Benefits of a critical chain – a System Dynamics based study

Abstract Number: 002-0351

Second World Conference on POM and 15th Annual POM Conference
Cancun, Mexico, April 30 - May 3, 2004.

Jan Jürging
Industrieseminar, University of Mannheim
Schloss Südflügel, 68131 Mannheim
E-Mail: juering@is.bwl.uni-mannheim.de
Tel.: +49 – 621 – 181 1750
Fax: +49 – 621 – 181 1579

Abstract

Introducing the “Theory of Constrains” (TOC) Goldratt established a management philosophy for production systems and later transferred this approach to project management. This resulted in scheduling and management techniques, which are explicitly aware of uncertainty and address the increasingly importance of completion time itself and its reliability. Other concepts, like simultaneous engineering, have been developed for a time reduction in projects, but shorter project durations are accompanied by higher risks and a rising complexity. Similar to “The Goal” the TOC’s solutions for project management follow common sense and their value seems obvious. System Dynamics is used as simulation methodology, because it has proven its applicability to project management. It takes a systemic view on the problem, incorporates hard and soft variables, and handles complexity, non-linearity, uncertainty, and feedback. A System Dynamics model will be presented that validates the TOC’s benefits concerning time, costs, and quality comparing it with simultaneous engineering.
**Introduction**

The development of new products has become a focal point of international competition in many industries (Clark and Fujimoto 1991; Murmann 1994; Adler et al. 1995). The emergence of intense global competition, the creation of fragmented markets, and rapidly changing technologies force companies around the globe to develop higher quality products faster, and more effectively (Clark and Fujimoto 1991).

Intense internal competition has emerged from globalization of manufacturing and distribution due to faster transnational flows of information, material, and money (Gehani 1992). New competitors enter the market and increase the competitive pressure; e.g. in the high-performance segment that was traditionally dominated by Mercedes-Benz, BMW, and Porsche, today Japanese companies offer highly competitive and successful cars, such as Toyota’s Lexus, Honda’s Accura or Nissan’s Infiniti.

Customers have become more sophisticated, demanding more for less – more technology, more features and differentiated products, all at lower price. By reacting to the demanding environment, the industry has created new market segments and developed new products even for small niches. The creation of fragmented markets has lead to a compression of product life cycles and longer pay-off periods (Bullinger et al. 1992).

Intense international competition force companies to develop higher quality products faster, and more effectively. There have been several attempts achieving a shorter project durations and two of those shall be compared in this paper. Simultaneous engineering (SE) with its phase overlapping strategy will compete with theory of constraints’ application to project management, called critical chain project management (CCPM). Both concepts for sure are capable of reducing project durations,
but which one is performing better concerning all project success, that are time, cost and quality. All three performance measurements will be targeted, because having an eye only on the time aspect, will not make much sense, because a time advantage might just be traded off in disadvantage to the other important factors of a projects success.

First, I will shortly present the fundamentals of both strategies and then introduce System Dynamics, as a modeling technique, that is capable of revealing the differences. The example project shall be a product development project, because the introduction of CCPM has been conducted on development project as well as the framework I will be using for my understanding of simultaneous engineering.

**Simultaneous Engineering**

A central characteristic of a simultaneous engineering process is the overlapping of activities, because parallel execution of tasks will help to achieve short time-to-product (Womack et al. 1990; Krishnan et al. 1997). To manage concurrent product development successfully, managers must weight the gain from overlapping product development activities against the extra time and cost caused by downstream rework that results from proceeding in parallel based on preliminary information. Without careful management of the overlapped development process, the development effort and cost may increase, and product quality may worsen.

The explanation for this acceleration effect is simple. Instead of processing the project in a purely sequential fashion, one task at time (Takeuchi and Nonaka 1986) the development team works concurrently on several tasks. The interaction between the tasks facilitates communication and leads to compressed development cycles. Studies
by (Clark and Fujimoto 1991) and (Terwiesch and Loch 1999) have statistically confirmed the accelerating effect of overlap.

The sequential or PPP (Phased Project Planning) development process was originally developed by NASA to manage aerospace projects. In order to ensure error-free development, the project passes checkpoints at the end of each development phase and all items required by that checkpoint are checked, if necessary corrected, and released to the next phase (Smith and Reinertsen 1991).

