Proceedings of DETC '01
ASME 2001 International Design Engineering Technical Conferences
Computers and Information in Engineering Conference
Pittsburgh, Pennsylvania, September 9-12, 2001

DETC2001/DTM-21693

ASSESSMENT OF REWORK PROBABILITIES FOR SIMULATNG PRODUCT DEVELOPMENT PROCESSES USING THE DESIGN STRUCTURE MATRIX (DSM)

Ali A. Yassine*[+], Daniel E. Whitney* and Tony Zambito**

*Massachusetts Institute of Technology, Center for Technology, Policy and Industrial Development
Cambridge, MA 02139

**Ford Motor Company
Dearborn, MI 48126

ABSTRACT

This paper uses the Design Structure Matrix (DSM) to model and simulate the performance of development processes. Though the simulation is a powerful tool for analyzing process performance, its ability is limited by the quality of input information used in the analysis. DSM 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 parameter: rework probabilities. Furthermore, we show how does this assessment method (combined with simulation) allow managers to evaluate process improvement plans based on two resulting process measures: reliability and robustness. The method is illustrated with a real application from the automotive industry.

(Keywords: Design Structure Matrix, Product development, Simulation, Process Re-engineering)

INTRODUCTION

Complex product development processes are usually managed by mapping them through various kinds of project flowcharts and diagrams (i.e. Gantt charts, CPM, PERT, IDEF, etc.) that attempt to capture and manage process complexity and iteration. While many of these methods are capable of illustrating timing, information flows, and task interdependencies, they fall short of enabling project teams to effectively model, and gain deeper understanding of task interdependencies and iteration in the process. The design structure matrix (DSM) methodology provides a means to model and manipulate iterative tasks and multidirectional information flows. 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. 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 appear in process and project management literature.

[+] Corresponding author: MIT, CTPID, Rm. E40-253, Cambridge, MA. 02139.
Tel. (617)258-7734, Fax (617)452-2265, E-mail: yassine@mit.edu
This research has been supported by Ford Motor Company.

Recent project management literature points out to the potential and success of simulation techniques in managing development processes. However, few studies have been conducted as compared to manufacturing processes, for example. Perhaps one of these first attempts to handle feedback relationships, account for iteration, and enable simulation-based analyses was GERT (General Evaluation Review Technique) [Wiest, 1977].[1] Another paper by Adler et al. (1995) describes the use of discrete event simulation to study product development (PD) performance in companies pursing multiple, concurrent, non-unique PD 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 (1998) used Monte Carlo simulation based on a DSM representation of development projects. There simulation revealed several interesting process characteristics and performance measures including expected project duration, cost, and risk.

Though all these models emphasize the fact that simulation can be a powerful tool for analyzing process performance, their ability is limited by the quality of input information used in the analysis. These models require development data that is hard to assess or estimate directly from process participants. Such data include an estimate of iteration or rework probabilities. 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.

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 judgments necessary to conduct decision analysis. Similarly, Ford and Sterman (1998) acknowledge the fact that system dynamics modelers face difficulties in eliciting and representing expert knowledge so that useful models can be developed. Consequently, they developed an elicitation method that modelers 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 does this assessment method along with simulation results allow managers of development processes to evaluate process reengineering plans based on two resultant 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 analyzing 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 from automotive hood development. Finally, the example is used to show how can the simulation and the subjective assessment proposed in this paper be used to draw insights about ways to streamline and reengineer existing development processes.

2. THE DESIGN STRUCTURE MATRIX METHOD

The matrix representation of a directed graph is a binary (i.e. a matrix populated with only zeros and ones) square matrix with m rows and columns, and n non-zero elements, where m is the number of nodes and n is the number of edges in the digraph. The matrix layout, shown in Figure 1, is as follows: the process elements' names are placed down the side of the matrix as row headings and across the top as column headings in the same order. If there exists an edge from node i to node j, then the value of element ji (row j, column i) is unity (or marked with an X). Otherwise, the value of the element is zero (or left empty). The diagonal elements of the matrix do not have any interpretation in describing the system, so they are usually either left empty or blacked out.

