Journal of Engineering Design

Investigating product development process reliability and robustness using simulation

Ali A. Yassine

University of Illinois at Urbana-Champaign, USA

First Published on: 14 November 2006


To link to this article: DOI: 10.1080/09544820601011690

URL: http://dx.doi.org/10.1080/09544820601011690

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.informaworld.com/terms-and-conditions-of-access.pdf

This article maybe used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.
Investigating product development process reliability and robustness using simulation

ALI A. YASSINE*
University of Illinois at Urbana-Champaign, USA

This paper uses the design structure matrix to model and simulate the performance of product development processes. Although simulation is a powerful tool for analysing process performance, its ability is limited by the availability of accurate input information. Design structure matrix simulation requires process data that is hard to assess or estimate directly from development participants. In this paper, we propose a methodology that allows a more practical estimation of an important simulation input parameter: rework probability. Furthermore, we show how this assessment method (combined with simulation) allows managers to evaluate process improvement plans based on two resultant process measures: reliability and robustness. The method is illustrated with a real application from the automotive industry.

Keywords: Design structure matrix; Product development; Process simulation; Process re-engineering

1. Introduction

Complex product development projects are usually managed by mapping them through various kinds of project flowcharts and diagrams (i.e. Gantt charts, Critical Path Method (CPM), probabilistic evaluation and review technique (PERT), Integrated Definition (IDEF) methods, etc.) that attempt to capture and manage process complexity (Kusiak 1999, Austin et al. 2002). While many of these methods are capable of illustrating timing, information flows, and task interdependencies, they fall short of enabling managers to effectively model, and gain deeper understanding of, task interdependencies and iteration in the process (Austin et al. 1996, Eppinger 2001). For instance, CPM and PERT methods only work with acyclic graphs. On the other hand, IDEF0 models are most suitable to represent atemporal information flows. As the network of tasks in the engineering design process is usually cyclic and time-based, CPM, PERT, and IDEF0 methods become insufficient.

The design structure matrix (DSM) methodology provides a means to model and manipulate iterative tasks and multidirectional information flows (Steward 1981, Eppinger 1991). The DSM allows complex processes to be illustrated and modified through graphical and numerical analyses in a single manageable format. Using the DSM methodology to study development processes enables graphical representation of how tasks and information flows affect other groups of tasks, where potential issues lie, and insight about how they may be
resolved (Browning 2001, Yassine and Braha 2003). However, DSM models, as they stand, do not include the duration of tasks, the impact of iteration or rework, and consequently do not provide a time line or estimate for the project duration. As a result, simulation techniques for design processes in general, and DSM models in particular, started to emerge in process and project management literature (Baldwin et al. 1999; Browning and Eppinger 2002, Cho and Eppinger 2005, Abdelsalam and Bao 2006).

Recent literature points to the potential and success of simulation techniques in managing development processes (Adler et al. 1995, Jin and Levitt 1996). However, few studies have been conducted as compared with manufacturing processes, for example. Perhaps one of these first attempts to handle feedback relationships, account for iteration, and enable simulation-based analyses was the general evaluation review technique (Wiest and Levy 1977). (The general evaluation review technique is an extension to the popular project management technique PERT.) A paper by Adler et al. (1995) describes the use of discrete event simulation to study product development performance in companies pursuing multiple, concurrent, non-unique product development projects. The simulation allowed them to identify bottleneck activities and several development process characteristics. In a similar venue, Baldwin et al. (1999) also used discrete event simulation to manage the design process in building construction projects. Finally, Browning and Eppinger (2002) and Cho and Eppinger (2005) used simulation based on a DSM representation of development projects. Both simulation models revealed several interesting process characteristics and performance measures including expected project duration, cost and risk.

Although all these models emphasize the fact that simulation can be a powerful tool for analysing process performance, their ability is limited by the availability of accurate input information. These models require development data that are hard to assess or estimate directly from process participants; such as estimates for iteration or rework probability. However, none of these papers describe reliable ways of arriving at these probabilities and merely assume that it is possible to obtain these measures in real-world project environments.

