Improved span time performance in NPD using better coordination

Samuel Suss and Vince Thomson

Department of Mechanical Engineering, McGill University
817 Sherbrooke Street West, Montreal, Canada H3A 2K6
Tel. 1-514-398-2597, Fax 1-514-398-7365.
samuel.suss@mail.mcgill.ca, vince.thomson@mcgill.ca

Abstract. New product development (NPD) processes are characterized by uncertainty and iteration making them difficult to plan and manage. A novel dynamic model of NPD that explicitly models communication techniques is used to examine ways to improve span time and effort performance through improved coordination. Results of simulations under various scenarios of epistemic and aleatory uncertainty illustrate how coordination and adapting agile NPD methods to non-software product development can be used to attain significantly better performance.

Keywords: Collaborative product development, coordination, complex product development, concurrent engineering

1 Introduction

New product development (NPD) is a critically important part of product lifecycle. It consumes a large proportion of the overall time of bringing a product to market, and determines about 70% of product cost [1]. The implementation of engineering design tools, concurrent engineering (CE) practices and product data management systems has reduced NPD cycle times in recent years. However, in large NPD projects where hundreds of engineers work to develop complex products, there remain significant inefficiencies. Researchers have estimated that the amount of waste in aerospace and defence NPD programs is 60–90% of the charged time with about 60% of all tasks being idle at any given time [2]. The actual time engineers spend on value-added activities is much less than half of their total working time. There is much efficiency lost in wasted communication, waiting for information and lack of coordination.

Although effective teamwork and individual initiative are important in the successful performance of NPD, the manner in which the engineering design process in NPD is structured and coordinated is found to be the factor of significant consequence [3-9]. Engineering design is inherently iterative and uncertain [10, 11]. The way in which the process is divided into differentiated tasks leads to dependencies between the tasks. Teams of specialists carrying out these tasks create evolving information that reduces uncertainty during the process and that is required by other

dependent tasks. It is the way that this developing information is shared among interdependent tasks that leads to rework and this significantly affects the required effort and span time of the process. Thus, in planning an NPD process it is the choice of the work breakdown structure and the way in which interdependent work is managed that determines to a great extent the difficulties that teams and individuals encounter in their efforts to successfully complete a project.

With a focus on engineering design of complex products, the goal of this research is to find methods for achieving significantly faster NPD, i.e., the reduction of the span time to create a product. We argue that this can be done with better coordination of the NPD process and that this in turn can be achieved through a better understanding of how coordination strategies and tactics impact the creation and communication of information, and under what conditions they are effective. In order to accomplish this, we treat the NPD process as a complex system of elements: resources, tasks, and developing information, that interact to change the state of the system, and we study engineering design processes using computer modeling and simulation.

One of the main distinctions regarding uncertainty in engineering design is between the lack of knowledge (epistemic) and stochastic (aleatory) uncertainty [12]. Epistemic uncertainty is due to ignorance or incomplete information. Some epistemic uncertainty is reducible, for instance, via further studies, measurements or expert consultation. Stochastic uncertainty is the inherent variation associated with a system or environment such as dimensional variation in a production system, the variation in task duration, or the unexpected change in product requirements originating from the customer. Fundamentally, the rate of progress towards the certainty of information is affected by how a design task reduces epistemic uncertainty, and this is a function of the amount of work done in the task and the uncertainty of input information. The aleatory uncertainty of information developed in a task and its input information is always present, and its influence must be taken into consideration [13].

2 The model

In order to analyse the effects of coordination, we model information flow explicitly and use it to measure task progress in an NPD model, hereinafter referred to as the collaborative process model or CoPM for brevity [14]. We consider a generalized engineering design process composed of tasks arranged into phases with a design review at the end of each phase. Each of the tasks is considered to be performed by a cross-functional development team. In keeping with the practice observed in industry to ensure that design work being done by development teams integrates together correctly into a product, we also include a system level integrator in our process model.

CoPM considers the process as a dynamic system with tasks, resources and information as its elements. The tasks are linked together through their need for information from each other in order to make progress during the design work. Resources are linked to information and tasks by the requirement that a resource assigned to a task can do work or process information, but not both simultaneously.

Information processing requirements are linked to the dependency strength between tasks.

In CoPM, we split a task into three subtasks: *work*, which is comprised of the technical activity required to complete the task; *read*, which is the time spent by a resource performing the activity required to read, interpret and comprehend incoming information; and *prepare*, which is the time spent by the resource in preparing information for communication. We model each of these subtasks as a stochastic process so that the amount of time a resource needs to perform each activity is chosen from an inverse triangular probability density function (PDF). The parameters of the triangular PDF chosen to represent the *work* subtask reflect the capabilities of the team chosen to perform this task.