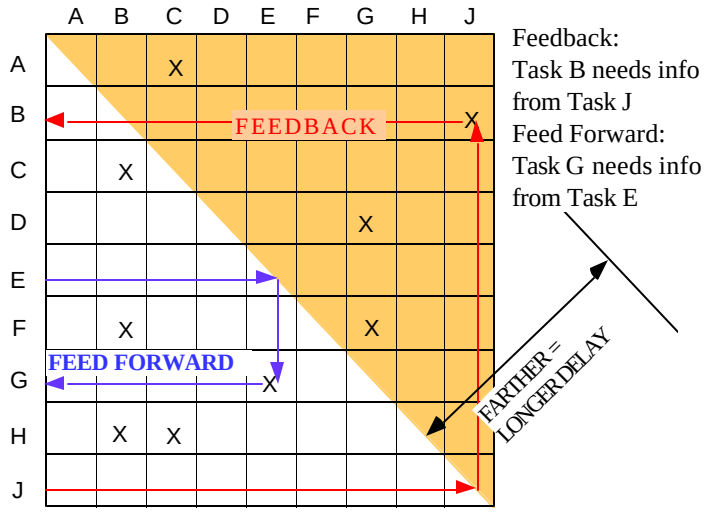

Figure 1: A Simple DSM

[1] GERT is an extension to the popular project management technique PERT (Probabilistic Evaluation and Review Technique).

A major advantage of the matrix representation over the digraph is its compactness and ability to provide a systematic mapping among process elements that is clear and easy to read regardless of size. If the system is a project represented by a set of tasks to be performed, then the order of elements in the rows (or columns) of the DSM represents the execution sequence of the tasks. Accordingly, the off-diagonal marks in a single row of the DSM represent all of the tasks whose output is required to perform the task corresponding to that row. Similarly, reading down a specific column reveals which task receives information from the task corresponding to that column. Marks below the diagonal represent forward information transfer to later (i.e. downstream) tasks. This kind of mark is called forward mark or forward information link. Marks above the diagonal depict information fed back to earlier listed tasks (i.e. feedback mark) and indicate that an upstream task is dependent on a downstream task. Backward dependencies are a major source of iteration in complex processes.

2.1. Partitioning and Tearing

Partitioning is the process of manipulating (i.e. reordering) the DSM rows and columns such that all the information (i.e. dependency marks) in the new DSM arrangement flows forward and there is no feedback; thus, transforming the DSM into a lower triangular form. For complex engineering systems, 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 minimizes them and move the rest as close as possible to the diagonal (this form of the matrix is known as block triangular). In doing so, fewer system elements will be involved in the iteration cycle 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 analogus 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 to binary DSMs, Numerical DSMs (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 judgment 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 3-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, 1998). 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

In this paper, we use the simulation utility described in Browning and Eppinger (1998) to demonstrate our proposed assessment and analysis procedures.[2] Therefore, in the rest of this section, we will describe this simulation technique more elaborately as compared to the other simulation models referenced earlier in this paper.

The DSM-based simulation model for a development process, as discussed by Browning and Eppinger (1998), quantifies 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.

[2] The software is downloadable for free from the MIT DSM web site: http://web.mit.edu/dsm/

Since iterations may or may not occur (depending on a variety of variables), 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 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, the simulation 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. We propose a three-stage procedure:

1. SUBJECTIVE ASSESSMENT: Define two independent constructs (dimensions): Information Variability, and Task Sensitivity.
2. MAPPING and CALIBRATION: The values of those two dimensions are mapped to a probability space. Then, they are correlated to probabilities using some proportionality constant based on one or more known criteria, such as project duration, project cost, …etc.
3. VALIDATION: Confirm the above resultant probabilities.

3.1. Subjective Assessment of Information Variability (IV) and Task Sensitivity (TS)

Off-diagonal elements in our NDSM are called "task volatility".[3] Task volatility (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. Task volatility is the product of two components: Information Variability and Task Sensitivity.

A. Information variability (IV) describes the likelihood that information provided by an input task would change after being initially released. 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. Information variabilities are located along the bottom of the matrix and correspond to the task in that column (see 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.

