Sensitivity Analysis of Iterative Design Processes

E. W. Johnson and J. B. Brockman
Department of Computer Science and Engineering
University of Notre Dame

R. Vigeland
Mentor Graphics Corporation

Abstract

As design processes continue to increase in complexity, it is important to base process improvements on quantitative analysis. In this paper we develop an analytical approach to analyze sequential design processes using sensitivity analysis. Two applications illustrate this approach, one involving a Pareto analysis of an ASIC design process and the other an optimization of a software design process to determine the lower bound of the process completion time.

1: Introduction

In order to remain competitive, it is critical for design houses to be able to continuously monitor and improve their design processes. Typically, suggestions for process improvements have been based upon anecdotal data and designers' perceptions of process bottlenecks. The danger of such an ad hoc approach is that costly process changes can be mistargetted, resulting in little or no improvement. To avoid such problems, process improvement decisions should ideally be based upon solid, quantitative analysis. One source of difficulty, however, is a lack of analytical tools. In [2], a methodology was presented for monitoring, modeling, and simulating sequential, iterative design processes. In this paper, we extend the analytic component of that work with the development of an efficient approach to sensitivity analysis. Through sensitivity analysis, it is possible to identify those features of a design process that have the greatest impact on overall process completion time which in turn can assist in making informed decisions regarding changes.

2: Modeling Process Completion Time

A *design process* may be defined as a set of activities that take a design problem from an initial specification to a finished artifact that meets those specifications. In between, the design process may be broken down into a sequence of fundamental, atomic operations or *tasks*. The coupling or *transitions* that occur between tasks represent the flow of information that occurs as the process is executed. These transitions can either push the design further toward the final state or cause the design to iterate back toward the initial state to repeat a task or set of tasks. To quantify design process time, two sets of parameters are needed: the amount of time spent in each of the individual tasks, and the likelihood that tasks will be repeated [2]. Graphically, the design process can be represented using a directed graph where the nodes correspond to the tasks and the arcs signify the transitions between tasks which is illustrated in the example below.

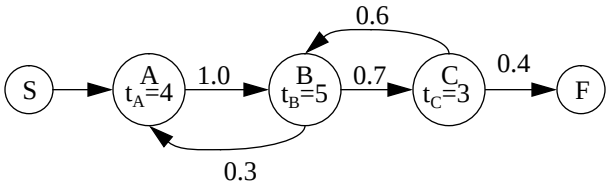

Figure 1. Example Design Process

Given both the process structure and the parameter values, completion time can be estimated either using systems that employ Monte Carlo simulations [4][5] or through analytical methods [6]. The analytic completion time model in [6] serves as a basis for this work, and is summarized below.

The analytical technique in [6] utilizes the fact that design process execution can be represented as a reward Markov chain where each state in the chain represents an instance when a task is executed. If the reward for each state is the amount of time needed to perform the task, then the overall process time can be found by determining the reward for the overall process execution. An efficient technique for finding the overall reward involves breaking the chain into a series of stages where the boundaries of each stage correspond to the location in the chain when

ICCAD '96
1063-6757/96 $5.00 © 1996 IEEE

each task is executed for the first time. The overall process time is then computed by summing the time remaining from the beginning task in each stage. For the example in Fig. 1, this means taking the sum of the remaining time in Stage 1 from Task A, Stage 2 from Task B, and Stage 3 from Task C. Algebraically, the remaining time spent in a stage from each task can be represented as a set of linear equations. For the process shown in Fig. 1, the set of linear equations for Stage 3 are:

$$r_{3A} = r_{3B} + 4.0$$
$$r_{3B} = 0.3r_{3A} + 0.7r_{3C} + 5.0\,.$$
$$r_{3C} = 0.6r_{3B} + 3.0$$

In general, the equations can be rewritten in matrix form,

$$PR = T\,, \qquad \text{(EQ. 1)}$$

where P is a n x n matrix describing the transition probabilities in the stage, T is the duration of each task executed in the stage and R is the vector of remaining times in the stage from each task. For our example, the matrix format for the third stage is

$$\begin{bmatrix} 1.0 & -1.0 & 0.0 \\ -0.3 & 1.0 & -0.7 \\ 0.0 & -0.6 & 1.0 \end{bmatrix} \begin{bmatrix} r_{3A} \\ r_{3B} \\ r_{3C} \end{bmatrix} = \begin{bmatrix} 4.0 \\ 5.0 \\ 3.0 \end{bmatrix}.$$

Once in this format, Gaussian elimination can be used to compute the remaining time from the beginning task in the each stage. The overall process time is then found by taking the sum of those results.

