other framework - which must be able to import it.

The framework executive systems could be running on separate hosts, necessitating network communication. The simplest data import/export mechanism (and the one used in Minerva II) is to use files in a shared file-system such as AFS or NFS. If the design data of interest is not already available as a shared file (e.g., it is stored within a CAD framework database), then the framework encapsulation is responsible for converting the database-stored data into a file with a common format that may be imported by the other framework encapsulation.

3.3 Framework Encapsulation Implementations
Several CAD frameworks have been encapsulated into the Minerva II Design Process Manager using the method described above. These include:

- the Hercules Task Manager [14], configured to perform tasks in the area of analog integrated circuit design;
- the Hercules Task Manager, configured to perform software development tasks; and
- the Cadence Design Framework II.

Although these framework encapsulations are limited in terms of the number of tasks encapsulated[3], they illustrate that the concept applies to the frameworks of different types.

---

[1] On UNIX systems, this can be interpreted as a tool exit status of 0.

[2] Of course, a check for syntactical, or other, correctness could be a task within a design plan, or could be part of a tool encapsulation itself.

[3] Description of a *task* being encapsulated refers to a task which is able to be returned to a design process manager in a generated plan. A framework encapsulation need not allow all tasks which a framework can perform to be available for plan generation.

The major cost in constructing a framework encapsulation is in implementing the communication mechanism between the framework encapsulation and the CAD framework. For the frameworks above, this effort was on the order of one week. Once this mechanism is built, and one task within a framework has been encapsulated, it is relatively simple to encapsulate further tasks. Similarly, once one data import or export function has been encapsulated, it is relatively simple to encapsulate others. Once this is completed, it may be possible to automate the framework encapsulation process using a simple configuration file as input.

The frameworks are programmed, or interfaced to, using different extension languages: Hercules can be controlled using Tcl [11], and the Cadence Design Framework II with SKILL [4]. The framework encapsulation for each of the frameworks is written in Tcl along with the extension language for that framework. For example, the Cadence Design Framework II framework encapsulation consists of Tcl routines which, when called, generate SKILL code to be sent to a Cadence UNIX process. Framework encapsulations are implemented as Tcl *packages* which can be dynamically loaded into a Tcl interpreter, such as one of the Tcl interpreters within the Minerva II design process manager.

4. MINERVA II

The Minerva II Design Process Manager is a discipline and framework independent software tool providing design process management capabilities serving multiple designers working with multiple CAD frameworks, possibly in multiple design disciplines. Minerva II is configured to manage the solution of problems in a particular discipline by a description written in DDDL (**D**esign **D**iscipline **D**escription **L**anguage [15]). In this section we briefly describe how Minerva II interacts with framework encapsulations and designers[1]. The main events in this interaction are plan generation, plan validation, plan execution, and result validation. These are shown in the communication sequence of Fig. 1 and are described in the following sections, followed by a discussion of the advantages and limitations of Minerva II.

4.1 Plan Generation

Plan generation occurs within all of the framework encapsulations loaded into the Minerva II server. Each is passed information about the type of problem that is being addressed so that any plans which address that problem type may be returned, usually within a fraction of a second. These plans are then subject to validation/ selection by the designer before execution.

4.2 Plan Validation

Plan validation (or selection) within Minerva II is performed by the designer in a window like that shown in Fig. 3. Each plan is labelled with the name of the framework (encapsulation) which generated it and the textual description of the plan. Also shown is a graphical representation of the flow to be executed by the framework if this plan is selected. This flow may have associated choices to be made, e.g., tools to be placed in particular nodes of the flow. These choices are presented in the listboxes to the right of each flow. Plans are selected by checking the button on the left of the plan. Any number of plans may be selected for execution. Where multiple options are given for a plan and it is desired to execute the plan more than once with different options, the plan may be duplicated (using the "Duplicate Plan" button shown in Fig. 3) and different options chosen. After suitable plans (and options) have been selected, these are returned to the respective framework encapsulations for execution.

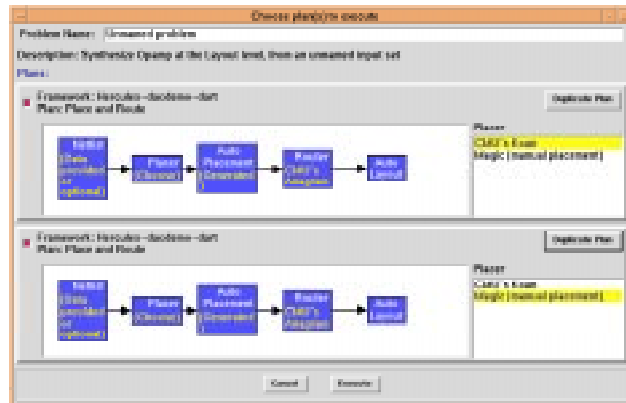[1] Further details may be found in [13] and [15].