Typical probability assessment questions normally include a direct approach by asking participants, for example, ‘what is the probability of repeating your work if such and such piece of input information changes?’ Many participants would find it difficult to provide a specific numeric value (Tversky and Kahneman 1981). However, it could be less stressful if they were asked to provide a probability range, or even to select from a multiple choice of possible values/ranges. Based on our interview experiences, we found that the qualitative format (by choosing low, medium, or high, for example) was the easiest to solicit probability estimates from development participants.

In this paper, we propose a methodology that allows practical estimation and assessment of rework probabilities. Similar assessment procedures were suggested in the decision analysis and system dynamics literature. For example, Merkhofer (1987) and Shephard and Kirkwood (1994) present an extensive interview procedure to solicit expert probabilities and value judgements necessary to conduct decision analysis. Similarly, Ford and Sterman (1998) acknowledge the fact that system dynamics modellers face difficulties in eliciting and representing expert knowledge so that useful models can be developed. Consequently, they developed an elicitation method that modellers use when interviewing experts for tacit process knowledge. Our proposed assessment technique is greatly influenced by this line of research, but is tailored to focus on issues related to the development of the DSM and the corresponding rework probabilities.

Finally, the paper discusses how this assessment method along with simulation results allows managers of development processes to evaluate re-engineering plans based on two resultant process measures: reliability and robustness. We measure reliability in terms of process duration variance. The less variability in the duration of a development process, the
more reliable that process is. Robustness, however, deals with the ability of a process to absorb design changes. A robust process is the one with a duration that is insensitive to changes in design information.

The rest of the paper proceeds as follows. Section 2 presents an overview of the DSM method and the main techniques used in analysing the matrix: partitioning, tearing, and simulation. In section 3 we present a three-phase subjective assessment procedure for assessing rework probabilities used in DSM simulations. We demonstrate the utility of this procedure with a real application of an automotive hood development process in section 4. In section 5 the example is used to show how the simulation and the subjective assessment proposed in this paper can be used to draw insights about ways to streamline and re-engineer existing development processes. Finally, we present the summary and conclusion of the paper in section 6.

2. The DSM method

The DSM is essentially a node–node interaction matrix that captures the existence, type and other attributes of relationships between system (or process or project) entities such as engineering design tasks (Steward 1981, Eppinger 1991). Figure 1a shows a simple DSM. In a DSM the names of the tasks are listed down the side and across the top, in the same order. Interactions are marked with ‘X’ or ‘•’. Such marks are often replaced with codes to indicate type or strength of interactions (Pimmler and Eppinger 1994). Marks above the diagonal feed information forward, while marks above the diagonal feed information back. The diagonal elements of the matrix do not have any interpretation in describing the process (or process), so they are usually either left empty or blacked out.

These DSMs are usually captured by conducting a series of structured face-to-face interviews with domain experts (i.e. product designers), although it is not uncommon to attempt building a DSM from standard documented design procedures or pre-existing IDEF models. However, past research (Whitney et al. 1999) has shown that written documentation in companies fails to capture more than a small fraction of the information and knowledge that is used during product development. The rest is in people’s heads. Of that which is documented, the

![Figure 1. DSM representation of development processes.](image)
majority concerns individual components, while the missing elements address system issues involving interactions and mutual constraints. Missing knowledge at the system level is particularly damaging to product design and development because it is hard to detect and is the main source of expensive design iteration.

2.1 DSM partitioning and tearing

The DSM can be manipulated by swapping the location of rows and columns, a process called partitioning, in order to arrive at a streamlined information flow and task execution order (Steward 1981; Yassine et al. 1999). In this streamlined DSM (i.e. partitioned DSM), the relationships between the tasks become evident, as shown in figure 1b. Task C, for example, requires information only from task B, so this pair constitutes a sequential design task relationship. Neither tasks A or K require information from each other, so they can be performed in parallel. Finally, tasks (L, J, F, and I) are involved in a cyclic information flow as indicated by the feedback marks contained within the square block drawn around this set of tasks (we refer to this set of tasks as an iterative block). Because of this coupling information structure, none of the tasks can have complete information before beginning. They must take an iterative approach. They each will begin with preliminary information, complete their design tasks using that preliminary information, provide their results to each other, and repeat the process.