Supported by our own observations in product development organizations and those reported by Allen [15], we assume that the total amount of information that must be communicated between interdependent tasks is directly proportional to the strength of their mutual dependency [14]. We reason that, if there is greater uncertainty in an activity's output, it is likely that more estimates of the output information need to be generated and communicated to dependent activities before the design activity is completed. Similarly, if there is greater sensitivity to an activity's output, it is likely that more information needs to be transferred before the linked activities arrive at a jointly satisfactory solution. For each increased level of uncertainty, the effect of sensitivity is magnified and vice versa; so, we assume that the effects of information uncertainty and task sensitivity are multiplicative.

Information is conceptualized as discrete units, and thus, each unit requires a stochastic amount of processing time during the *prepare* or *read* subtasks. There are, therefore, a fixed number of units of information that must be received by a dependent task from other interdependent tasks. The number of information units that must be communicated from any task i to another task j for each pair of tasks in the process are represented as a matrix, **NC,** proportional to a design structure matrix **D,** where the numerical value of each of the elements represents the dependency strength between the pair of tasks. The dependency strength may be formed from the product of the initial uncertainty and sensitivity between each pair of tasks [16].

Each task and each unit of information carries out its own processing thread in the simulation of the design process. Each thread is subject to mathematical and logical rules governing executed events, and the values of the state variables that are assigned accordingly. The primary state variables are WT_{ti}, the amount of work performed by task i until time t; WD_i, the amount of work required to be performed by task i (which is initially set by an inverse PDF, but later can be increased when rework is required); I_{ti}, the number of units of information read by task i until time t; and γ_i, the total number of units of information from all other tasks required to be read by task i.

The mathematical and logical rules governing the processing threads and values of state variables are summarized as follows:

(1) Work in a task stops when there is insufficient input information. This occurs when WT_{ti}/WD_i, the fraction of work performed in a task i, exceeds a

stochastically determined starve condition which is distributed normally about I_{ti}/γ_i , the fraction of information received by the task.

(2) Information is generated in a task in accordance with the progress of work. The n^{th} unit of information is 'created' in task i when $WT_{ti}/WD_i \geq n\ I_{ti}/\gamma_i$.

(3) Work in each task is performed in cycles between which communication is done, if required. The maximum length of each cycle, Δt_i, is a modeling input for each task i in the process being modeled.

(4) The state of progress of a task at any time t, S_{ti}, is assumed to be a linear combination of the achieved technical work fraction and the received input information fraction. Note that S_{ti} varies from 0 to 1.

$$S_{ti} = \frac{1}{2}\left(\frac{WT_{ti}}{WD_i} + \frac{I_{ti}}{\gamma_i}\right) \quad \textbf{(1)}$$

(5) The ratio of the epistemic uncertainty of each unit of information created by task i at time t to the initial epistemic uncertainty of that task, ε_{ti} , is given by a modified Gompertz function which is an S-shaped curve that allows for different rates of approach to the lower and upper asymptotes at the start and end of a task:

$$\epsilon_{ti} = 1 - e^{\left(-b_i e^{(-c_i S_{ti})}\right)} \quad \textbf{(2)}$$

where b_i and c_i are input parameters that govern the shape of the profile of each task.

(6) The ratio of the aleatory uncertainty of each unit of information created by task i at time t to the initial uncertainty of that task is given by the following equation:

$$\varphi_{ti} = \epsilon_{ti} m_i \mathbf{UNIF}(0,1) \quad \textbf{(3)}$$

where m_i is a scaling factor and **UNIF**(0,1) is a sample chosen from the uniform probability distribution between 0 and 1.

(7) If the uncertainty of received information in a task is greater than the uncertainty of information received earlier, rework is required. The amount of rework is calculated as the amount of work done in the intervening work cycles where the input uncertainty is lower than the most recent information received. Rework is manifested as an increase in WD_i. The changes to all the WD_i during a simulation are accumulated and recorded as an output statistic called *churn*.

(8) A task is completed when it finishes its work requirement, i.e., $WT_{ti}=WD_i$, and when it has received all of its required input information, i.e., $I_{ti}= \gamma_i$, or unless it is stopped because the allotted span time for that task (a model input for each task) has been exceeded. A phase is completed when all the tasks in the phase are done.