In some design processes there may be tasks that are not a part of the main information flow but that are executed only if an iteration is taken. For example, when a design error occurs, an extra task might be placed in the process to analyze or correct that error. We define these type of tasks as *conditional tasks*, while the tasks that are a part of the main information flow will be called *required tasks*. The previously described analytic model can be extended to processes that include these conditional tasks by modifying how the tasks and transitions are placed in matrix. Conditional tasks appear first in the matrix followed by the required tasks as shown in Fig. 2,

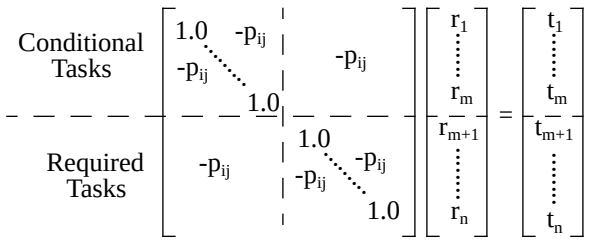

Figure 2. General Equation Format

where of n separate tasks, m are conditional and $n-m$ are required. The off-diagonal elements, $-p_{ij}$, have non-zero values where coupling between tasks occurs. When a process contains both conditional and required tasks, completion time can be calculated using the same algorithm as outlined earlier. It is important to note, however, that stage boundaries in the Markov chain occur only when required tasks are executed for the first time.

3: Calculating Process Time Sensitivities

In general, sensitivities provide an understanding of how variations in system parameters influence system performance, and therefore can be used to find optimal systems. One of the main advantages of analytical solutions over simulation is that they provide closed-form expressions for subsequent sensitivity analysis which represents a substantial savings over sensitivities computed via finite-differences.

Given the analytical solution for calculating process completion time, we can efficiently determine how sensitive the overall time is to changes in the individual task durations, t_i, and transition probabilities, p_{ij}. Consider the system of linear equations that describe the expected time remaining in a stage as shown in Eq. 1. A direct method for computing the sensitivity of all components of R with respect to a specific parameter h is to differentiate the equation with respect to h,

$$P\frac{\partial R}{\partial h} + \frac{\partial P}{\partial h}R = \frac{\partial T}{\partial h}\,, \qquad \text{(EQ. 2)}$$

rearrange,

$$T\frac{\partial R}{\partial h} = -\left(\frac{\partial T}{\partial h}R - \frac{\partial W}{\partial h}\right), \qquad \text{(EQ. 3)}$$

and then solve for the sensitivities, $\partial R/\partial h$.

To compute the process completion time sensitivities, the direct method is first applied with the performance parameters which yields the following equations,

$$\frac{\partial R}{\partial p_{ij}} = -P^{-1}\frac{\partial P}{\partial p_{ij}}R \qquad \frac{\partial R}{\partial t_i} = -P^{-1}\frac{\partial T}{\partial t_i} \qquad \text{(EQ. 4)}$$

Then, the overall sensitivity with respect to each parameter is determined by the summation of the stage sensitivities.

When the sensitivity of one performance output with respect to various parameters is of interest, the *adjoint* sensitivity method [1] can be used to calculate sensitivities in a more computationally efficient manner. Computing process completion time sensitivities using the adjoint method is presented in [3].

4: Pareto Analysis on an ASIC Design Process

One method for identifying areas of improvement is

through Pareto analysis. A Pareto analysis ranks the sensitivities to determine which parameters have the greatest impact on a specified performance. For design processes, a Pareto analysis is useful because it determines which transitions and individual task durations have the most influence on overall process completion time.