For complex development processes, it is highly unlikely that simple row and column manipulation will result in a lower triangular form (Yassine et al. 2000). In this case, DSM reordering will not completely eliminate feedbacks, but minimize them and moves the rest as close as possible to the diagonal (this form of the matrix is known as block triangular). In doing so, fewer development tasks will be involved in iteration resulting in a faster and more predictable (i.e. reliable) development process.

Tearing is the process of choosing the set of feedback marks that if removed from the matrix (and then the matrix is re-partitioned) will render the matrix lower triangular. The marks that are removed from the matrix are called ‘tears’. Identifying those ‘tears’ that result in a lower triangular matrix is analogous to identifying the set of assumptions that need to be made in order to start design process iterations when coupled tasks are encountered in the process. Having made these assumptions, subsequent iteration will be required to check or revise them.

2.2 Numerical DSMs

In binary DSM notation, a single attribute is used to convey relationships between different system elements; namely, the ‘existence’ attribute, which signifies the existence or absence of a dependency between the different elements. Compared with binary DSMs, numerical design structure matrices (NDSM) could contain a multitude of attributes that provide more detailed information on the relationships between different process elements. An improved description/capture of these relationships provides a better understanding of the development process. As an example, consider the case where task B depends on information from task A. However, if this information is predictable or has little impact on task B, then the information dependency could be eliminated. Binary DSMs lack the richness of such an argument. Some of the attributes that can be used are as follows:

- **Level numbers.** Steward (1981) suggested the use of level numbers instead of simple ‘X’ marks. Level numbers reflect the order in which the feedback marks should be torn. Level numbers range from 1 to 9 depending on the engineer’s judgement of where a good estimate, for a missing information piece, can be made.
• Importance ratings. A simple scale can be constructed to differentiate between different importance levels for the ‘X’ marks. As an example, we can define a three-level scale as follows: 1 = high dependency, 2 = medium dependency, and 3 = low dependency (Pimmler and Eppinger 1994). In this scenario, we can proceed with tearing the low dependency marks first and then the medium and high in a process similar to the level numbers method, above.

• Probability of repetition. This number reflects the probability of one activity causing rework in another. Upper-diagonal elements represent the probability of having to loop back (i.e. iteration) to earlier (upstream) activities after a downstream activity was performed. Lower-diagonal elements represent the probability of a second-order rework following an iteration (Browning and Eppinger 2002). Partitioning algorithms can be devised to order the tasks in this DSM such that the probability of iteration or the project duration is minimized.

This paper is concerned with the estimation of the last DSM measure since it constitutes the hardest to obtain input for simulating a development process that involves iteration, as will be discussed next.

2.3 DSM simulation

Few DSM-based simulation models (and tools) already exist in the literature (Baldwin et al. 1999, Browning and Eppinger 2002, Zhuang and Yassine 2004, Cho and Eppinger 2005, Abdelsalam and Bao 2006). Although all these models are mainly aimed at determining the process completion time and cost for a given task arrangement (i.e. process architecture), some have an added component for determining the optimal architecture also (for example, Zhuang and Yassine 2004, Abdelsalam and Bao 2006) by testing (i.e. calculating time and cost) various architectural arrangements. However, if we focus on the simulation (and not optimization) component, then minor differences would only exist between the various DSM simulation models. The core difference lies in two aspects of the simulation: the sampling of task durations from known distributions and the modelling of the dynamic progress of the project (or process). For duration sampling, some use, for example, Monte Carlo sampling and others defend the use of Latin Hypercube sampling (LHS) techniques (McKay et al. 1979). For modelling the progress of a project, some use discrete-event simulation and others prefer to use simple Monte Carlo simulation (Law and Kelton 1991). In this paper, we use the simulation models described in Browning and Eppinger (2002) and Cho and Eppinger (2005) to demonstrate the proposed assessment and analysis procedure due to their availability and ease of use. Both software tools are Excel add-ins and are downloadable for free from the DSM website (http://www.DSMweb.org).