(9) Once tasks in a phase are completed the possibility of design version rework is checked, where its probability is governed by the minimum of:
    (a) the certainty of information in each task at the end of the phase,
    (b) the percentage of completion of information exchange of each task,
    (c) the percentage of completion of work of each task, and
    (d) the percentage of completion of system level oversight.

(10) Design version rework requires less effort each time it is repeated. This is because of first order learning [17] and because of a reduction in epistemic uncertainty. The effects of both of these are included in the model.

Although we assume that information develops uniformly with progress made in the technical work of the task, information is not communicated as soon as it develops, but rather is prepared for communication and transmitted periodically as one would expect in practice. In CoPM, therefore, the created information entities wait until an event triggers their release to the *prepare* queue. This event occurs each time the ratio WT_{ti} /WD_i reaches an integer multiple of the modeling input CI_i which defines the communication interval as a fraction of the nominal span time of task *i*. Thus, when the following condition is true, the k_{th} batch of information entities is released to the *prepare* queue of task *i*:

$$WT_{ti} / WD_i \geq k\ CI_i \quad \text{for all } k = 1, 2, 3, \ldots, 1/CI_i \tag{4}$$

By focusing on how information is used during the design process (design work plus communication or processing of information), the CoPM considers the trade-off between direct design effort (WT) and indirect activities such as communication (I, γ) and project management (system level oversight, which is explicitly modelled). This allows the CoPM to be very realistic in comparison to actual design tasks in industry both in how the model operates and in results.

3 Insights from simulations with the model

Detailed results for a wide variety of scenarios can be seen in [14]. Space limits the descriptions here to highlights of the findings in each of the following categories.

3.1 Overlapping sequentially dependent tasks with various scenarios of uncertainty

The effects on effort and span time for overlapping sequentially dependent tasks were studied with CoPM for various profiles of reduction in epistemic and aleatory uncertainty (see Fig. 1 for an example showing results with high aleatory uncertainty).

For tasks with fast reduction of epistemic uncertainty there is a substantial reduction in span time when tasks overlap. This is evident even when there is a large magnitude of aleatory uncertainty and strong task interdependence, but this requires frequent communication of interim information (CI_i =0.1) and no delays in informa-

tion flow due to latency or resource constraints. This is not the case for tasks with slow reduction in epistemic uncertainty, where overlapping tasks by more than 50% has little additional benefit in span time, and in fact, incurs a significant increase in effort.

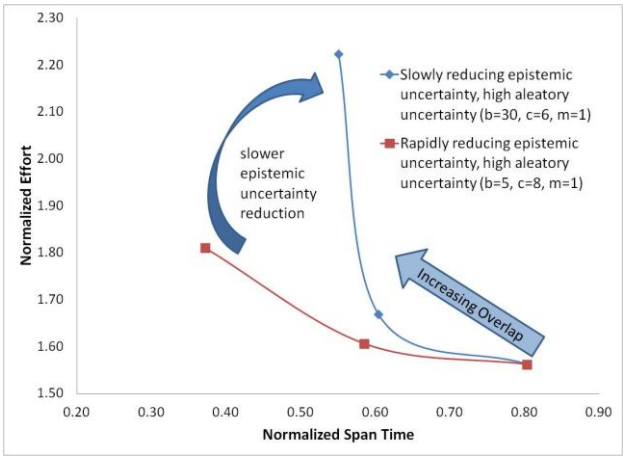

Fig. 1. Effort versus span time for two profiles of epistemic uncertainty reduction for overlapping of five sequentially dependent tasks

Examination of simulation results shows that span time and effort performance is affected by the increasing amount of churn that is generated when there is high overlap. When uncertainty reduces more slowly, there is more rework generated with increasing overlap, and tasks do not make progress, but spend more time doing rework due to imprecise information. The additional rework results in increased effort, and an increase in span time. Simulations with several different values of random uncertainty *m* and slowly reducing epistemic uncertainty show that the reduction in span time is insensitive to the magnitude of *m* at 50% overlap. However, for fully overlapped tasks, there is a wide divergence with *m*, indicating an increasingly larger amount of effort and increasingly smaller additional reduction in span time when random uncertainty is higher. This result is primarily due to the increasing amount of churn that is generated.

3.2 The effects of communication interval

With high interdependency, parallel execution with interim information exchange is an important method of enabling dependent tasks to effectively continue with their work. The timeliness with which interim information is exchanged allows each task to make progress towards a successful design review at the end of each phase. However, exchanging uncertain interim information can lead to unnecessary rework which impedes the progress of tasks.

