Incorporating Design Schedule Management into a Flow Management System*

Eric W. Johnson & Jay B. Brockman
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556

Abstract - In this paper we present an approach to incorporate design schedule management services into a flow management system. The basis of our approach is to derive a design schedule from the simulation of a flow execution. Actual flow execution can then be tracked against the proposed schedule via design metadata. We verify our approach by implementing design scheduling into the Hercules Workflow Manager.

I. Introduction

As designs continue to grow in size and increase in complexity, the pressure placed on project managers to meet design schedules increases. *Project Management Systems*, such as MacProject or Microsoft Project[1], are available to assist the project manager in managing design schedules. These systems provide the ability to create proposed schedules and track the actual schedule as a design moves to completion. At present, schedule management systems are typically separate from the system that controls the creation and execution of the design process itself. Project managers acquire projected and actual completion dates from the different designers working on the project, and manually insert the information into their project management system. At the same time, there is an increasing interest in graph-based *Design Flow Management Systems* for executing and tracking design processes. As a process is executed within a design flow management system, the designer would report the progress to the project manager, who in turn would make the appropriate update in a project management system. This paper describes an approach for integrating both design schedule management and process management features in a single system.

By merging schedule management and process management together, schedule management services can be directly linked to the design process. There are several advantages to having a combined system. First, the design process decomposition developed during schedule planning can also be utilized for flow management. Second, the project schedule can be automatically updated because the status of the flow is maintained within the flow management system. Finally, previous schedule information can be used for planning future projects. For example, previous schedule data can be used to predict the duration of future projects or to optimize the resources associated with future projects.

In this paper we will show how to incorporate a schedule model into a general flow management model and describe how project planning can be accomplished in the context of design process execution. We believe that our approach is sufficiently general to be applicable to a variety of flow management systems. In the next section we will review several available design flow management systems highlighting the similarities of their underlying representations. Section III describes how a schedule model can be integrated into the flow management model, and Section IV illustrates the integrated representation in a specific flow management system, namely the Hercules Workflow Manager. Finally, we present our conclusions in Section V.

II. Flow Management Systems

Design flow management systems have been developed to provide designers with services to aid in the development and execution of the design process. A survey of flow management systems include the *Roadmap Model* developed at Philips Research Laboratories [6], the *NELSIS* CAD Framework [1], the *Hercules* Workflow Manager [11], the *History Model* developed at UC-Berkeley [4], the *Hilda* CAD Framework [2], and the *VOV* CAD System [3].

In general, it is a non-trivial task to compare flow management systems because each system may be based on a different data model or employ a different modeling language. Nevertheless, the four-level architecture in [6] and [7] can serve as a common basis for discussion. Through this generalization, we will later show how systems of this general architecture can integrate the schedule model. These four levels are described below:

- Level 1 - contains the basic elements used to create design flows.

1. MacProject is a product of the Apple Corporation and Microsoft Project is a product of the Microsoft Corporation.

*This research is supported in part by a grant from the Mentor Graphics Corporation.

- Level 2 - instantiations of Level 1 data linked together to create design flow models.
- Level 3 - describes the metadata objects created from the execution of a flow.
- Level 4 - depicts the actual design data generated from the execution of a flow.

Each of the flow management systems listed above can be described in relation to this four-level architecture.[1] A brief description of each system is presented. Table 1 follows with a summary of the system models represented in the four-level architecture.

The "Data Flow Based Architecture" or *Roadmap Model* described by van den Hamer and Treffers [6] is based on the Object Type Oriented Data Model (OTO-D) data model. The OTO-D model is used for both design flow management and storage for design metadata. The structure of the *RoadMap Model* introduced the idea of a multi-level architecture for a flow model.

The *NELSIS* CAD framework, developed at Delft University, also uses a flow-based design architecture. As with the *Roadmap Model*, the information model used for the design flow architecture in *NELSIS* is based on an OTO-D model [1], but is expanded to also provide support for data hierarchy [12].

The *Hercules* Workflow Manager is a flow management system within the *Odyssey* CAD framework initially developed at Carnegie Mellon University[11]. *Hercules* uses a task schema to help designers formulate and execute tasks. The task schema provides construction rules which define how tool and data resources can be combined to form tasks.

The *History Model* is a CAD system developed at U. C. Berkeley to provide support for the dynamic aspects of VLSI design [4]. The model is based on a task specification language and provides an integrated framework for managing both design operations and design data.

