Application of a Markov Model to the Measurement, Simulation, and Diagnosis of an Iterative Design Process

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

Abstract - This paper presents the use of a Markov-based model for analyzing iterative design processes. Techniques are developed for collecting process metadata and calibrating the model. An experiment is described that demonstrates the utility and accuracy of the model for simulating design processes and identifying design process bottlenecks.

I. Introduction

In today's highly-competitive market, it is imperative not only to deliver products that work, but also to deliver them on time. While great effort and expense has been invested in developing—and purchasing—more powerful point CAD tools, surprisingly little work has been done in creating techniques for analyzing and improving the design process itself. To address this need, we have developed a system for measuring, diagnosing, and simulating iterative, sequential design processes that utilizes a Markov-based model. To demonstrate the efficacy of the system, an experiment was performed with a group of designers, each working on the same design problem (the development of a computer program), wherein the methodologies used by each of the designers were observed, quantified, and fit as parameters of the model. Simulation results show that the model accurately represented the time spent in each of the stages of the design process, as well as in the process as a whole.

This paper also addresses several important issues related to the application of the model. Specifically, we describe techniques for collecting process metadata, and for analyzing the design process after completion.

II. Related Work

In developing a system for design process modeling and analysis, two components are critical: a means for collecting process metadata and a methodology for analyzing this data. To these ends, we view this work as an outgrowth of ideas from the areas of CAD frameworks and management science.

In recent years, systems known as *CAD frameworks* have been envisioned to provide designers and CAD system developers with services such as tool integration, data management, process management, and methodology management. In attempting to achieve this goal, various framework efforts have brought us closer to the stage where nearly all design activities may be performed in a computing environment, and the actions and outcomes of the design process may be archived. Of these, *design methodology management systems* have addressed issues such as which CAD tools and design data are used, how they are used, by whom, and in what order. A popular approach to methodology management has been through the use of design *flows*, which model the design process as a sequence of transfers between tools and data. A common feature provided in flow management systems such as [1], [2], [3], and [4], is the ability to store and trace *design metadata*, or information about the design, including the history of activities. In particular, [6] focussed on issues relating workflow management and schedule management. Another approach to design management is expressed in [5], which views the design process not in terms of flows, but rather as a series of design problems to be solved. This work pays particular attention to how both design actions and artifacts may be decomposed in order to simplify problem solution.

The notion of measuring, analyzing, and improving the time required to complete processes has long been a key issue in management science. Graph-based process representations and methodologies [7][8] have been developed to facilitate this. Of particular interest is the body of work that addresses techniques for managing the design of coupled systems. One representation that has been heavily used is the *design structure matrix* [12], which is essentially a nodal incidence matrix that captures input/output couplings between tasks in a design process. The design structure matrix representation has formed the basis for work such as [9], which attempts to minimize the time required to complete the process by finding an optimal ordering of the task sequences. In [13], it was shown how these tools could be used to reduce the time of a printed wiring board development cycle.

Of particular relevance to our work are the results described in [11]. One of the main contributions of [11] was to improve upon the design structure matrix to include individual task times and coupling strength between tasks, which were represented as probabilities. From this improved representation, it became possible to obtain a more accurate approximation of the execution time of a design process by evaluating the process as a reward Markov chain [10]. This notion forms the basis for our work as well, but we have expanded the model to account for external design factors, and dealt with issues of metadata collection and calibration with real design processes.

III. The Sequential Design Process Model

A *design process* may be defined as the set of activities involved in taking a design problem from an initial specification to producing a finished artifact that meets these specifications. Prior to the application of the design process, we may say that the design is in an initial state, *S*, and after the completion of the design process, when all specifications have been met, we may say that the design is in a final state, *F*. Between the initial and final states, the design process may be broken down into a sequence of fundamental, atomic operations called *tasks*. In the context of an ASIC design process, tasks may include such operations as conceptual design, schematic entry, technology mapping, and

33rd Design Automation Conference ®

DAC 96 - 06/96 Las Vegas, NV, USA

simulation, while in a software design process, tasks may include coding, compilation, and testing. For the class of design processes considered in this paper, we will assume that each task needs to be performed sometime during the process.

While it may be theoretically possible to take a design process from start to finish by executing each task exactly once in a specified order, in practice, this is rarely the case. Typically, certain tasks need to be repeated in a design process before all specifications are met. The most common causes of repetition are due to incomplete information or errors introduced early in the design process that are not detected until a later task. Thus, depending upon the outcome of a given task, the design process may make a transition to one of several possible next tasks: either a task that is further downstream and closer to the final state, or to repeat a task earlier in the process.