Aside from the detailed differences that are inconsequential to this paper, all DSM-based simulation models work as follows. First, as mentioned earlier, they all quantify a process configuration’s expected duration/cost and variance. Variances in duration and cost are largely attributed to the number of iterations required in the process and their scope. Since task rework may or may not occur (depending on a probabilistic check of feedback marks in the DSM), the DSM simulation model treats iterations stochastically, with a probability of occurrence depending on the particular package of information triggering rework.

The model characterizes the design process as being composed of activities that depend on each other for information. Changes in information potentially cause rework. Thus, rework in one activity can cause a chain reaction through supposedly finished and in-progress activities. Activity rework is a function of the probability of a change in inputs and the impact of that change.
As input, a DSM simulation model requires a binary DSM base model and some additional data. For each activity interface (i.e. marks in the DSM), the model requires an assessment of the probability of a typical change in the data causing rework for a dependent activity and the impact of that rework should it occur. Impact values are percentages of an activity’s initial duration. Activity duration and cost are random variables, represented by triangular distributions using three-point estimates: best duration, likely duration, and worst duration.

3. Rework probability assessment

In this section, we describe a process to subjectively assess the rework probabilities, which are necessary to run the simulation discussed in the previous section. Assessing reasonable rework probabilities is challenging. Our experience indicates that engineers and designers are generally uncomfortable providing direct probability assessment. Moreover, estimates provided vary widely between respondents for the same rework probability. There is a need to use a more reliable and indirect method to solicit these rework probabilities (see earlier paragraph on typical probability assessment questions in section 1). We propose a three-stage procedure:

1. Subjective assessment. Define two independent constructs (dimensions): information variability (IV), and task sensitivity (TS).
2. Mapping and calibration. The values of those two dimensions are mapped to a probability space. They are then correlated to probabilities using some proportionality constant based on one or more known criteria, such as project duration, cost, or performance.
3. Validation. Confirm the above resultant probabilities.

3.1 Subjective assessment of IV and TS

Off-diagonal elements in our NDSM are called ‘task volatility’ (TV). (Marks above the diagonal represent first-order rework, and marks below the diagonal represent the potential for second-order rework; Browning and Eppinger 2002.) TV describes the volatility of dependent tasks (located in the rows) with respect to changes in information from input tasks (located in the columns). Because TV describes volatility of a task with respect to an input task, it implies a probability that the dependent task will be reworked to some extent. This number is located in the matrix at the intersection of the row of the dependent task and column of the input task. TV is the product of two components: IV and TS.

3.1.1 Information variability. IV describes the likelihood that information provided by a task would change after being initially released. (Ideally, this value would be independent of the person performing the task. We also assume that this estimate is based on initial (i.e. first-time) release of information. Both assumptions were brought to the author’s attention by one of the reviewers.) Since IV is associated with the stability of a particular task’s information, each input task has its own IV value. That is, the information from a particular task has its own probability of changing. For example, if a task is concerned with the estimation of loads that would be applied onto a latching mechanism and these loads are historically known to be stable or lie within a narrow margin, then the information released by this task is pretty stable and the probability of information changing at later stages of the development process is also low.
Information variabilities are located along the bottom of the matrix and correspond to the task in that column (see later figure 5). It is difficult, if not impossible, to come up with a universal objective measurement scale for information variability to be used in all product development situations. We therefore construct a discrete, subjective measurement scale for this measure. For further details on constructed attributes see Keeney (1992). The estimated variability of information provided by a task is arbitrarily categorized in three levels, each having a numerical value, as shown in table 1.

### Table 1. Levels of information variability.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Likelihood of change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Stable</td>
<td>Low</td>
</tr>
<tr>
<td>2</td>
<td>Unknown</td>
<td>Medium to high</td>
</tr>
<tr>
<td>3</td>
<td>Unstable</td>
<td>Very high</td>
</tr>
</tbody>
</table>

3.1.2 **Task sensitivity.** TS describes how sensitive the completion of a dependent task is to changes or modifications of information from an input task. Each task’s sensitivity to changes in information from a particular input task varies. Thus, TS depends on the level of dependency between two particular tasks. For example, while the gear shaft diameter is highly sensitive to changes in load requirements, the latch stiffness is pretty insensitive to large changes in loads. So, the ‘design shaft’ TS is high relative to the ‘estimate load’ task, but the ‘design latch’ TS is low relative to the ‘estimate load’ task.