Hilda is a CAD Framework developed at Siemens Research Laboratory that uses a Petri net representation to describe design flows [2]. The Petri net representation allows *Hilda* to represent many flow management features. Since *Hilda* uses a Petri Net representation for the process flow, the functional building blocks are those associated with a Petri Net model.

Unlike the previous systems which focus on design flow management, the *VOV* CAD System, developed at UC-Berkeley, concentrates on monitoring and tracking design activities [3]. The authors in [3] feel a design process cannot be planned *a priori* and instead must be created as the designers work through design process.

III. Design Schedule Model

There are several different schedule models in use today. Some simply present tasks graphically (Gantt Chart), while others represent both the tasks and the constraints between them (PERT) [5]. Constraint or network models predominate in project planning. Intuitively when designers plan a schedule, they break the design process into a series of activities and estimate the time each of the activities should take as well as the resources needed. Partitioning the design process into a series of tasks is the same strategy that is modeled at Level 2 of the flow management system architecture. ***Therefore, one way to view the development of a design schedule is as a simulation of the execution of a flow.*** Just as Level 3 data is created when an actual flow is executed, Level 3 data may also be created when the execution of a flow is simulated. The difference, however, is that the Level 3 data created by the execution of a flow is the metadata associated with the actual design data, while the data created by the simulation of an execution should establish an approximate time

1. We present only a comparison between system models. For a thorough comparison of system features based on these models, see [9].

TABLE I. System Representation using the Four-Level Architecture

Level	RoadMap Model	NELSIS	Hercules	History Model	Hilda	VOV
1	FlowType (Tool) Pin (PinType)	Tool FlowGraph Activity Port (DataType)	Task Entity Tool Dep. Data Dep.	Task Templates		
2	Flow InSlot OutSlot	FlowHierarchy PortInst Channel	Task Node Task Tree Arc	Design Tasks Design Activity	Transitions Places Arcs Patterns	(Reusable) Trace
3	Run Representation RepUsage	ActivityRun Transaction	Entity Inst. Inst Dep.	Design Process	Tokens	Transitions Places Trace
4	Representation File Group	Transaction Design Object	Cyclops Data Object	Data Object	Tokens	Places

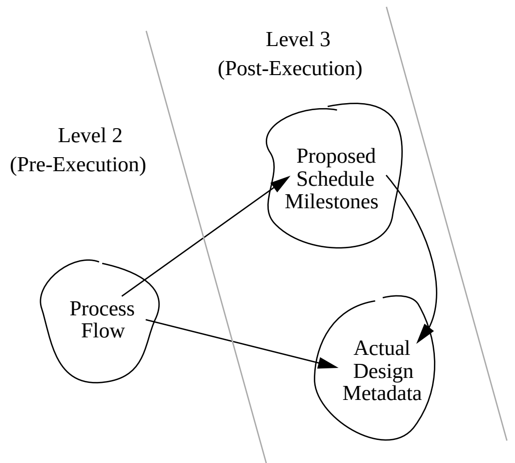

Fig. 1. Schedule Model within System Representation

frame for the execution of an activity. In other words, if Level 3 design metadata describes when an activity is performed and by whom, *Level 3 schedule data* ought to describe when an activity *should* be performed and which person or persons are assigned the task of performing that activity.

The methodology for schedule development via flow simulation can be incorporated into the four level flow management system as shown in Fig. 1. The additional information needed for the schedule model is represented at Level 3. This information describes the schedule created and approximated by the simulation of the task execution.

Simulating an execution involves creating design schedule instances for every activity in the design process. For each activity, schedule information can be gathered describing an activity before the actual execution of the process. This data can include the estimated starting time and duration length for an activity, and the resources needed. One of the main advantages of integrating design schedule management with flow management is that the metadata from previous designs is available. Thus, the duration of an activity can be based either on the designer's intuition or on the measured results of similar tasks.

In a typical design process, a given activity may need to be run several times before the design goals are achieved. In order to track the performance of a design flow against a schedule, it is necessary to establish which run of an activity signifies that a task is completed. In Fig. 1, this is represented as a link between schedule flow data and actual flow data. This link is created when the designer determines that the execution of an activity is completed. This link provides the schedule flow data with information about the execution of the process such as the actual starting and completion times of the design data associated with the respective activity.