We refer to a design process where there exists the possibility of at least one task being repeated as an *iterative* design process. Further, if all tasks are executed one-at-a-time, the process is *sequential*. In order to model the time that it takes to complete an iterative, sequential design process, it is thus necessary to consider two things: the times that it takes to complete each individual task and the likelihood that tasks will be repeated during the process. An iterative, sequential design process may be represented by a directed graph *(V, E, S, F)*, where the vertices $V_1 \ldots V_n$ correspond to tasks, the edges $E_1 \ldots E_m$ correspond to transitions between tasks, and *S* and *F* are the initial and final states, respectively.

The information needed to determine the expected time required to complete a design process may be represented as weights on the vertices and edges, where the weight of a vertex T_i represents the duration of task V_i and the weight of an edge P_i represents the probability that transition E_i is taken at any given point in time. Note that the sum of the probabilities of all edges emanating from a given vertex must equal 1. Taken together, the set of values $\{T, P\}$ constitute the set of *performance parameters* of a design process. In general, both the durations and the transition probabilities may in practice vary with time; in order to simplify the model, however, we will regard the values of both of these sets of quantities as being static throughout the execution of the process. On the other hand, these values typically cannot be known with certainty and thus should be represented by random variables. Thus, we will regard each of the process performance parameters $\{T, P\}$ as being distributed with some mean value and variance.

Given the individual task durations and transition probabilities $\{T, P\}$, we may express the total process time as τ, where $\tau = \tau(T, P)$. The value of τ may be determined by viewing the process graph as a *Markov chain*, where each state in the chain corresponds to the completion of one task in one instance [11]. In general, the cost of the chain can be evaluated through simulation, but given the restrictions of time-invariant durations and probabilities, it has been shown in [11] that the expected value of the total cost may also be evaluated through the solution of a system of linear equations.

IV. Process Model Calibration

To effectively analyze the design process, it is crucial to accurately model both the task durations and transition rates. A basic premise of this work is that these parameters can be estimated using process metadata. In contrast to design metadata, *process metadata* describes information about the design process itself, such as how long it took to complete the process or when iterations in the process occurred. We define the extraction of performance parameters from process metadata as process model *calibration.* In this section we outline the two issues associated with the calibration procedure; the collection of the process metadata and the extraction of the process parameters.

Collecting Process Metadata

In collecting metadata, care must be taken to ensure that the data is both accurate and complete. Therefore, the most important phase of metadata collection is the development of a model that can be used as a basis for collection. The model is a set of objects that together provide an abstraction of the design process and that will ultimately be used to define containers in a database to store the metadata. With a model in place, techniques for collecting the process metadata can then be developed.

Ideally, all metadata should be collected in a manner that is transparent to the designer, so as not to interfere with the design process. In practice, however, we have found that several different techniques must be employed to ensure that the metadata captured is complete. These techniques include *passive monitors, run-time tracking forms,* and *surveys.* Passive monitors are background processes that collect process metadata automatically using routines that monitor design activity during process execution. Monitors help to ensure accuracy, because designers cannot rationalize specific design decisions or the amount of time spent on each task prior to reporting results. An example of a system that uses passive monitors is described in [3]. Complementing the passive monitors are run-time tracking forms. Tracking forms prompt the user for additional information, such as the reason for an iteration, *during* process execution. Further information may be gathered through surveys after the design process is complete. We note that while run-time tracking forms and surveys are less accurate than passive monitors, their use may be justified when automatic metadata collection is too difficult or inconvenient to implement.

Model Parameter Determination

Calibrating the process models is accomplished by using process metadata to calculate the model's performance parameters. Because we have assumed the performance parameters to be time invariant, the individual task durations are computed by finding the average time spent in each task by the designer. The transition rates are calculated as the ratio of the number of transitions for a specific path exiting a task to the total number of transitions leaving that task.

V. Experimental Results

An experiment was conducted to validate the process model using a sequential design process. The main purpose of the experiment was to illustrate how to effectively gather process metadata, and how to calibrate the process model in a small-scale environment. The experiment outlined in this section consisted of the following steps:

- Identification of the tasks and transitions that represent the process.
- Execution of the process by a group of designers and collection of a set of sample metadata.
- Calibration of the process models using the metadata.
- Application of the models to analyze the process.