Table 2 describes the three subjective levels of TS developed using the techniques for constructing subjective attributes as described in Keeney (1992). The TS values are not shown in the later DSM of figure 5; instead, we directly entered the product of the TS values assessed with the corresponding IV values for each task.

Table 2. Levels of task sensitivity.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Dependent task is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Low</td>
<td>Insensitive to most information changes</td>
</tr>
<tr>
<td>2</td>
<td>Medium</td>
<td>Sensitive to major information changes</td>
</tr>
<tr>
<td>3</td>
<td>High</td>
<td>Sensitive to most information changes</td>
</tr>
</tbody>
</table>
Table 3. TV values, their significance, and proposed strategies.

<table>
<thead>
<tr>
<th>TV value</th>
<th>Description</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2</td>
<td>Dependency is weak Low risk of rework</td>
<td>Feedback and forecast information may be used, especially if it promotes process robustness</td>
</tr>
<tr>
<td>3, 4</td>
<td>Dependency is moderate Moderate risk of rework</td>
<td>Avoid using forecast and feedback information where possible</td>
</tr>
<tr>
<td>6, 9</td>
<td>Highly sensitive to change High risk of rework</td>
<td>Task sequence is critical to process reliability. Do not use forecast and feedback information</td>
</tr>
</tbody>
</table>

3.2 Mapping and calibration

Since TV represents a probability that some amount of rework will occur, it is necessary to calibrate the 1–9 scale to rework probabilities for use in the simulation. Several measures such as process cycle time and/or process development cost can be used to calibrate the model. For example, if the development process for a given product typically involves approximately a cycle time of $T$ days, then we can correlate the rework probabilities for the process model such that the simulations provide an average duration of $T$ days (provided that there is confidence in the information variabilities and task sensitivities data). This is accomplished by simulating the process over various probabilities to find a proportionality constant that scales the range of TV values to a sensible range of rework probabilities. To do this, each task’s volatility in the matrix is assigned a probability from 0% to some maximum value $P$. The probability assigned to each task volatility value is based on the magnitude of that task volatility. That is, the maximum probability is assigned to the highest TV value and the probabilities are proportionally decreased with decreasing TV values. The maximum probability values are varied for each simulation, and process durations are recorded. Figure 2 illustrates average process durations associated with various trial values for the maximum probability.

From figure 2, we can easily see that if the process under study typically consumes $T = 1000$ days, then the baseline duration correlates with a maximum rework probability of approximately 52%. Therefore, this is the maximum rework probability which will be used in all subsequent process simulations of this particular development process.

![Average Duration of the Development Process vs. Maximum Probability of Rework](image)

Figure 2. Average duration associated with different rework probabilities (average durations were obtained by simulating an actual process known to require approximately 1000 days).
The mapping of task volatilities to rework probabilities is provided in figure 3. It is possible to use linear or non-linear (e.g. quadratic) mapping functions as shown in figure 3. The choice depends on the kind of behaviour we would like this mapping function to have. For example, linear mapping would result in rework probabilities that increase linearly with the TV values. On the other hand, quadratic mapping allows the rework probability to increase slowly when the TV values are near 1, but as these values depart further away from 1 (towards 9) the rework probability increases more rapidly. A linear mapping for the rework probabilities was used in this paper since it provides more conservative rework probability estimates. That is, for the same TV value, the rework probability obtained by using linear mapping is always higher than the one obtained from a non-linear mapping, resulting in longer development cycle times.

