CENTER FOR QUALITY OF MANAGEMENT JOURNAL

REPRINT NUMBER:
RP10000

Cycle Time Reduction Special Issue

Volume 8, Number 2 **Autumn 1999**

ISSN: 1072-5296

This page left intentionally blank
for purposes of duplex printing.

Observations from the 1997–98 CQM Study Group on Cycle Time Reduction

Compiled by Neil Rasmussen and David Walden from the work of the entire study group

1. Formation of the Study Group on Cycle Time Reduction

Neil Rasmussen is co-founder, Vice President, Chief Technical Officer and a Director of American Power Conversion (APC). As the senior technical executive of a high growth, high tech company operating in the global market, he is greatly concerned and involved with decreasing the cycle times of key business processes of APC.

•••

David Walden spent over 30 years in business, as a technical contributor, technical manager and general manager. In all three roles, he spent much time trying to get high tech projects done sooner and business processes to run faster.

Since its inception, the Center for Quality of Management (CQM) has regularly undertaken study efforts to address an expressed interest in a particular area by CQM member companies. A 1997 survey of member companies and CEOs revealed that cycle time reduction was a topic of great interest to CQM members. Thus, in 1997–98, CQM undertook a study group on cycle time reduction (CTR).

The importance of cycle time reduction

There are many reasons for CQM member companies to be interested in studying cycle time reduction, including:

- Customers like immediate response.
- There is pressure in many industries to introduce new products and services faster and faster.
- Reducing cycle time makes it necessary to predict less far into the future.
- There are usually lots of "low hanging fruit" — processes where a quick look will reveal many unnecessary or too long steps and delays that are easy to eliminate.
- Reducing cycle time typically results in simplifying a process and thus also helps with defect reduction. By eliminating steps, you eliminate places where mistakes might be made. Techniques such as design for manufacturing and design for assembly, or other mistake-proofing techniques, prevent defects from happening or uncover them earlier and in so doing reduce cycle time.
- Reducing cycle time often also reduces costs because reducing cycle time can reduce costly mistakes and rework and the less time that is scheduled, the less effort tends to be spent.
- Reducing cycle time tends to draw a company and its people toward other important improvement areas and techniques, such as customer focus, defect reduction, benchmarking, and the like.
- Cycle time reduction has potential to *make time available*. The first argument against any organizational change and improvement activity is, "there is no time to do more." Therefore, why not begin to phase in improvement activities by doing something to make time available?

We are not claiming that cycle time reduction is somehow the "one true place" to do improvement work. However, in the absence of reasons for focus in another area (such as defect reduction, customer focus or benchmarking), cycle time reduction is a high potential place to start improvement activity that deserves serious consideration.

Study group approach

Following the CQM's standard practice, the cycle time reduction study group consisted of individuals from member companies, university affiliates and from the CQM staff. The list of participants in the cycle time study group is included in Appendix A.[1] The scope of study included cycle time related literature, unpublished experiences and theories, company case studies, and consultant's techniques.

Study group members met twice a month for four hours from September 1997 to June 1998. They heard presentations from experts (see Appendix B), did individual reading and study and reported back to the whole group (see Appendix C), and circulated other papers and book chapters to each other to read.[2]

Roughly two-thirds of the way through the study period, Neil Rasmussen of APC and one of the study group members, gave an interim progress report at a public CQM roundtable meeting (available at http://www.cqm.org/whats_new/study_group_interim.htm).

[1] Each participant contributed to the study and each can be considered a co-author of this paper.

[2] About 100 documents were circulated altogether, a list of which is available to CQM members from the CQM office in Cambridge and on the CQM members only Web site (available at http://cqmextra.cqm.org).

2. Positioning of this Paper

The typical CQM study group attempts to go through the following ideal process:

- Gather and understand existing thinking.
- Compare, reconcile (particularly vocabulary differences) and integrate.
- Form hypotheses (including a step-by-step process).
- Attempt to validate hypotheses, through field trials.
- Draw conclusions and publish them.

Some CQM study groups have succeeded with this ideal process. For instance, from 1991–93 a working group developed the Concept Engineering method thatis now widely disseminated. Some CQM study groups have stopped their work drawing only the most limited conclusions, such as the study group on System Dynamics in 1996.