Experiment Description

For this experiment, we investigated the development of a software program (postfix calculator). A simple sequential process flow was defined for this application and is shown in Fig. 1. The process consists of four tasks—conceptual design, program coding, program compilation, and program testing—with six possible transitions between tasks. For this experiment, we constrained the methodology so that the conceptual design of the problem was completed before any coding was started. This

constraint was imposed purely to facilitate capturing the time spent in that task and could be removed with improved metadata collection technology. Although the structure of the process was predefined, various execution options were offered for individual tasks. For example, two editors were available for design coding, and testing could involve either executing the program or using a debugger. The purpose of these options was to give the individuals a series of methodology choices that could be used to investigate how design decisions impacted their overall process time.

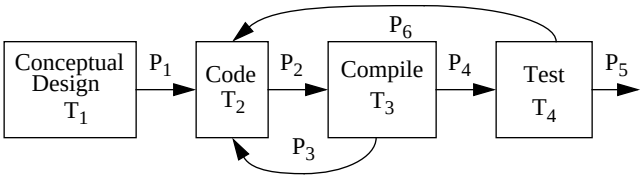

Fig. 1. - Experimental Process Flow

A group of nineteen designers participated in this experiment. During process execution both passive monitors and run-time forms were employed to collect process metadata. Passive monitors were used to track when files were created or updated in the working directory and to tag when tasks were started and completed. These monitors were implemented as background processes and were transparent to the designers. Designers executed tasks in the environment by selecting a specific task in a task window which either invoked the appropriate tool or gave the user a set of tool choices. At certain stages in the process, designers were also asked to select options from a form describing why certain tasks would be executed or what results occurred. Finally, upon completion of the design projects, each designer was asked to submit a survey designed to collect additional information not gathered by the passive monitors. This included information about the clarity of their design, their programming experience, and other the design decisions made throughout the process.

Process Calibration

Following the design exercise, the metadata was collected and tabulated to extract the amount of time spent in each of the tasks, identify when and where transitions were taken, and track the reasons for iterations for each designer. Table 1 summarizes the average and standard deviations associated with the performance parameters for one of the designers and for the group as a whole. The large standard deviation for the coding task duration resulted from the fact that in some iterations code was being developed while in others, compilation errors were simply being corrected. This variation could be reduced by modeling the initial code development and correction as separate tasks.

TABLE I. Parameter Values for Single Designer and Group

	Example Designer		**Group**	
Parameter	**Ave.**	**Std. Dev.**	**Ave.**	**Std. Dev.**
T_1 (Concept)	900	-	2735	2556
T_2 (Code)	246.4	391.2	408.8	290.5
T_3 (Compile)	5.0	4.7	8.1	5.8
T_4 (Test)	72.5	36.7	105.9	51.4
P_3	0.36	-	0.33	0.14
P_6	0.96	-	0.88	0.11

Once the parameters were determined, two tests were performed to verify that the process model provided an accurate representation of the process time. First, using the information from Table 1, a Monte Carlo simulation was run to compare average completion time generated by the simulator to the actual process completion time of the designer. The second test involved calibrating the process model for the overall group and again performing a Monte Carlo simulation to predict the average process completion time. These results are illustrated in Table 2.

TABLE II. Simulated vs. Actual Process Duration

	Simulated	**Actual**	**% error**
Designer	12554	12587	0.2
Group	8839	10075	12.2

The simulation results show for this experiment, the parameters of the process model—task durations and transition rates—accurately capture the time spent in the process for the individual designer and the group.

VI. Process Parameter Analysis

One advantage of collecting process metadata is that it can be used to perform a post-mortem analysis on the design process of individual designers. We have begun to investigate two types of analyses: first, analyzing the performance parameters associated with individual designers and then determining how different factors influenced the overall process time.

Designer Analysis

In both industrial and academic environments, it has been observed that some designers are better than others at creating high-quality designs in short periods of time. Through analysis of the calibrated task durations and transition rates, insights may be gained that can lead to improved designer performance.

Consider the designer whose performance parameters are illustrated in Table 1. Although the average durations for each task as compared to the group are lower, his overall process time was greater than that of the group because both of the iteration rates are higher. Therefore, it would seem logical that if the designer could reduce the total number of iterations, the overall process time would decrease. Since the designer spent comparatively little time in conceptual design, our initial action was to investigate the relationship between conceptual design time and testing iteration rate for the entire group. The group metadata supported the expected trend that the more time spent in conceptual design, the lower the rate. Therefore, if the designer would have spent more time in conceptual design, he could have possibly reduced overall process time. Although investigating the performance parameters alone may provide some insight into reducing the completion time, further analysis cannot be accomplished without studying how specific design factors influence the individual performance parameters.