3.3 Validation

It is necessary to validate the established rework probabilities by reviewing them with the subject matter experts. This is accomplished by re-interviewing these experts. They are asked the following question: ‘Is it reasonable to assume that dependent task, X is reworked p% of the time due to changes in input information Y?’ For example, ‘Is it reasonable to assume that the gear shaft diameter is revised 17% of the time due to changes in load requirements?’ The expert can confirm the probability value, thus validating it or not. If the probability is confirmed, it is not changed. If it is not accepted, the respondent is asked a series of questions to extract the reasons why the probability value is not valid. These types of questions are represented by the flowchart in figure 4. The first question attempts to identify the main drivers of change for the task being investigated (say task X, for example). If the respondent identifies y1 (for example) as one driver, then the interviewer proceeds to validate the influence of y1 on X, which was already believed to have a specific probability value based on its TV value (35% in the flowchart example). If the expert does not confirm the probability, then the interviewer attempts to update it by relating the change to either the IV or TS value. The same line of questioning is repeated for all the drivers of change for task X and all the rework probabilities in the row of task X are validated.
4. Application: automotive hood development process

A high-level description of the automotive hood development process is as follows (Zambito 2000). (Note that the time and cost data used in this application are scaled to protect confidentiality, but preserve its characteristics) Marketing acquires and aggregates consumer needs data and supplies them to product development. Product design then generates product concepts, which are evaluated for manufacturing feasibility by manufacturing and for consumer acceptance by marketing. After iterating through this phase to gain marketing, product development and manufacturing concurrence on a set of feasible concepts, product concepts are developed further until a single concept is selected. The product concept, manufacturing tooling and marketing strategy evolve to completion through an iterative process between marketing, product development, and manufacturing that ensures the latest consumer needs data will be met while manufacturing feasibility is maintained.
The DSM for the hood development process is shown in figure 5. The shaded and outlined regions illustrate subsets of tasks that are coupled, which are referred to as iteration or feedback loops. The DSM contains five iterative loops. (An iterative loop depicts a situation where earlier tasks are revoked (i.e. reworked) when new information unravels later into the development process. In a DSM this will be evident by the existence of marks above the diagonal.) While these data are typical of DSM models, this NDSM contains additional data that offer further insight into the process and are useful for making process-reengineering decisions. These include estimated duration and cost data associated with each task, as well as two-dimensional task volatility indices. For clarity, figure 5 has been annotated to identify each type of data and their location in the matrix.

4.1 Data collection for base-case analysis

Populating the hood development DSM involved collecting timing, cost, dependency and other task data from historical data and various stakeholders. A broad cross-section of stakeholders were interviewed when historical data were unavailable or needed verification. This helped eliminate single-perspective biases and ensured that the DSM accurately described the actual hood development process. Stakeholder interviews included experienced representatives from manufacturing, assembly, computer-aided engineering, computer-aided design (CAD), product engineering and product engineering management. In total, 15 personal and/or telephone interviews with 10 stakeholders were conducted. The duration of each interview varied from 45 to 90 min. The interviews were structured as follows:

- Identifying the task(s) of the hood development process that the interviewee is involved in, along with a brief summary of the task.
- Determining the nominal duration (and cost) of the task.
- Determining the rework duration (and cost) of the task.
- Determining the inputs needed to carry out the task.
- Determining the sensitivity of the task to changes in above inputs (TS values).
- Determining the variability of the task’s output (i.e. IV value).

The interviews resulted in the identification of the 43 tasks shown in the DSM of figure 5. The nominal durations and rework times (and cost) are also displayed in the figure. Most historical data were easily concurred through interviewing various stakeholders, with the exception of some lower-level task durations. This issue arises in large part because many lower-level task durations are not tracked independently, but rather are included as part of a larger set or cycle of tasks. An example of this is task 10, developing an initial attachment scheme. This task is an important part of developing a design concept; however, its duration is not tracked independently. To resolve this issue, experienced product engineers who have developed attachment schemes on many hood systems were independently asked to provide durations for this task, and consequently we were able to solicit the three-point estimates (required for the triangular distribution) for this task’s duration. The duration shown in the matrix represents the average duration for the tasks.

After identifying the tasks and their durations in the first part of the interview, the interviews progressed into determining the inputs required to perform the task and the sensitivity of the task to changes in the identified inputs (i.e. TS values). Each interviewee was asked to qualify the sensitivity of their task (to changes in each particular input) based on the scale presented in table 2.