Figure 3. Minerva II Plan Validation window. The one returned plan has been duplicated using the "Duplicate Plan" button so that both tool choices may be made (i.e., two plans executed).

As mentioned earlier, the flow representation used by Minerva II is a traditional bipartite flow representation. The displayed flow is a mapping from the representation used by the encapsulated framework. For example, the plan in Fig. 3 (there really is only one, which has been duplicated) was generated by the framework encapsulation for the Hercules Task Manager. The Hercules task graph representation of that same flow is shown in Fig. 4.

4.3 Plan Execution

Plan execution takes place within the encapsulated CAD framework executive systems. The tools executed as part of the plan may or may not be interactive. If user interaction is required, it is the framework encapsulation's responsibility to ensure that the tool display is correct, as it is possible for the tool to be executing on a remote host.

4.4 Execution Result Validation

A plan execution can have two possible outcomes: *failure*, meaning that the tool execution failed, e.g., the tool returned an error status; and *success*, meaning that a new object version is to be created with new properties generated as part of the execution. The results of a plan execution through a framework encapsulation are presented to a designer as a new object version that has been created (Fig. 5). The designer may then accept or reject the result.

4.5 Advantages and Limitations

Framework encapsulations allow for unified design process management by a design process manager such as Minerva II. Unified design process management has several advantages over traditional approaches. Designers are able to focus upon the design
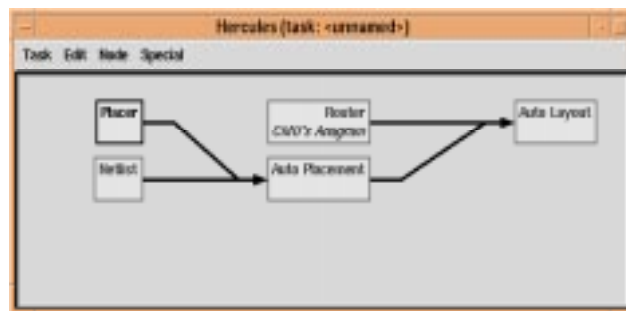

Figure 4. Hercules Task Manager representation of the design flow shown in Fig. 3.
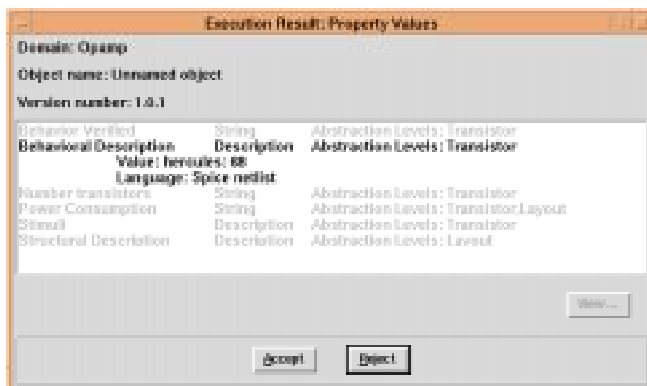
Figure 5. Plan Execution Result Validation in Minerva II.

problems being solved, rather than on flow, tool or data manipulations. Interactions between designers working in different disciplines (e.g., hardware and software design) and different frameworks can be managed easily. Consistency constraints can be monitored so that inconsistencies can be detected earlier and backtracking commenced sooner. Both backtracking and problem decomposition may be managed. All of these features can lead to shorter design cycles.

There are some disadvantages to our approach. Writing framework encapsulations and the DDDL description of a design discipline takes time. However, as was noted earlier, the major cost in encapsulating a framework is implementing the communication mechanism between the framework encapsulation and the framework executive system. Once that is operational, encapsulation of additional capabilities is a relatively simple process.

Just as tool encapsulations may not allow access to the complete functionality of a tool, framework encapsulations may not allow access to the complete functionality of a CAD framework. In order to perform additional operations it may be necessary to use a CAD framework independently from the design process manager. Similarly, just as an encapsulated point tool may not provide a suitable interface to completely automatically control tool function or invocation, a CAD framework executive system may not implement all the necessary capabilities (or at least, may not provide an interface to allow all operations to be performed programmatically). In this case, a *partial encapsulation* may be provided with manual intervention required to perform some steps.