Factor Analysis

Design factors are those characteristics associated with the process, the resources used with the process, or the design artifact itself that influence the overall process completion time. For our experiment, we identified eighteen distinct factors in the seven different areas: artifact performance, artifact quality, tool selection, experience, impasse resolution strategy, development methodology, and environment factors. These factors were identified by determining which characteristics have an influence on the overall software process and the individual tasks associated with the process.

A simple yet informative method for analyzing the influence of design factors is to investigate their relationship to the performance

parameters and identify group trends. For example, using these trends, we could further investigate why the aforementioned designer took longer on the design project than the group average. Metadata collected from the survey showed that the designer did not use a debugger when testing, and utilized class notes when solving specific programming problems. Experimental trends suggest that if the designer utilized the on-line help facility instead of his notes, his coding time would decrease. Similarly, if the designer had used a debugger when testing his code, he could decrease the testing iteration rate. Both of these improvements should cause a decrease in the overall process time.

While these findings may seem obvious to an experienced designer, they do provide valuable recommendations to those designers who are more inexperienced. These recommendations allow inexperienced designers to increase their task-relevant maturity and therefore improve their efficiency on future design processes.

VII. Conclusions

In this paper we have described a Markov-based model that can be used for measuring, diagnosing, and simulating sequential, iterative design processes. The model represents these processes using two sets of parameters, task durations and transition rates. To demonstrate the model's potential, we performed an experiment in which a group of designers worked on the same design problem using a specified design process. Passive monitors, run-time forms and surveys were used to collect metadata representing the actions of the designers during process execution and information about the designers themselves. After the metadata was tabulated, Monte Carlo experiments were run to compare the actual overall process time with the simulated time. The results showed that the Markov-based model parameters exhibit considerable promise for benchmarking iterative design processes. In this paper we also showed that the system could be used to perform a post-mortem analysis of the design process by investigating how designers could improve their design skills and how different design factors influence the overall process time. Although the experimental process was associated with software design, we believe that the same techniques could be utilized with the design of integrated circuits and extended to larger designs.

Our future work will focus on two main areas. First, using the present experiment results, we will investigate the use of sensitivity analysis to determine how the performance parameters influence the overall process time. Second, using the experience gained from this experiment, we hope to extend our model to include both concurrency and group interaction. This extension will allow us to investigate processes that are both multidisciplinary and distributed.

VIII. Acknowledgments

This effort was supported in part by an ACM/IEEE Design Automation Scholarship and NASA Research Grant NAG-1-1561.

IX. 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] J. B. Brockman, and S. W. Director, "The schema-based approach to workflow management," In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 10, October 1995, pp. 1257-1267.

[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] 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.

[5] M. F. Jacome and S. W. Director, "Design process management for CAD frameworks," In *Proceedings of the 29th IEEE Design Automation Conference*, June 1992, pp. 500-505.

[6] E. W. Johnson and J. B. Brockman, "Incorporating design schedule management into a flow management system," In *Proceedings of the 32nd IEEE Design Automation Conference*, June 1995, pp.82-87.

[7] J. J. Moder, C. R. Phillips and E. W. Davis, *Project Management with CPM, PERT, and Precedence Diagramming*, Van Nostrand Reihold, 1983.

[8] A. B. Pritsker, *Modeling and Analysis using Q-Gert Networks*, John Wiley and Sons, 1977.

[9] J. L. Rogers, "A knowledge-based tool for multilevel decomposition of a complex design problem", NASA Technical Paper 2903, May 1989.

[10] S. M. Ross, *Stochastic Processes*, John Wiley and Sons, New York, 1983.

[11] R. P. Smith and S. D. Eppinger, "A predictive model of sequential iteration in engineering design", *MIT Sloan School of Management Working Paper*, No. 3160-90-MS, 1994.

[12] D. V. Steward, "The design structure system: A method for managing the design of complex systems," *IEEE Transactions on Engineering Management*, vol. EM-28, no. 3, August 1981, pp. 71-74.

[13] C. Yeh, and R. E. Fulton, "A multidisciplinary approach for PWB design process optimization," From the *Fourth AIAA Symposium on Multidisciplinary Analysis and Optimization*, 1992, pp. 110-118.