Finally, the interviewees were asked about the variability of their task’s output and the reasons behind variation (i.e. IV value). Discussions relating to task durations immensely
Figure 5. DSM for hood development process.
helped us in assessing the variability. For example, responses (from different experts) relating to task 10 duration varied from 2 to 20 days. Follow-up interviews and clarifying questioning enabled a deeper understanding of why the duration was perceived to vary so widely. Typical clarifying questions were: ‘Under what circumstances could it take 2 (or 20) days to develop an initial attachment scheme?’ and ‘How typical are these circumstances?’ In many cases, the longer durations were found to be due to rare events. In task 10, for example, the 20-day duration occurs if a completely new attachment strategy was established. The 2-day duration occurs when a previously proven attachment scheme is used, which was said to be common. This understanding allowed the interviewed experts to decide on an IV value of 3 from table 1 and also estimation of the three-point task duration.

4.2 Re-engineering the hood development process

For improving the base hood development process, we focus on the first and biggest coupled DSM block: ‘Concept Generation, Development, and Preliminary Verification’ (block 1 in figure 5 consisting of tasks 2–24). This iterative block is mainly due to the coupling of two design activities: ‘Develop Initial Design Concept’ (task 7) and ‘Evaluate Functional performance’ (task 24). Details of this reengineering procedure are found in Yassine et al. (2000) and Zambito (2000).

Inspecting the DSM in figure 5 shows that testing occurs well after the preliminary CAD model has been developed (task 7). The preliminary CAD model consists of early design concepts for the outer panel, inner panel, attachment scheme and reinforcements. However, these tests provide the first indication of the structural performance of the hood subsystem. If the subsystem clearly passes the initial tests, it is optimized to reduce weight and cost (typically by decreasing material gauge, which is a direct feedback to task 2; ‘Select Materials for all System Components’). If the subsystem fails any of these tests, the structural components (inner panel and reinforcements) are revised in an effort to resolve these issues. This may include increasing the size or adjusting the location of the inner panel’s structural beams, increasing material gauge and/or material type of various components, or a combination of these actions. The preliminary CAD model (developed in task 7) is then reworked to reflect the changes. The finite element analysis analyst then updates the structural model based on the revised CAD model and reruns the tests. If the subsystem fails, the entire process is reiterated until the subsystem passes the suite of tests. Thus, the analysis process is in a build then test sequence, which is largely susceptible to significant rework.

Our re-engineering approach is focused on exploring alternate ways of conducting the development activities in this iterative block using an improved process. The new process uses the minimum amount of data needed to evaluate the hood functional performance early in the process. Figure 6 describes how the original tasks map to those associated with the new process. Task 7 is decomposed into two tasks. Task 9 is conducted using the hood generator (an expert system that can quickly create an approximate geometric model suitable for Finite Element Analysis). Task 8 is the activity of developing the design intent CAD model (it is no longer the preliminary CAD model because it is requirements driven). Task 24 is redefined to include two additional tasks: develop structural requirements (through topology optimization), and develop conceptual design strategy (with the output of the topology optimization). Task 24 maps almost directly to task 10. However, the term ‘evaluate’ is replaced with ‘verify’, since this analysis is a verification rather than an evaluation process. The new tasks are inserted into the first block of the original development process in the sequence given by their new task numbers, and all interdependencies are verified.
5. Process management and re-engineering insights

5.1 Measuring process performance

Running the simulation with the maximum rework probability set to zero (0%) provides the shortest possible cycle time associated with these tasks durations and process structure. This analysis results in a mean cycle time of 700 days compared with the same process whose baseline duration is 1000 days.

While achieving the optimal performance (i.e. minimal duration) may be unlikely, it is useful to understand how well a process is capable of performing in the absence of iteration. An obvious benefit of this determination is that it allows managers to gauge the validity of proposed process improvement targets. Another is that it enables cost/performance trade-off analyses. That is, managers can use this information to assess the sacrifices that are required to achieve optimal (or near optimal) performance. For example, it might be possible to remove a significant amount of rework by assigning redundant resources to improve the quality or feasibility of assumptions through parallel validation testing, historical data analysis, or experience. Another reliability improvement might be gained through the purchase of a new technology that provides assumptions that are more reliable; thus, limiting the ranges of various inputs (i.e. reducing information variability). The costs of these efforts could be weighed against the benefit of having near optimal process performance.