5. CONCLUSIONS

Existing tool interoperability approaches do not suffice for today's leading-edge design processes where multiply-sourced CAD tools and CAD frameworks are required. Attempts to standardize design data interfaces (procedural and messaging interfaces and file formats) have, in general, not been accepted and are unlikely to be accepted due to rapid technological change and vendor-perceived competitive advantage reasons.

The approach we have taken in this paper leaves existing tools operating within existing environments but introduces a new level of design management: design process management. A design process management tool communicates with existing CAD framework executive systems through framework encapsulations. Framework encapsulations are closely related to, but a step above, tool encapsulations used in CAD frameworks. Framework encapsulations encapsulate several aspects of lower level design

management services for use by a design process management tool. These aspects include design plan generation, design plan execution with result return, and data import and export.

To illustrate the applicability of this approach, framework encapsulations have been applied to a design process management tool called Minerva II. Framework encapsulations of the Hercules Task Manager and the Cadence Design Framework II allow Minerva II to control the design process at a high level - managing tool interoperability across multiple CAD frameworks and simplifying design management.

6. REFERENCES

[1] J. B. Brockman, T. F. Cobourn, M. F. Jacome, and S. W. Director, "The Odyssey CAD Framework," *IEEE DATC Newsletter on Design Automation*, Spring 1992.

[2] CAD Framework Initiative, Inc., *Design Representation Programming Interface: Electrical Connectivity*, Version 1.4.0, Oct. 1994.

[3] CAD Framework Initiative, Inc., *CFI Message Dictionary*, Version 1.0 (Initial Draft), Jan. 1994.

[4] Cadence Design Systems, Inc., *SKILL Quick Reference*, Version 4.3, Mar. 1994.

[5] J. Câmara and H. Sarmento, "AutoCap: An Automatic Tool Encapsulator," in *Proceedings of IFIP WG 10.5 - 4th International Working Conference on Electronic Design Automation Frameworks*, pp. 31-40, Nov. 1994.

[6] Electronic Industries Association. *Electronic Design Interchange Format Version 3 0 0*. Doc. EIA-618 (4 volumes), Dec. 1993.

[7] M. F. Jacome, *Design Process Planning and Management for CAD Frameworks*. Ph.D. Thesis, Carnegie Mellon University, Department of Electrical and Computer Engineering (Report CMUCAD-93-65), Nov. 1993.

[8] M. G. Jacome and S. W. Director, "A Formal Basis for Design Process Planning and Management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 10, pp. 1197-1211, Oct. 1996.

[9] N. A. Kartam and D. E. Wilkins, "Towards a Foundation for Evaluating AI Planners," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* (AI EDAM), vol. 4, no. 1, pp. 1-13, 1990.

[10] S. Kleinfeldt, M. Guiney, J. K. Miller, and M. Barnes, "Design Methodology Management," *Proceedings of the IEEE*, vol. 82, no. 2, pp. 231-250, Feb. 1994.

[11] J. K. Ousterhout, *Tcl and the Tk Toolkit*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1994.

[12] Sun Microsystems, *The ToolTalk Service - a SunSoft White Paper*, Sunsoft Inc., Jun. 1991.

[13] P. R. Sutton, *A Framework and Discipline Independent Approach to Design Process Management*, Ph.D. Thesis, Carnegie Mellon University, Department of Electrical and Computer Engineering (Report CMUCAD-97-18), May 1997.

[14] P. R. Sutton, J. B. Brockman, and S. W. Director, "Design Management Using Dynamically Defined Flows," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 648-653, June 1993.

[15] P. R. Sutton and S. W. Director, "A Description Language for Design Process Management," in *Proceedings of the 33rd Design Automation Conference*, ACM/IEEE, pp. 175-180, June 1996.

[16] K. O. ten Bosch, P. Bingley, and P. van der Wolf, "Design Flow Management in the NELSIS CAD Framework," in *Proceedings of 28th ACM/IEEE Design Automation Conference*, pp. 711-716, June 1991.

[17] P. van den Hamer and M. A. Treffers, "A Data Flow Based Architecture for CAD Frameworks," in *Proceedings of 1990 IEEE International Conference on Computer-Aided Design* (ICCAD), pp. 482-485, Nov. 1990.