Value	Description	Likelihood of Change
1	Stable	Low
2	Unknown	Medium to high
3	Unstable	Very high

Table 2: Levels of information variability (IV)

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

Value	Description	Dependent Task is:
1	Low	Insensitive to most information changes
2	Medium	Sensitive to major information changes
3	High	Sensitive to most information changes

Table 3: Levels of task sensitivity (TS)

Interdependency marks in the DSM are replaced by numerical Task Volatilities (TV), the product of IV and TS.

[3] Marks above the diagonal represent first order rework and marks below the diagonal represent the potential for second order rework (Browning and Eppinger, 1998).

While IV and TS are closely related, it is worth noting that they are independent. Task sensitivity is a measure associated with a dependent task, whereas information variability is associated with an input task. Therefore, it is appropriate to multiply TS and IV to describe task volatility. Given the possible values of IV and TS, we find that TV values can range from 1 to 9.

A low value for either TS or IV neutralizes the impact of the other. Thus, one might suggest that TV values of 1, 2, or 3 indicate that a dependent task is stable with respect to the corresponding predecessor task. For example, if variability of information from a predecessor task is high (say 3) but the sensitivity of the dependent task is low (say 1), then it is unlikely that the dependent task will be affected. However, this is not necessarily true for the reverse case. While a combination of low variability and high sensitivity indicates that the dependent task is unlikely to be affected, the dependent task will require rework if the input information changes at all. With this in mind, we use a conservative approach by assuming TV values of three (3) imply moderate dependency, rather than a weak dependency. Table 3 shows the possible ranges of TV values, their significances, and high-level strategies for handling each level.

TV Value	Description	Strategies
1, 2	- Dependency is weak. - Low risk of rework.	Feedback and forecast information may be used, especially if it promotes process robustness.
3, 4	- Dependency is moderate. - Moderate risk of rework.	Avoid using forecast and feedback information where possible.
6, 9	- Highly sensitive to change. - High risk of rework.	Task sequence is critical to process reliability. Do not use forecast and feedback information.

Table 4: TV values, their significance, and proposed strategies

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 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 1000 days, then we can correlate the rework probabilities for the process model such that the simulations provide an average duration of 1000 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 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.

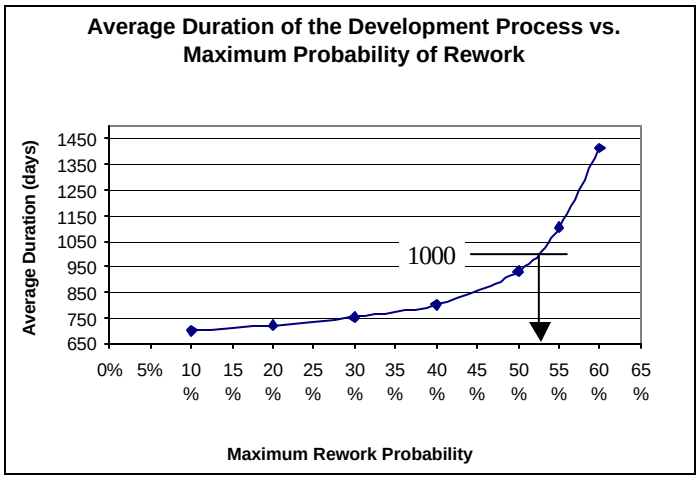

Figure 2: Average duration associated with different rework probabilities – (The average durations were obtained by simulating an actual process known to require approximately 1000 days)

From this analysis, we see that 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 process simulations. The mapping of task volatilities to rework probabilities is provided in Figure 3.