Due to the parallel execution of tasks, concurrent engineering allows shorter cycle time, more flexibility and a higher degree of integration, but coordination and information exchange becomes more complex.

In a fully sequential process, as shown on the left side of Figure 1, upstream releases no information, until the developers have gained a relatively high level of knowledge about their task. When downstream activity begins, it works with finalized information, so that the likelihood of changes and iterations to occur at downstream phase is relatively low.

In case of an overlapped process, upstream activity sacrifices information quality to enable the “flying start” of downstream activity. As shown in Figure 1, downstream

Figure 1: Sequential vs. Overlapped Approach. Development of Knowledge Level over time.
developers have to base their work on medium level of upstream knowledge, which creates high risks of future changes and rework. Therefore the problem posed by preliminary information release is to manage the created uncertainty. The decision maker has to consider a main tradeoff: working sequentially and exchanging finalized information increases project completion time, but decreases the probability of rework due to changes in upstream work, whereas frequent exchange of preliminary information decreases completion time by enabling concurrent execution of tasks, but increases the risk of rework and cost escalation. If the change risks are great and downstream activity is sensitive to upstream changes, the benefit of overlapping, the potential time gain, may be totally traded-off against the time required for the additional rework to incorporate changes in the exchanged information. Due to the high complexity of the product development process, managers face problems in identifying when and how development phases should overlap and in predicting the impact on project performance.

A Conceptual Framework for Overlapping Activities

The System Dynamics Model, which is developed in this context, is mainly grounded on a work conducted by Krishnan et al., titled “A model-based framework to overlap product development activities” (1997). In this framework, two concepts determine the overlap trade-off: the rate of “Upstream Evolution” and the “Sensitivity of Downstream” activity to changes in the exchanged information.

Upstream Evolution

Upstream evolution refers to the refinement of the upstream generated information from its preliminary form to a final value (Krishnan et al. 1997). Krishnan distinguished
between fast and slow evolution. As shown in Figure 2, fast evolving information offers a given level of completeness earlier than in case of Slow Evolution, because major changes happen in the beginning of the upstream phase.

![Graph showing Upstream Evolution and Downstream Sensitivity](image-url)

**Figure 2: Upstream Evolution and Downstream Sensitivity**
(adapted from: Krishnan et al., 1997)

**Downstream Sensitivity**

The other dimension, Downstream Sensitivity, describes how much rework is caused by a change in upstream information (Krishnan 1993). Similar to Evolution, Downstream Sensitivity can be high or low. If Downstream Sensitivity is high, then the duration of the downstream work required to incorporate changes is high.

In the case of low sensitivity, substantial changes (in the exchanged information value used by the downstream phase) can be accommodated quickly by the downstream activity (Krishnan et al. 1997). Figure 2 illustrates the described cases of low and high sensitivity.

Krishnan provides a conceptual framework that describes the relationship between Upstream Evolution and Downstream Sensitivity and suggests different types of
overlapping strategies. By combining the two dimensions and taking slow and fast evolution, respectively high and low sensitivity into consideration, Krishnan constructed a 2x2 matrix. See Figure 3 for the four possible cases:

Figure 3: Types of Overlapping Strategies based on Evolution and Sensitivity

(adapted from: Krishnan et al., 1997, Figure 9)

1) **Iterative Overlapping**: Downstream activity may begin early with preliminary information, because downstream sensitivity is low, so that even if large changes occur in the exchanged information, their effect on downstream rework (duration) is low.

2) **Preemptive Overlapping**: In this case, downstream activity begins after receiving finalized information from the upstream activity. Due to the fast evolving information, upstream activity may freeze information early, because the expected quality loss is small compared to the better lead time performance.

3) **No Overlapping**: In this case, when the downstream sensitivity is high and upstream information evolves slowly, Krishnan dissuades from overlapping of
development activities, because it is neither desirable to begin downstream activity with preliminary information nor feasible to freeze information early.