Consider an ASIC design process employed by an actual design house as illustrated in Fig. 3. Prior to analysis, a group of designers was surveyed to determine their design experiences over a dozen different projects. The information was then normalized to obtain the average task duration times and transition probabilities. Using the methods described in Section 2, the completion time for this process was estimated at 156.2 days or 7.1 months, assuming 8 hour days and 22 working days per month. Performing a Pareto analysis of the ASIC process reveals which transitions and task durations have the largest affect on process time. The five transition probabilities and task durations with the largest effect are shown in Table 1 and Table 2.

Transition	Sensitivity
From Task 5 to Task 12	1.43
From Task 7 to Task 12	0.84
From Task 8 to Task 12	0.81
From Task 3 to Task 2	0.66
From Task 11 to Task 13	0.49

Table 1. Pareto Analysis of Transitions

Table 1 shows that the transition $p_{5,12}$, which occurs after the "Simulation and Static Timing" task, has the largest effect on the overall process time. Changing this probability by one percent would cause the overall process time to change by almost 12 hours. This iteration loop is the innermost of the three iteration loops. Intuitively, since more tasks would have to be repeated, one would suspect

Duration	Sensitivity
Task 4	2.34
Task 5	2.34
Task 2	2.00
Task 3	2.00
Task 12	1.34

Table 2. Pareto Analysis of Tasks

that the outermost loop, caused by the $p_{7,12}$ transition, would have the largest impact on the overall process time. The larger probability associated with $p_{5,12}$, however, causes the larger sensitivity. In fact, the structure of the matrix alone is not sufficient to determine which task or transition has the greatest impact on the process time. The dominant sensitivity is highly dependent on the task durations and transition probabilities themselves. It is also important to note that transition probabilities continually change between projects and that the $p_{5,12}$ transition may not always have the most affect on the overall process time. This illustrates the importance of sensitivity analysis in helping to identify which tasks and transitions are critical to the overall process time and that re-employing sensitivity analysis is necessary for continuous process improvement.

5: Optimization of a Software Design Process

In general, iteration is required in a process when an error introduced in one task is not detected until a later task, causing a task or set of tasks to be repeated until the error is removed, at least to a degree sufficient for the quality and performance of the artifact being designed. The sources of error in design vary widely between design processes. Errors may result from improper implementation or interpretation of design specifications.

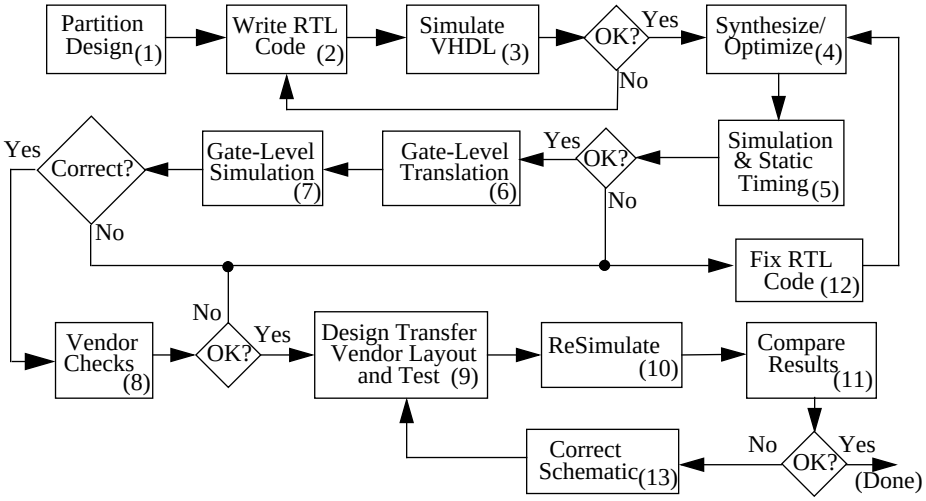

Figure 3. ASIC Design Process

It is possible, in many cases, to reduce the likelihood that tasks need be repeated by spending additional resources when they are originally performed. For example, spending more time up front in the conceptual design of an artifact can reduce the chances of rework at many stages of the process. There reaches a point, however, when more resources may not decrease and in fact cause an increase in process completion time. An optimization, based on sensitivity analysis, may help determine this point.