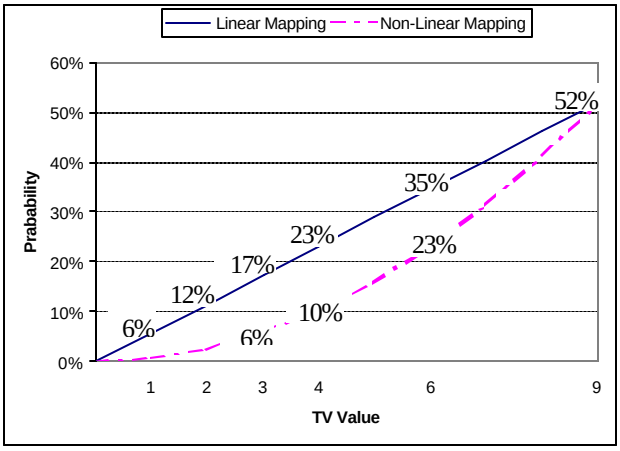

Figure 3: Probability mapping

It is possible to use linear or non-linear (for example, quadratic) mapping functions as shown in Figure 3. The choice depends on the kind of behavior 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. This is the kind of behavior desired in our mapping function; however, a linear mapping for the rework probabilities was used in this paper since it provides more conservative probability estimates.

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. INDUSTRIAL APPLICATION: AUTOMOTIVE HOOD DEVELOPMENT PROCESS[4]

A high-level description of the hood development process is as follows (Zambito, 2000). 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.

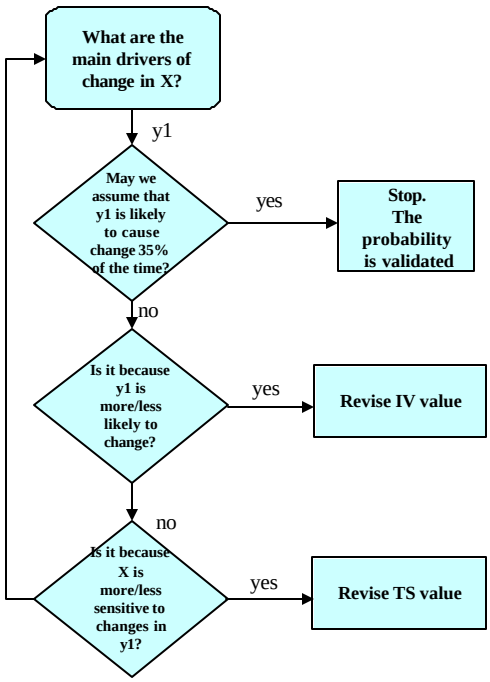

Figure 4: Validation phase

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.[5] 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 its location in the matrix.

4.1. Data Collection

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

[4] Note that the time and cost data used in this application are scaled to protect confidentiality, but preserve its characteristics.

[5] An iterative loop depicts a situation were 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.

unavailable or needed verification. This helped eliminate single-perspective biases and ensure that the DSM accurately described the actual hood development process. Stakeholder interviews included experienced representatives from manufacturing, assembly, computer-aided engineering (CAE), computer-aided design (CAD), product engineering, and product engineering management. In total, 15 personal and/or phone interviews with 10 stakeholders were conducted. The duration of each interview varied from 45 to 90 minutes. The interviews were structured as follows:

a. Identifying the task(s) of the hood development process that the interviewee is involved in, along with a brief summary of the task.
b. Determining the nominal duration (and cost) of the task.
c. Determining the rework duration (and cost) of the task.
d. Determining the inputs needed to carry out the task.
e. Determining the sensitivity of the task to changes in above inputs (TS values).
f. 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 (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 shown in Table 2.