The study group on cycle time reduction resulted in something in between:

- Gathering and understanding much existing thinking.
- Comparing and reconciling, but not much integrating.
- Forming some hypotheses and drafting several versions of a "strawman" model, but not producing a step-by-step process.
- Individuals from the study group tried some of the methods in their own organizations; however, since there was no overall step-by-step process, a single overall method was not widely tested.

This paper takes no position about there being a single step-by-step cycle time reduction method that can be used in many different circumstances. Rather, this paper describes several key concepts that are broadly applicable and suggests a variety of methods that are applicable in different situations.

This paper is not a final report of the study group. It is not sufficiently comprehensive or representative of everything that was studied and considered by the study group members. Rather, the paper is a single, limited perspective on a subset of what the group considered.

The content of the paper is drawn primarily from an interim progress report on the cycle time reduction study group by Neil Rasmussen:

- the model for cycle time reduction Neil developed as he was participating in the study group and which he is trying at APC
- the materials collected during the course of the study
- a presentation on the study group's activities that David Walden has given several times at CQM chapter meetings.

Ideas and experiences from a few other sources not covered by the study group are included at relevant points in the paper.

There is not space in this paper to give a complete basic introduction to cycle time reduction. Appendix D includes a short list of valuable references on cycle time reduction. Reading this set of books will move most new students of cycle time reduction substantially toward the realm of what experts know.

3. Some Study Group Observations Relating to Cycle Time Reduction

Observation: On the relationship between cycle time and throughput

The most basic observation to make — in case it is not clear to some readers new to thinking about cycle time reduction — is the relationship between cycle time and throughput. Generally speaking, throughput is the reciprocal of cycle time: as the cycle time of a process goes down, the throughput of the process goes up; as it takes less and less time to make each unit, more and more units get made in a period of time. This relationship is shown in Figure 1.

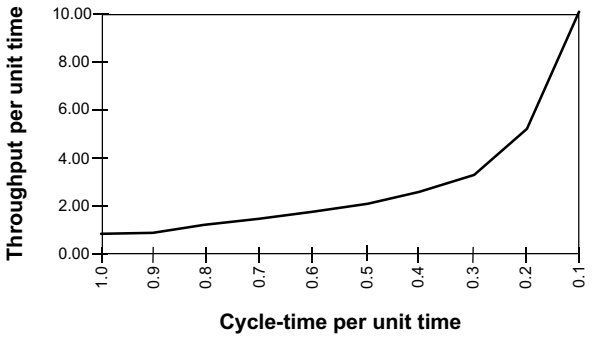

Figure 1 illustrates the general relationship between cycle time and throughput.

In reality, the relationship is more complicated than shown in the figure. A mix of project or subprocess sizes (some short, some long) may somewhat change the relationship from the above curve. Shortening the cycle time of some projects or subprocesses makes a big difference in overall throughput, while reducing the cycle time of others might be counterproductive overall. Such issues are discussed below. Nonetheless, in the small, reducing cycle time in a specific part of a process or project increases throughput through that part of the process or project. Whether or not this is a good thing to do to maximize overall throughput is another issue.

Observation: A preliminary model for cycle time reduction

Relatively early in the work of the study group, members derived a tentative model for processes for which high throughput is desired. This model is shown in Figure 2.

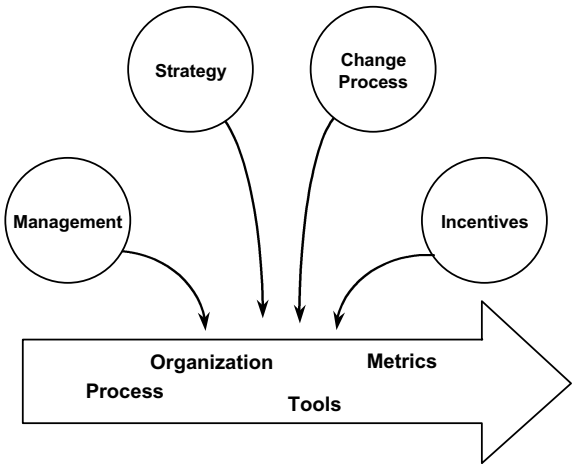

Figure 2 represents the initial study group model of the throughput process and its environment.

This model had an important role in the study group's deliberations. Section 3, however, will introduce an alternative model.

Observation: A comprehensive treatment of cycle time reduction does not currently exist

As the study group did its initial work, it quickly observed that many different people had done much work on the subject of cycle time reduction. At first glance, it appeared that different experts often had conflicting points of view; and some experts sometimes looked like they were claiming they had "the one true method."