As an example, a group of student designers were asked to create a software program that met a given set of specifications (postfix calculator) [2]. The students were asked to perform the experiment through a common environment consisting of passive monitors which monitored their design activities. The software design process associated with the experiment is shown in Fig. 4.

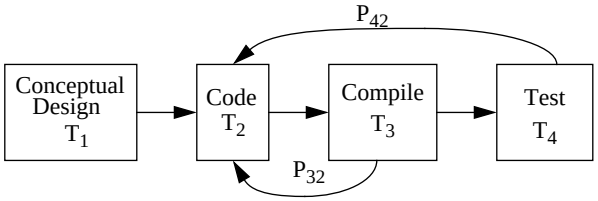

Figure 4. Simplified Software Process Flow

The two transitions that cause iteration, p_{32} and p_{42}, occur because of errors introduced in the "Coding" task by the designer. It was assumed in this experiment that the conceptual design of the problem was completed before any coding was started. Therefore no transitions causing iteration back to the conceptual design task existed. For this optimization problem, the transition probabilities p_{ij} due to the error, were assumed to be of the form,

$$p_{ij} = e^{-kt_j},$$

where k summarizes the design factors and t_j represents the duration of the task that introduced the error. This form was determined intuitively based on the observation that the more time spent in a task, the less likely an error will be introduced.

Given the functional form for relating task duration to transition probability, the value of the k parameters for individual designers calculated from design process metadata were:

$$p_{32} = e^{-0.0042t_2} \qquad p_{42} = e^{-0.00017t_2}$$

An optimization of the form,

$$\min_{t_i} T_P(t_i, p_{ij}) \qquad \text{(EQ. 5)}$$

where T_P is the total process time,

t_i is the individual task times and,

p_{ij} is the probability between tasks i and j,

was then performed to determine the amount of time that the designer should have spent in each task in order to minimize the overall completion time. Because the code and compile tasks did not introduce errors and the conceptual design task was first completed, their durations remained unchanged. The coding task duration, however, was driven to a value that would minimize the overall process time. The results show that by increasing the average coding time from 246 seconds to 1168 seconds, the overall completion time would decrease by 37 percent.

It is important to note that no causality is implied by this result; clearly *how* a designer spends the increased coding time is critical to realizing any decrease in overall time. A more appropriate interpretation of this result is to view it as a lower bound on design time, based on the empirical observation of the relationship between task durations and the likelihood of error.

6: Conclusions

In this paper we have shown how sensitivity analysis can be used to identify those mechanisms in design processes that have the greatest impact on design time. Because our approach is based on the use of an analytic model of the process, rather than Monte Carlo simulation, sensitivities may be calculated more efficiently. Two applications illustrated the effective use of sensitivity analysis. In the first application, the sensitivities were ranked in a Pareto analysis of an ASIC design process. The results of the analysis showed that it is not always intuitive to determine which parameters affect process time. The second application involved using sensitivities as gradients in an optimization problem to identify a lower bound on process time for a software design process.

7: References

[1] S. W. Director and R. A. Rohrer, "Automated Network Design, The Frequency Domain Case," *IEEE Transactions on Circuit Theory*, vol. CT-16, pp. 330-337, August 1969.

[2] E. W. Johnson, L. Castillo, and J. B. Brockman, "Application of a Markov Model to the Measurement, Simulation, and Diagnosis of an Iterative Design Process", In the *Proceedings of the 33nd IEEE Design Automation Conference*, June 1996, pp. 185-188.

[3] E. W. Johnson and J. B. Brockman, "Improving Sequential Design Processes using Sensitivity Analysis," Department of Computer Science and Engineering Technical Report #96-1, University of Notre Dame, 1996.

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

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

[6] 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.