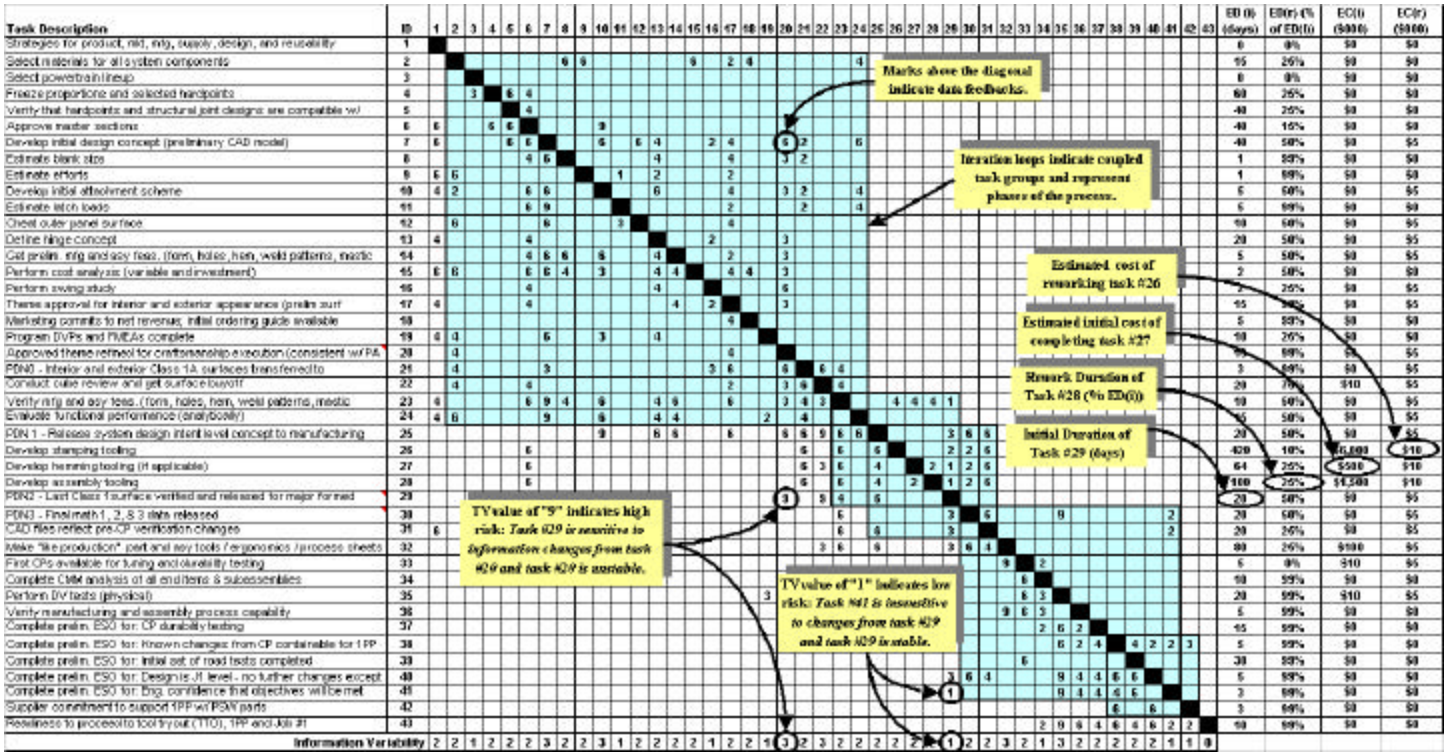

Figure 5: DSM for the hood development process
[ED(i) represent the expected duration of task i,
ED(r) represent the rework duration of task i as a % of ED(i)
EC(i) represent cost of task completion and EC(r) represent the cost of rework]

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 helped us in assessing the variability. For example, responses (from different experts) relating to task 10 duration varied from 2 days 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.

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 to 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 tradeoff 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 tradeoff 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 a very robust one. That is, a completely reliable process offers little robustness because it is not capable of incorporating changing inputs that are external to the process (changing consumer needs or regulatory requirements, for example). In many cases, iteration in the development of products is needed for optimizing key product attributes.

The probability curves shown in Figure 6 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 a corresponding reengineered process. Without going into the details of the reengineered process, the reader may think of it as similar to the original process but with few process changes resulting in the deletion of some old tasks and addition of new ones (Yassine et al., 2000).[6] Process reliability is described through the curves by the range of probable durations (the error bars in Figure 6). 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 6 shows that the reengineered process is more reliable. The error bars around the simulated process durations, at any maximum probability value, are smaller for the reengineered 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 6, would describe a less robust process whose duration is highly sensitive to changes in information. Conversely, a lower acceleration (flatter curve), as in the reengineered 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 reengineered process, compared to an increase of 750 days for the original process.

A cost (of process control) / benefit (cycle time reduction) strategy can also be established by analyzing 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.).