Simulations show that with slowly reducing epistemic uncertainty, span time reduces with smaller communication intervals until a minimum is reached at CI_i approximately equal to 0.1 (the optimal point). Span time reduces with more interim communication by nearly 75% at the optimal point. When CI_i reduces below the optimal point, span time rises, and the steepness and magnitude of the rise is strongly affected by the value of aleatory uncertainty *m*. With more rapid reduction in epistemic uncertainty, there is a 35% change in span time over the range of CI_i. The minimum span time with more rapid reduction in uncertainty is 36% lower than that with more slowly reducing uncertainty.

Examination of the results for rework, starve time and design version rework shows that the behaviour of the NPD system is largely effected by the increase in starve time when CI_i is above the optimal point. This results in an increase in design version rework. The starve time increase is a result of tasks not getting enough information to allow them to progress in their work when the communication interval is too high. This causes an insufficient reduction in uncertainty when a design review takes place and design version rework is triggered. When CI_i is below the optimal point, increased span time is caused by churn from too frequent communication of interim information that is subject to random uncertainty.

3.3 Effects of delays in information flow

In projects performed by many people in various locations, delays in simply getting information to the attention of those that need to make use of it can be a significant portion of the time required to do the work itself. These delays, where information must travel through several layers within an organization and queue for the attention of each reviewer, can cause unnecessary rework with significant knock-on effects. Moreover, since engineers and designers are often occupied with several projects at the same time, there is a delay before they turn their attention to following up on missing information required to make progress on one of their tasks. These delays in information flow are referred to as communication latency.

We divided latency into three types: the delay caused by the path the information must take through various levels in the receiving organization before it reaches the final addressee; the delay caused by resource constraints in the system level integrator process; and the delay occurring when the addressee team does not turn its attention immediately to incoming information.

CoPM simulations evaluated the impact of delays for various product development system structures and levels of uncertainty. Each of the delay types, when applied in isolation had a relatively small effect on span time. However, it was found that sources of delay in information flow combine in a non-linear way to increase span time significantly. Each source of delay increases the likelihood that information getting to a dependent task is not continuously reducing in uncertainty and this generates churn as tasks operate with imprecise information. Each additional source of delay further exacerbates the delays due to other sources and leads to a tipping point where the combination of delays is such that tasks cannot reduce their uncertainty sufficiently, and cycles of design version rework ensue.

This insight has important managerial implications in that reducing delays in information flow between interdependent tasks can have a large effect on reducing span time. Making an effort to reduce one or more sources of delay in information flow has an outsized effect on project performance.

3.4 Adapting agile NPD methods to non-software product development

In this section, we discuss a coordination scheme called 'scrum' that is part of agile product development methods [18]. In the scrum scheme, a development team is a cross-functional group that does the analysis, design, implementation, testing, etc., that is required to create a deliverable software product in short increments (typically one month). During this period of time, called a 'sprint', the team is required to produce an entire, tested version of the software that completely answers a planned set of requirements. Each successive sprint goes on to add additional requirements so that at the end of the project the software meets the complete list of requirements for the final product. The sprints are characterized by intense communication within the self-managing development team (typically co-located), and by an ironclad commitment to achieving the agreed to requirements (which cannot be changed during a sprint) within the allotted time frame. This product development method has been found to be effective in significantly reducing software development time.

Although this scheme is feasible in software development, it cannot be replicated literally in mechanical design where physical parts must be designed, materials procured, and then undergo a manufacturing process in order to produce a prototype that can be tested. However, in an analogy to the scrum concept, we could consider the goal of a sprint to be the solution to a series of well defined design problems, providing the information required to make a key decision in an NPD project. Each successive sprint would then provide further information to an additional series of design problems until the design of the final product is achieved.

A key element here is that the tasks involved in each phase completely solve a specific set of design problems that can be evaluated during a design review. Since NPD of a complex product is essentially a series of activities providing information that allow key design decisions to be made, the design review at the end of each of these short phases or 'sprints' formalizes the decisions. The tasks in each phase leading up to a design review are those that generate the information required to make these key decisions. As in the sprint method for software, the work in each phase must be done with intense interaction between all participants so that the design review is successful and rework is not necessary. In practice, this is facilitated by smaller sized sprints.

Thus, in our model, a simulated project is divided into a series of six short phases with five tasks, analogous to sprints with a design review at the end of each phase. To model the co-location and intense communication between the various tasks, we eliminated the system level integrator function as the scrutinizer of each information item, and included the integrator as a participant in the co-located group of development teams. All delays due to latency of information exchange were removed. For comparison, we performed simulations with the same technical work

content with two phases and five tasks with an integrator function acting as oversight of all information communicated between the tasks, and communication latency to get information from one development team to the other.