IV. Model Representation in Hercules

A network schedule model was implemented in the Hercules Workflow Manager. Figure 2 displays the four-level architecture as implemented in Hercules. The design schedule model was added to the Hercules representation by creating a set of objects that described schedule flow data associated with the simulation of an execution. This was added to the Level 3 Hercules representation and is shown in Fig. 3.

The design schedule objects added to the Hercules representation mirror the actual flow data objects. A Run in the actual flow space corresponds to a **Schedule** in the schedule flow space. **Schedule Instance Nodes** correspond to Entity Instances and are connected using **Schedule Instance Dependencies**.

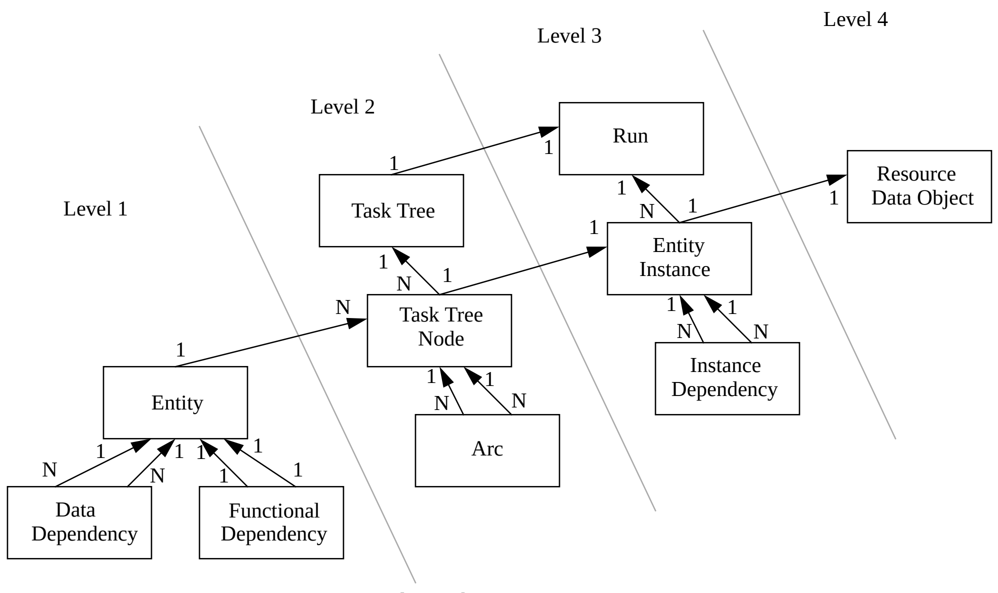

Fig. 2. Hercules Architecture Representation

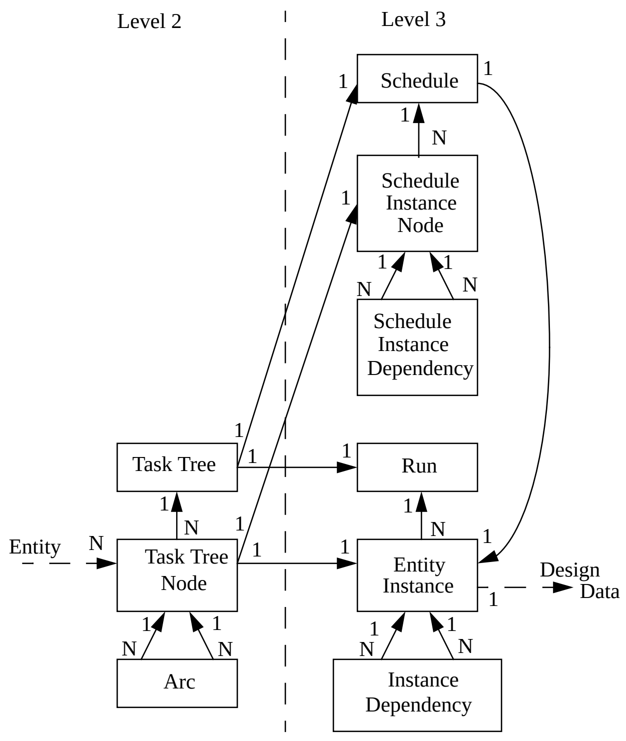

Fig. 3. Execution and Schedule Model in Hercules

A. Integrating Design Schedule and Execution

Integrating a design schedule plan into Hercules is an extension of actual process execution. The procedure associated with the development of a task schema and execution of tasks within Hercules is described in [11]. We will summarize this procedure and include the planning of a design schedule using the circuit design process as an example. At each step in the procedure, the state of the Hercules database will be described.