4) **Distributive Overlapping**: This scenario is called Distributive Overlapping, because the impact of overlapping is distributed between the upstream and downstream activities. In such a case, the developers may choose between the iterative approach (as described in case 1) and the preemptive approach to overlapping (case 2). By analyzing the trade-offs among the performance parameters, the downstream activity may begin with preliminary information (then downstream effort/rework is traded-off against lead time) or decide to iterate once only, freezing the information early in the upstream process and creating a trade-off between upstream quality and lead time.

Krishnan discusses the application of the framework, using an electronic pager development process and two cases in automotive development, solving numerical examples (Krishnan 1993; Krishnan et al. 1997). In conclusion, he states: „It is generally the case that a fast evolution, low sensitivity situation is more favorable for overlapping“ (Krishnan et al. 1997).

**Critical Chain project management (CCPM)**

Goldratt is known in the operations management society for his introduction of the Theory of constrains (TOC), which was transferred to other management areas. Applying TOC to project management Goldratt (Goldratt 1997) established the Critical Chain project management. CCPM is introduced by Goldratt and supporting writers, e.g. (Leach 1999; Simpson and Lynch 1999) and is presented as an alternative to the
classical project management techniques. CCPM claims to be a revolutionary way of thinking and can lead to an improved project performance.

CCPM shortens project duration not by overlapping phases, but rather by sticking to the sequential approach and redefining how phases and tasks should be scheduled here. First an initial schedule for project tasks is developed, taking into account the dependencies among the tasks and resource availabilities. In this schedule the longest path can be identified as the critical chain. The next important step and the first one differing from classical approaches is to recalculate the tasks durations. The logic behind this can be summarized as follows:

- Most task durations underlie uncertainty
- When people are asked for their duration estimates, they add a safety margin, so that they can be sure to deliver on time
- In most cases not all of the safety margin will be used but is not passed to the succeeding tasks, so that time is lost whenever a tasks is completed early, because succeeding resources are likely not to be ready yet and next, in classical methods there are no incentives to finish early. Resuming: gains are lost and delays are passed over in completely

The CCPM suggests cutting tasks duration to its 50 percent likelihood of completion and in return time buffers with different functions are incorporated into the schedule. The extracted safety margins are pooled and placed at the very end of the project. This is called the project buffer and here the time advantage is gained, because the size of this buffer is only half of the size the sum of the safety margins extracted would be. About half the tasks will take longer than the estimate and here the project buffer can be absorbed. Pretty much the same scheduling happens to the tasks located on non-critical
paths. The safety margins are taken out and pooled in a feeding buffer. The feeding buffer is located before the non-critical path is converging into the critical path. This buffer's function is to protect the critical path. The third type of buffer is called resource buffer, which is used to preserve gained time. Whenever a task is coming close to its finish the succeeding tasks has to be notified, so that the succeeding resource has a few days to wrap the non-critical work and begin the critical tasks as early as possible.

At this point a new project schedule has been established by CCPM, which is working with the following principles:

- Resources assigned to a critical task are supposed to work on this continuously and finish as soon as possible
- If a task finishes early the successor can begin its work right away, because of the resource buffer
- If a task finishes late the feeding or project buffer will absorb the delay
- Project control becomes easier, because the buffer's statuses can work as warning signals and managerial action is only demanded when the buffers shrink too much

Despite the resection of CCPM there is also critic. For example (Hill et al. 2000) provides recent results with contradictory findings concerning the included safety margins in task duration estimates. Also critic the classical methods are often accused of apply to CCPM. CCPM takes a static view on the project and all the dynamics, non-linear relationships and feedback loop inert to a project are not taken into account. Also nothing is said about the other performance measurements, how this CCPM schedule influences quality and costs of product to deliver.
Next System Dynamics is presented as a methodology that has proven its applicability to project management and is capable of comparing a project that is once schedules with the principles of CCPM and once with an overlapping of phases as suggested by simultaneous engineering.