5.2 Process reliability and process robustness

As stated earlier, process reliability deals with the amount of variance associated with the duration of a process. Furthermore, process robustness is concerned with the ability of a process to absorb design changes. In most cases, there is an inherent trade-off between process reliability (controlling the variability of a process) and process robustness (allowing value-added iteration). For example, a purely sequential process has no potential for feedback, and thus will reliably produce an expected process duration and development cost. While it is typically infeasible to develop complex products using a purely sequential or parallel process, such a process would not be very robust. That is, a completely reliable process offers little
robustness because it is incapable of incorporating potential changes in required input values for some of the tasks.

The probability curves shown in figure 7 would provide useful insights into the reliability and robustness of a process. The figure shows two sets of simulations: the original hood development process and the corresponding re-engineered process. The reader may think of it as similar to the original process (shown in figure 5) but with few process changes resulting in the deletion of some old tasks and addition of new ones as described in section 4.2. (Indeed some of the information dependencies have been also altered in the new process.) Process reliability is described through the curves by the range of probable durations (the error bars in figure 7). That is, a curve with a wide range of variation could produce a wide range of cycle times for the same level of rework probability. A tight range would produce a more reliable cycle time for a given rework probability. Accordingly, figure 7 shows that the re-engineered process is more reliable. The error bars around the simulated process durations, at any maximum probability value, are smaller for the re-engineered process, as observed in the figure.

The acceleration (curvature) of the curves describes the robustness of the process in terms of cycle time with respect to the maximum rework probability. A curve with a high acceleration, as in the original process of figure 7, would describe a less robust process whose duration is highly sensitive to changes in information. Conversely, a lower acceleration (flatter curve), as in the re-engineered process, would describe a process that is more robust to changes in input information. For example, a change in the maximum rework probability from 10% to 60% will result in an increase of about 150 days in the expected duration of the re-engineered process, compared with an increase of 750 days for the original process.

A cost (of process control)/benefit (cycle time reduction) strategy can also be established by analysing the profile of the probability curve. Assume that the maximum rework probability can be reduced by more closely controlling the process (i.e. adding verification tasks, project managers, redundancies, etc.). These controls obviously add some cost. Conversely, reducing present control would tend to increase the maximum rework probability but reduce the cost of control. For example, figure 7 shows that investing in reducing the maximum rework probability, for the original process, from 60% to 55% would result in expected development time savings of more than 300 days. Further reductions of the maximum rework probability would result in a greater decrease of expected development time, but start to exhibit a diminishing rate of return. Therefore, it may be economical not to invest in controlling the process, and hence reducing the maximum rework probability, below the 50% level.
6. Conclusion

This paper presents an assessment procedure for rework probabilities used in project and process management simulation models in general, and in DSM simulation models in particular. The assessment proceeds in three phases: subjective evaluation of task variability and sensitivity, mapping and calibration and validation. The application example shows that the probabilities required for simulating a DSM can be evaluated subjectively. Furthermore, it has been shown that this assessment method can also be used to shed some light on evaluating process improvement and re-engineering efforts by defining two new process measures: reliability and robustness. We show how the interplay between process reliability and robustness can be used to optimize the product development process. We propose that iteration is a trade-off between these two process measures, suggesting that optimal process performance can be achieved with a hybrid (reliable/robust) process.

We demonstrate the proposed assessment and evaluation methods using a real-world example of an automotive hood development process. The analysis began with a baseline process model. This model is correlated to the actual process by adjusting rework probabilities to obtain the appropriate process duration. Then the base model was restructured and re-engineered through the modification of existing tasks and the insertion of some new tasks in an attempt to decrease development cycle time and rework cost. Both models were then simulated and results were compared, showing quantitatively the superiority of the re-engineered process in terms of both reliability and robustness.

Finally, perhaps the most useful conclusion from this analysis is that the goal of restructuring an iterative process is not to break all iterative loops. Robustness obtained from iterative task structures can be more valuable than the reliability obtained in a sequential process structure.

References