The first step in the procedure involves defining a task schema. A task schema describes the entities (tool and data classes) and the relationships between entities that are needed to model all tasks in a design process. Formally, the task schema can be described as a series of construction rules stated as mathematical expressions. Given a set of entities, $\{d_1, d_2, \ldots, d_n\}$, the expression $d_i \leftarrow f(d_1, d_2, \ldots, d_n)$ states that an instance of type d_i is created by applying function f to a n-tuple of entity instances of types $d_1, d_2, \ldots, d_n$. For example, an instance of a performance can be created applying a circuit simulator to instances of netlist and stimuli. This construction rule describes the "Simulate Circuit" activity and can be written as:

$$\text{performance} \leftarrow \text{simulator}\,(\text{netlist, stimuli})$$

The objects depicted in the schema correspond to the objects found in Level 1 of the data model. The Hercules task database is initialized from the schema by generating a series of containers that will hold the Entity Instances created during flow execution. The task schema for our example is shown in Fig. 4. The Hercules database thus would include containers for each of the entities shown in this schema.

In order to integrate schedule management into Hercules, containers must also be created for the Level 3 schedule instances. These containers are created from the same task schema. As the task entities are parsed into the database, schedule containers are created from the functions associated with each construction rule. For our example, two schedule instance containers are created, one to represent the activity "Create Circuit" and another to represent the activity "Simulate Circuit." Since these schedule containers are created from the task schema, the schedule model has no effect on the representation of Level 1 data in Hercules, nor should it in other flow management systems.

Once the task schema is defined and the task database is created, designers may begin scheduling and executing tasks that are based on the schema. In Hercules, a user prepares a task for execution by first extracting a task tree that covers the scope of the intended task. Next, tools and input data are bound to the task by assigning unique tool or data instances to each of the leaf nodes of the tree. The task is then ready for execution. Similarly, a user prepares for schedule planning by extracting a task tree that covers the scope of the intended task to be planned[1]. At this point, the task is ready for schedule planning or in other words, the execution of the task is ready to be simulated. During schedule planning, Hercules performs a post-order traversal of the task tree—that is, running from primary inputs to outputs, creating new schedule instances in the Hercules database for each activity. For our example, the state of the Hercules database after the planning step is shown in Fig. 5.[2] Note that in the figure different versions of schedule instances for each task can be generated. This is because the schedule plan can be updated at any time during the design process.

Once schedule planning has been performed, the designer can execute any portion of the design flow. As stated above,

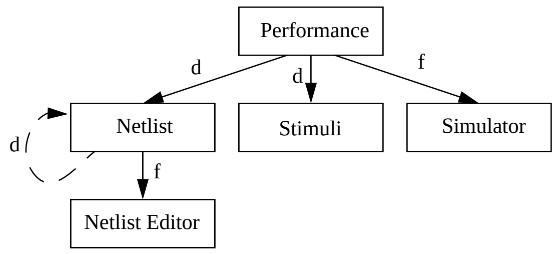

Fig. 4. Example Task Schema

1. At present the schedule portion of Hercules does not incorporate automatic prediction of run times. In the future, however, instances of tools and data that are bound to tasks may serve as inputs to such a prediction model.

2. Entity containers are shown as boxes, instances of entities are presented as circles and the lines show dependencies between instances.

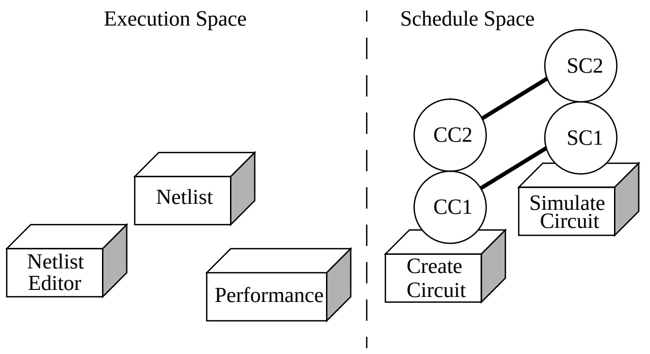

Fig. 5. Hercules Database during Planning Phase

a user prepares a task for execution and executes the task tree. Again Hercules performs a post-order traversal of the task tree. At each step in the execution, entity instances are created in the Hercules database for each non-leaf node. These instances contain the metadata associated with the actual design data and provide links to the actual design data. Fig. 6 shows the state of the database after the initial execution of the design process. Since tools are not tied to specific tasks and iterations of tasks can be performed, each entity container can hold multiple entity instances.