**System Dynamics for modeling product development projects**

The type of system I am examining, a product development process, is a highly complex system. It consists of multiple interdependent components, is highly dynamic, involves feedback processes, nonlinear relationships, and can only be sufficiently understand and described by investigating „hard“ and „soft“ data (compare Sterman’s definition of complex dynamic systems, (Sterman 1992)). The following four paragraphs describe these characteristics in more detail:

1) **Product development projects require the coordination of multiple and inter-dependent variables and components**

   To develop a new product successfully, a number of functional departments and experts have to collaborate intensively over an extended period of time. Such a project is complex, and may involve hundreds of people, working together over months (Clark and Fujimoto 1991; Smith and Eppinger 1997). Even though, I consider a strongly simplified situation of a development process, the modeled system still consists of a large number of variables that are related in time and space.

2) **Product development is dynamic and involves feedback processes**

   Product development involves multiple interacting feedback processes: Clark and Fujimoto argue that product development can be described as a „system of interconnected problem solving cycles“ (Clark and Fujimoto 1991). Related
research has confirmed the iterative and dynamic nature of such processes (Adler et al. 1995; Smith and Eppinger 1997). Moreover, there are multiple time delays in carrying out the development program, in discovering, processing and correcting errors, and in responding to changes in project resources and scope (Sterman 1992; Clark and Wheelwright 1993; Terwiesch and Loch 1999). I will focus on the important feedback relationships between the development activities. These activities are tightly coupled, and frequent exchange of information is required as the project progresses.

3) **Product development involves nonlinear relationships**

Product development involves multiple nonlinear relationships, that is, cause and effect do not have simple, proportional relationships (Sterman 1992). For example, consider the relationship between changes in the upstream information, and the downstream work required to incorporate the change. Almost universally, changes in information become more expensive and harder to include, the later they are implemented (Loch and Terwiesch 1999).

4) „**Hard“ and „**Soft“ data

To understand the dynamics of product development, it is not sufficient to merely consider the technical dimensions of such projects. Product development is essentially a human enterprise. Therefore, project organization, communication between different functions, and managerial decision making are important determinants of the project’s evolution. (Abdel-Hamid and Madnick 1989; Clark and Fujimoto 1991; Abdel-Hamid et al. 1993; Clark and Wheelwright 1993).

System dynamics is capable of representing systems with these characteristics (Sterman 1992). The system dynamics methodology was developed by Jay Forrester in the 1950s.
at MIT (Forrester 1961). It studies how the structure of a system drives its behavior, and describes the cause and effect relationships with stock, flows and feedback loops. The system dynamics approach models explicitly the decision making process, and studies the impact of policies and rules, depending on the system conditions (for definitions of system dynamics see (Forrester 1961; Sterman 2000)).

Previous work in the field has shown that system dynamics modeling is capable in improving the management of product development projects. System dynamics models have been used to manage projects more effectively and to assess sources of cost and schedule overruns (Cooper 1980; Abdel-Hamid and Madnick 1991; Williams et al. 1995). (Sterman 1992) comes to the conclusion that “System dynamics has the most highly evolved guidelines for the proper representation, analysis, and explanation of the dynamics of complex technical and managerial systems”.

**Central Assumptions of the project model**

Based on the system dynamics methodology and in the course of my research work a product development project simulation is developed. The model is a set of nonlinear differential equations, describing the flow of work and information within the project and the policies used to manage the exchange of information.

Since I am interested in the simplest possible model that can capture the previously described management approaches, I investigate the interaction of only four phases in the development process, information supplying upstream activities, and the information absorbing downstream activities. Each phase is represented by a generic structure, which is customized to reflect a specific stage of product development. For
example, consider two major development stages, such as concept development (upstream activity) and product design (downstream activity).

These activities can overlap or be scheduled in a CCPM manner to minimize development lead-time. The upstream activities have to resolve complex engineering problems and the associated information flows to the downstream activity. The release of preliminary information by upstream activities allow downstream activities to start before the upstream activity is completed (Krishnan et al. 1997).

Based on Krishnan’s assumptions, a situation of sequential dependence is considered where frequent information exchange between the upstream and downstream activity enables the concurrent execution of coupled activities (Krishnan 1993; Krishnan et al. 1997). That is, the development process is constraint by physical and information relationships among the tasks and phases. In the model I consider task durations and also include information dependence relationships (Ford and Sterman 1998).