Simulations show that with slow reduction in epistemic uncertainty and high aleatory uncertainty, span time is 33% lower with the scrum method (Fig. 2). When the magnitude of aleatory uncertainty increases from 0 to 0.9 the span time remains approximately constant in the scrum scenarios, but increases by approximately 25% in the standard scenarios. This result is primarily due to lower churn when using the scrum method and the mitigation of increases in random uncertainty. When epistemic uncertainty reduces more rapidly, both the standard and scrum methods are insensitive to increases in the magnitude of aleatory uncertainty.

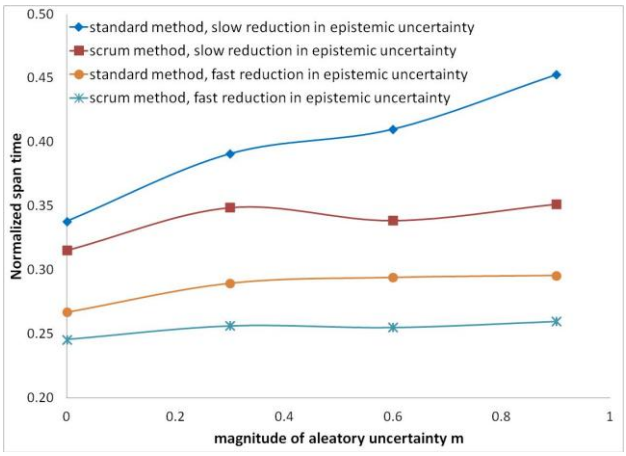

Fig. 2. Comparison of span time in scrum and standard NPD for several cases of uncertainty.

The scrum method enables more frequent and rich communication of information, minimizing latency and its effects. Short sprints with more design reviews provide opportunities to efficiently judge the work accomplished in each sprint and to provide feedback, keeping the work from straying too far off track. In interviews with senior NPD project managers in the aerospace industry, we found that frequent design reviews are sometimes used. In the words of a veteran NPD project director we interviewed, "with frequent interim design reviews we were able to keep the work on track … stay 'out of the ditch' and keep the work as close as possible to the 'white line' in the middle of the road." Thus, the scrum method works by allowing intense coordination among the people doing the work who can most effectively do this coordination, while providing opportunity for managers to keep the work on track and to refine and clarify requirements as the project progresses.

4 Discussion and conclusion

The explicit modeling of information flow in an NPD process enabled CoPM to capture the dynamic complexity of projects with interdependent tasks. This was accomplished through the explicit and detailed modeling of information exchange, the linkage of information exchange to the work accomplished in each task, the deployment of resources, and the techniques used to manage the NPD process.

The model differentiated between unnecessary rework and the iterative refinement of tasks that occurred as information was communicated from task to dependent task. This iterative refinement was incorporated in the information exchange between each pair of tasks, where each group of interdependent tasks had to exchange information that was increasingly updated as the process was carried out. Rework was emergent in CoPM due to the effects of the changing uncertainty of information and its interaction with the dynamics of the information flow. Information improved during each task, and the attributes of this information with regard to quality and uncertainty evolved as the task progressed. Interim information that was communicated carried these information attributes that were related to the state of the work in the task that created it. This information, when used by other tasks, had an effect on their progress. If information was imprecise or unstable, and if it was used inappropriately as a basis for further work or decisions, the arrival of the contradictory information triggered rework of some or all of the work.

Simulations with CoPM showed that an appropriate interval between communication of interim information optimized task progress and minimized project span time and effort. It was found that the optimal point of communication was quite defined when the uncertainty of tasks reduced slowly and there was significant magnitude of random uncertainty. If uncertainty reduced quickly or there was little random uncertainty, the more frequent the communication the better. The mechanisms that increased span time and effort at either side of the optimal point were design iteration rework (churn) due to the too frequent communication of uncertain information, and design version rework where there was insufficient frequency of communication to avoid a deadlocked condition in interdependent tasks.

Where information flow was restricted between interdependent tasks, effects due to delays and resource bottlenecks combined in a non-linear way leading to tipping points. Each type of delay had a small effect by itself, but exacerbated the effects caused by other delays leading to increasing levels of churn and design versions. Conversely, the reduction of some delays and bottlenecks with the use of managerial and technological solutions had a highly leveraged effect on reducing span time.