The execution of the design process continues until the designer verifies that the objectives of each task have been met. At this time, the entity instances in the database represent the final design data used in the process. The schedule instances can now be linked with the design data instances to represent completed tasks. The final state of the Hercules database including all entity instances, schedule instances and the links between them is portrayed in Fig. 7.

B. Examining Design Schedule Status

At any point in the design process, it is desirable to be able to compare the status of the execution of a task with the schedule plan. In Hercules, two approaches were developed to examine the status of a task with respect to a schedule: either visually, or through queries. A Gantt Chart is a common way to visualize schedule plan status. A Gantt Chart displays the schedule information as a series of tasks and displays graphically both the planned schedule and the

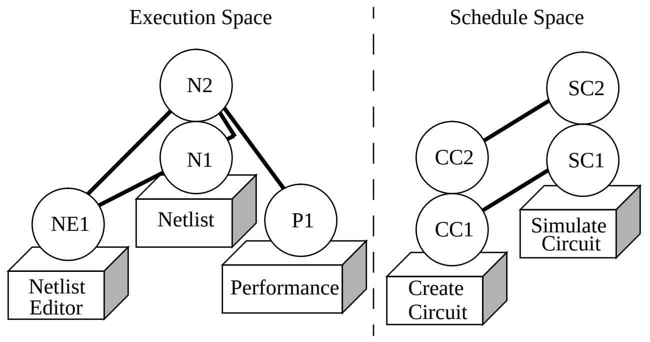

Fig. 6. Hercules Database during Execution Phase

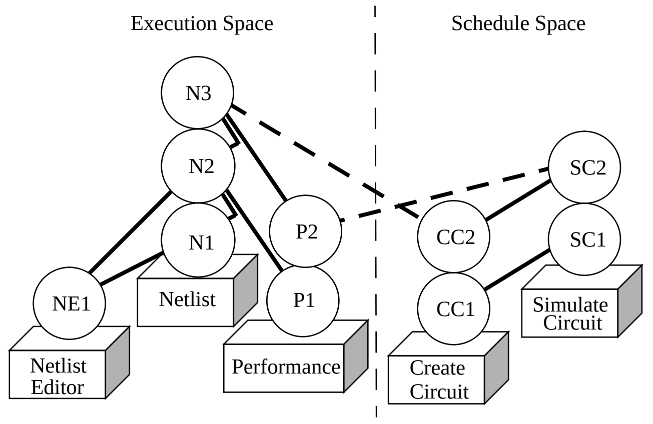

Fig. 7. Hercules Database at Completion of Execution

accomplished schedule. Both pieces of this schedule information are provided by the model described in Section III. For each activity, the proposed schedule information is obtained from the parameters stored in the schedule instance and the actual schedule information is gathered from the entity instance associated with the design data created from the execution of the activity.

Queries into the Hercules database are another means for providing schedule status information. Two types of queries are supported: queries into design schedule data, and queries into design schedule metadata. Queries into design schedule data are valuable because prior schedule plan data can be used as a resource. For example, a query to show the duration of an activity the last time it was performed could be used to predict the duration of the present design. Queries into design schedule metadata can be used to determine which schedules plans were used to create the present schedule plan. These queries are valuable because they can show the evolution of a design schedule.

C. Hercules User Interface Features

Enhancements were made to the Hercules user interface to access or modify schedule information in Hercules. These enhancements provide the user with new features including:

- Utilizing existing task tree for schedule plan development
- Updating schedule plan automatically as the design flow is executed
- Browsing the schedule instance database
- Viewing individual schedule plans

Schedule planning was integrated into the Hercules user interface such that the same task graph shown in the user interface could be used to both plan a schedule and execute a design flow. Fig. 8 shows the Hercules user interface. A visual representation of the task tree is the central feature of the user interface; schedule operations may be applied at