[6] Indeed some of the information dependencies have been also altered in the new process.

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 6 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 then 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.

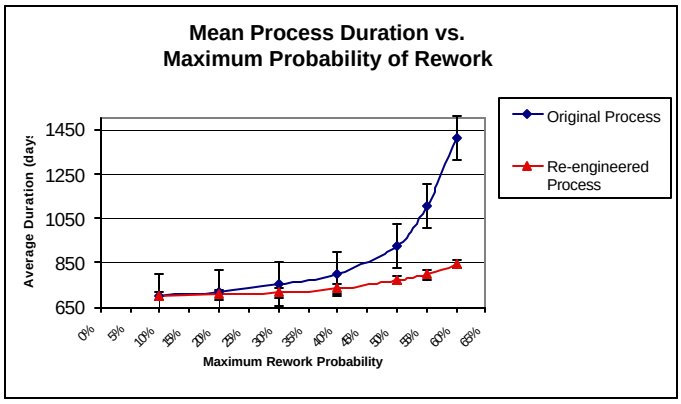

Figure 6: Reliability and robustness of development processes

6. CONCLUSION

This paper presented a subjective assessment procedure for rework probabilities used in project and process management simulation models, in general, and in the DSM simulation model developed in Browning and Eppinger (1998), 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, this assessment method can also be used to shed some light on evaluating process improvement and reengineering efforts by defining two new terminologies: reliability and robustness.

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.

REFERNCES

Adler, P., Mandelbaum, A., Nguyen, V. and Schwerer, E., "From Project to Process Management: Empirically-Based Framework for Analyzing Product Development Time," Management Science, Vol. 41, No. 3, March 1995.

Baldwin, A., Austin, S., Hassan, T., and Throrpe, A., "Modeling information flow during the conceptual and schematic stages of building design," Construction Management and Economics, 17, pp. 155-167, 1999.

Browning, Tyson and Eppinger, Steven, "A Model for Development Project Cost and Schedule Planning," M.I.T. Sloan School of Management, Cambridge, MA, Working Paper no. 4050, November 1998.

Ford, D. N., and J. D. Sterman, "Expert Knowledge Elicitation to Improve Formal and Mental Models," System Dynamics Review, Vol. 14, No. 4, pp. 309-340, 1998.

Keeney, R., Value Focused Thinking. Cambridge, Massachusetts: Harvard University Press, 1992.

Merkhofer, M. (1987). Quantifying Judgmental Uncertainty: Methodology, Experiences, and Insights. IEEE Trans. Sys., Man, and Cybern., vol. SMC-17, pp. 741-752.

Pimmler, T.U. and S.D. Eppinger. "Integration Analysis of Product Decompositions", ASME Conference on Design Theory and Methodology, Minneapolis, MN, pp. 343-351, September 1994.

Shephard, G. and Kirkwood, C. (1994). Managing the Judgmental Probability Elicitation Process: A Case Study of Analyst/Manager Interaction. IEEE Trans. on Engineering Management, vol. 41, no. 4, pp. 414-425.

Steward, Donald V., "The Design Structure System: A Method for Managing the Design of Complex Systems" IEEE Transactions on Engineering Management, vol. 28, pp. 71-74, 1981.

Wiest, J.D. and Levy, F.K., A Management Guide to PERT/CPM: With GERT/PDM/DCPM and Other Networks, Second ed. Englewood Cliffs, NJ: Prentice-Hall, 1977.

Yassine, Ali, Whitney, Daniel, Lavine, Jerry and Tony Zambito, "DO-IT-RIGHT-FIRST-TIME (DRFT) Approach to Design Structure Matrix Restructuring", Proceedings of the 12th International Conference on Design Theory and Methodology, Sept. 10-13, 2000 Baltimore, Maryland, USA.

Yassine, A., Falkenbug, D., Chelst, K., (1999), Engineering Design Management: an information structure approach., International Journal of Production Research, vol. 37, no. 13, pp. 2957-2975.

Zambito, Tony, "Using the Design Structure Matrix to Structure Automotive Hood System Development," MIT SDM Masters Thesis, January 2000.