When there was a high degree of interdependency between tasks in an NPD process, we found that span time could be reduced significantly by adopting the ‘scrum’ method. Here, the work had to be structured to allow for ‘sprints’ with intensive coordination between self-managing teams executing tasks that together solved well-defined design problems. After each sprint, an interim design review was performed to evaluate the intermediate outcomes, to provide feedback, and to set the detailed requirements for the next sprint. In this way, the impediments to information exchange were minimized and the need for system level oversight was effectively

met. This method had the greatest impact in reducing span time for tasks with higher random uncertainty.

In terms of takeaways for product development managers, there are two specific, actionable items. First, the research in this paper shows that management of information transfers is the key to an effective product development process. The work with the collaborative process model showed the high impact of information transfers on process effectiveness (amount of communication, timing of communication, system level oversight), but did not discuss many, alternative coordination mechanisms. A discussion on coordination mechanisms can be read in [19].

The second takeaway is the effectiveness of the scrum method. It was clearly shown that increasing design reviews shortens span time. The scrum method has been very effective in the software industry and is now being used in other design domains.

Overall, the results of the CoPM simulations help NPD projects by providing guidelines for improving information flow. CoPM has proven to be a useful tool by providing a deeper understanding of the mechanisms that drive project performance.

References

1. Wheelwright, S. and K. Clark, 1992, "Revolutionizing Product Development." New York: The Free Press.
2. Oppenheim, B.W., 2004, "Lean Product Development Flow," Systems Engineering, 7(4): p. 352-376.
3. Browning, T.R., E. Fricke, and H. Negele, 2006, "Key Concepts in Modeling Product Development Processes," Systems Engineering, 9(2): p. 104-128.
4. Joglekar, N. and D. Ford, 2005, "Product Development Resource Allocation with Foresight," European Journal of Operational Research, 160((1)): p. 72-87.
5. Kerley, W., et al., 2011, "Redesigning the design process through interactive simulation: a case study of life-cycle engineering in jet engine conceptual design," Int. J. Services and Operations Management, 10(1): p. 30-51.
6. Mihm, J. and C.H. Loch, 2006, "Spiraling out of control: Problem-solving dynamics in complex distributed engineering projects," in "Complex Engineering Systems," D. Braha, A. Minai, and Y. Bar-Yam, Editors. Perseus Books: New York.
7. Terwiesch, C., C.H. Loch, and A.D. Meyer, 2002, "Exchanging Preliminary Information in Concurrent Engineering: Alternative Coordination Strategies," Organization Science, 13(4): p. 402-419.

8. Wynn, D., P.J. Clarkson, and C. Eckert, 2005, "A Model-Based Approach to Improve Planning Practice in Collaborative Aerospace Design."
9. Yassine, A.A., et al., 2003, "Information hiding in product development: the design churn effect," Research in Engineering Design, 14(3): p. 145-161.
10. Safoutin, M.J., 2003, "A methodology for empirical measurement of iteration in engineering design processes," University of Washington, Seattle, Washington.
11. Fredriksson, B., 1994, "Systems Engineering–A Holistic Approach to Product Development," Griffin, 94: p. 95-105.
12. Oberkampf, W.L., et al., 2004, "Challenge problems: uncertainty in system response given uncertain parameters," Reliability Engineering & System Safety, 85(1-3): p. 11-19.
13. Suss, S., K. Grebici, and V. Thomson, 2010, "The Effect of Uncertainty on Span Time and Effort within a Complex Design Process," in Modelling and Management of Engineering Processes, P. Heisig, J. Clarkson, and S. Vajna, Editors, Springer: London p. 77-88.
14. Suss, S., 2011, "Coordination in Complex Product Development " PhD Thesis, McGill University, Montreal, Canada.
15. Allen, T.J., 2007, "Architecture and communication among product development engineers," California Management Review, 49(2): p. 23-41.
16. Yassine, A.A., D. Falkenburg, and K. Chelst, 1999, "Engineering design management: an information structure approach," International Journal of Production Research, 37(13): p. 2957-2975.
17. Von-Hippel, E. and M.J. Tyre, 1995, "How learning by doing is done: problem identification in novel process equipment," Research Policy, 24(1): p. 1-12.
18. Cockburn, A., 2006, "Agile Software Development: The Cooperative Game, 2006." Professional: Addison Wesley.
19. Liang, W.H., 2009, "Coordination Mechanisms for New Product Introduction," Master Thesis. McGill University, Montreal, Canada.