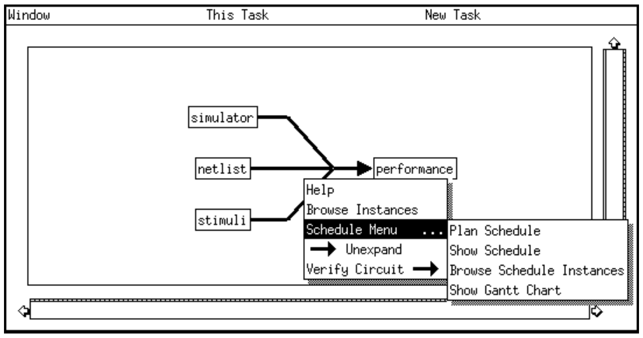

Fig. 8. Hercules User Interface

each node in the tree. A more complete description of the user interface without schedule additions is given in [11]. Mechanisms were also created in Hercules to automatically update actual schedule information as the process flow is executed. For example, once a data instance for the particular task is created, the actual start date for the task is set. Then when the task is completed, which may involve numerous iterations of the task, the user can link the final version of the task data to a schedule instance. If any slip in the schedule occurs, the schedule plan updates automatically to reflect the new schedule.

A schedule instance browser was developed to browse the schedule instances located in the Hercules database. The schedule instance browser is based on the existing entity instance browser used in Hercules and allows for the management of the schedule instances. With the schedule instance browser, the user can select, delete, or display schedule instances.

Viewing the schedule is important to both project managers and designers. A designer must be able to see the present time requirements of the current portion of a task. At the same time, a project manager must have the capability of viewing a portion of the overall schedule. Hercules can display both individual schedule information or a portion of the overall schedule using a Gantt Chart.

V. Conclusions

A limitation of current design management is that design schedule management and design process management are typically implemented separately. We have developed a model for integrating both design schedule management and design process management into a single system. The advantage of using a single system is that design schedule management features are directly linked to the design process allowing for easier tracking and updating of proposed schedules. It was shown that a schedule model could be implemented into a general four-layer flow management representation. Using this model, schedule planning can be accomplished by simulating the execution of a design flow. The schedule management model has been implemented and verified within the Hercules Workflow Manager. Because flow management systems provide similar representations and models to perform similar activities at each level, the implementation of the schedule model could be extended to other flow management systems.

Future work will focus on developing a schedule model that considers the architectural decomposition as well as the task flow along the lines of the model described in [8]. This will allowing greater precision in tracking, predicting, and optimizing design schedules.

References

[1] K.O. ten Bosch, P. Bingley, and P. van der Wolf, “Design flow management in the NELSIS CAD Framework,” In *Proceedings of the 28th ACM/IEEE Design Automation Conference*, 1991, pp. 711-716.

[2] F. Bretschneider, C. Kopf, and H. Lagger, “Knowledge based design flow management,” In *Proceedings of the IEEE International Conference of Computer-Aided Design*., 1990, pp. 350-353.

[3] A. Casotto and A. Sangiovanni-Vincentelli, “Automated design management using traces” In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, August 1993, pp. 1077-1095.

[4] T. Chiueh and R. Katz, “A history model for managing the VLSI design process,” In *Proceedings of the IEEE International Conference of Computer-Aided Design*, 1990, pp. 358-361.

[5] Q. W. Fleming, J. W. Bronn, and G. C. Humphreys, *Product and Production Scheduling,* Probus Publishing Company, 1987.

[6] P. van den Hamer, and M. A. Treffers, “A data flow based architecture for CAD frameworks,” In *Proceedings of the IEEE International Conference of Computer-Aided Design*, 1991, pp. 482-485.

[7] P. van den Hamer, K. O. ten Bosch, P. Bingley, M. A. Treffers, and P. van der Wolf, “A comparison of two approaches to design flow management by data schema analysis,” Jessi-SP1.

[8] M. F. Jacome and S. W. Director, “A Formal Basis for Design Process Planning and Management,” In *Proceedings of the IEEE International Conference of Computer-Aided Design*, 1994, pp. 516-521.

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

[10] G. N. Stilian, *PERT: A New Management Planning and Control Technique*, American Management Association, New York, 1962.

[11] P. R. Sutton, J. B. Brockman, and S. W. Director, “Design management using dynamically defined flows,” In *Proceedings of the 30th ACM/IEEE Design Automation Conference*, 1993, pp. 648-653.

[12] P. van der Wolf, O. ten Bosch, and A. van der Hoeven, “An enhanced flow model for constraint handling in hierarchical multi-view design environments,” In *Proceedings of the IEEE International Conference of Computer-Aided Design*, 1994, pp. 500-507.