The amount of downstream rework required to incorporate the changes in upstream information is determined by two factors: First, the size of the change between the new information and the previously released information, and second, the state of downstream work progress. Before downstream activity may approve and release its work, upstream activity has to finalize („freeze“) its work (Krishnan 1993).

The progress of a development phase and the generation of information is reflected in the completion of development tasks (Ford 1995; Krishnan et al. 1997).

Within each phase there is a set of tasks to be completed. These development tasks are the fundamental units that flow through the project (Ford 1995). Based on the description provided by Ford (1995, p. 30), I define a task to be „an atomic unit of development work“.
Changes and rework are portrayed by closed-loop iterative flows within a development phase. I distinguish between internal and external changes. As previously noted, external changes are generated when tasks in the downstream phase are attempted even though the upstream information needed to complete them fully is not known or imperfect. As this information becomes available, downstream tasks must be repeated. In contrast to external changes, internal changes are generated by the development team within their own phase. Internal iterations are required to correct errors, generate needed information or improve performance when the developed solution/design fails to meet the existing specifications (Smith and Eppinger 1993).

**The model’s structure**

The work process structure represents the core of the product development model, describing the flow of work within the development phase (see Figure 4 below). The structure and formulations are based on the descriptions provided by (Ford 1995) and (Ford and Sterman 1998). The work process structure is the most important out several modules the model consists of. The other modules (which are internal errors, external errors, information evolution, learning and experience, productivity) interact with the main module.

The development progress within a single phase is characterized by five states. In each phase tasks can either remain to be completed (Tasks not Complete), have been completed but require inspection (Tasks Completed Not Checke), have been inspected and require rework (Tasks To be Change) or have successfully passed quality assurance and have been approved, and finally, released to downstream phases (Tasks Release). Figure 4 shows the states and how they are related in the form of a stock and flow diagram. For easier understanding one can imagine a desk where work is coming in and
put the folder “work to do” and from there it flows through the other folders there might be, e.g. “tasks released” or “tasks to be changes”.

Three development activities determine the flow of tasks among the different states: completion, quality assurance and change. Completion is finishing a development tasks the first time. The quality of the completed tasks is checked and tested through quality assurance efforts. Tasks may require changes because they were done incorrectly or because the information they were based on has changed. Tasks found to be flawed are modified by the Change activity, i.e. errors are corrected or the quality of tasks is improved.

To get an idea of the project’s dynamics Figure 5 shows the feedback loops which drive the development process. A positive, or reinforcing loop (denoted by R) amplifies changes with even more change. This can lead to rapid growth, or vice versa,
to rapid decline. A negative, or balancing (B) loop seeks a goal and stabilizes the system (Richard and Pugh, 1981).

![Diagram of the Development Process Feedback Structure](image)

**Figure 5: The Development Process Feedback Structure.**

**Results from simulation**

In this paper only the very core results from the simulation findings will be presented. Figure 6 shows the accumulations of released tasks of the two management approaches. One can see, that the SE project finishes at a little over 280 days while the CCPM

---

1 Causal diagrams provide a compact representation of the interdependencies in a system and portray the feedback structure. Arrows and a sign (either + or -) indicate the causal direction and polarity of a link between two variables: a “+” denotes that an increase in the independent variable causes the dependent variable to increase, ceteris paribus (and a decrease causes a decrease). Similarly, a “-” indicates that an increase in the independent variable causes the dependent variable to decrease. Note that causal loop diagrams are not intended to provide mathematical specification of the relationships, which may be linear or non-linear, or of any time delays between cause and effect.
managed project comes in at about 330 days. But it is to consider, that all CCPM phases only needed approximately 65 work units until the goal of each phase was reached. Very different from that the SE phases needed 80 work units until completion.

**Figure 6: Comparison of a CCPM and SE model run**

![Tasks Released with CCPM](image)

![Tasks Released with SE](image)
Quality aspects are left beside in these runs because both project had to work until the promised specifications were reached, so that both delivered a “perfect” product. For a better understanding how these results were established one has to take a look at a lot more simulation graphs, which will be subject to the presentation at the Second World Conference on POM, but is you are besides that interested in the model, which is simulated and modelled in Vensim, please contact me under my e-mail address.
Literature