However, upon closer examination (especially after reconciling vocabulary differences), it became clear that the experts are not actually in so much conflict. Rather, different experts observe cycle time from different perspectives and thus focus on different aspects of the subject. (It's like the old story of the blind men and the elephant.) Furthermore, none of the existing work available up through the period of the study fully integrated the best of the available ideas on cycle time reduction, not even providing a simple encyclopedia of available methods.

Thus, in the spring of 1998 Neil Rasmussen laid out a chart (see Figure 3) showing the areas addressed by different experts and authors working in the area of cycle time reduction.

	Strategy	Manage-ment	Change Process	Process	Metrics	Org	Tools
Rienertsen				+	+	+	++
Goldratt	++	+		+			
Clark & Wheelright				+		++	
Rummler Brache	++		++	++			
Goldense					++		
Hammer	+	++	+	+			
Meyer	+	++				+	
TQM			+	+			++
PRTM	++		++	++	+		
Uchimaru				+	+	+	++

Figure 3 is a chart of experts and their areas of cycle time expertise. (Key: + means "relevant to" and ++ means "very relevant to.") It would be useful for someone to update and expand this chart.

Observation: Processes vary in how consistently they are operated from one cycle to the next

Many people think of some processes as being repetitive, that is, they operate almost identically from one process cycle to the next; and they think of other processes as non-repetitive, that is, they handle varying inputs and produce a variety of outputs each time they operate. Our viewpoint is that essentially all processes we have in business are repetitive in the sense that they all repeat. However, processes vary in the degree to which they operate consistently from one cycle to the next. For instance, many manufacturing processes consistently do the same thing thousands of times a day. On the other hand, a process like architectural design of the building follows the same general outline from one cycle to the next and may use some detailed subprocesses in essentially identical ways from one cycle to the next; however, there is also much variation from one cycle to the next, and each cycle produces a different output (a different building design).

Faster cycle time processes (for instance, those happening many times per day), necessarily or as a result of having lots of practice, operate more consistently. The reverse of this also seems to be true: more consistently operated processes are more amenable to systematic cycle time reduction efforts.

Some examples of more consistently operated processes include manufacturing, order fulfillment, invoicing, and financial reporting. Some examples of processes that are often less consistently operated include business planning and new product development.

Some examples of processes that vary widely from company to company in how consistently they are operated include sales and service.

Many participants in the study group were more interested in cycle time reduction in new product development than cycle time reduction in any other area. Thus, a number of the descriptions in this paper refer to new product development even though the methods are generally useful more broadly.

Observation: To reduce the cycle time of less consistently operated processes, operate them more consistently.

Cycle time reduction for consistently operated processes is relatively well understood. These are amenable to well-known Total Quality Management techniques, such as those made famous in Toyota's production system. Nonetheless, there are some special twists relating to cycle time reduction that are not well understood in many companies.

A key to cycle time reduction for less consistently operated processes is to operate them more consistently. The point is that you cannot improve the intangible. PDCA[3] does not work well unless you have a system or process for it to work on. Furthermore, having a system is as much the basis for skill development as it is for time reduction.

Any activity can be turned into a process, more or less, not just those that operate step-by-step such as manufacturing and administration.[4]

[3] Deming's Plan Do Check Act cycle for process improvement.

[4] See "TQM in Service: A Report by the CQM Study Group," Victor S. Aramati and Toby Woll, *Center for Quality of Management Journal*, Vol. 6, No. 2, pp. 5–25. (Article available online at: http://cqmextra.cqm.org/cqmjournal.nsf/issues/vol6no2.)

4. Introduction to Neil Rasmussen's APC Model for Cycle Time Reduction

Rummler and Brache, in *Improving Performance: How to Manage the White Space on the Organization Chart*, make the point that one needs to know what one wants. One should not simply dive in and do cycle time reduction. This advice is in contrast to some of the hype about cycle time reduction that suggests that if one concentrates on cycle time reduction, other improvements (such as quality, cost, customer satisfaction) will come for free. The fact of the matter is that cycle time reduction may not be superior or even appropriate for every company. Different companies have different needs, perhaps even different needs at different times. Before embarking on a cycle time initiative, a company should make sure this is what it needs most.

The model being used at APC for cycle time reduction includes discovering the context for doing a potential cycle time reduction project, to discover that it in fact is what is needed. The APC model also includes a supportive environment for cycle time reduction. This model is shown in Figure 4.

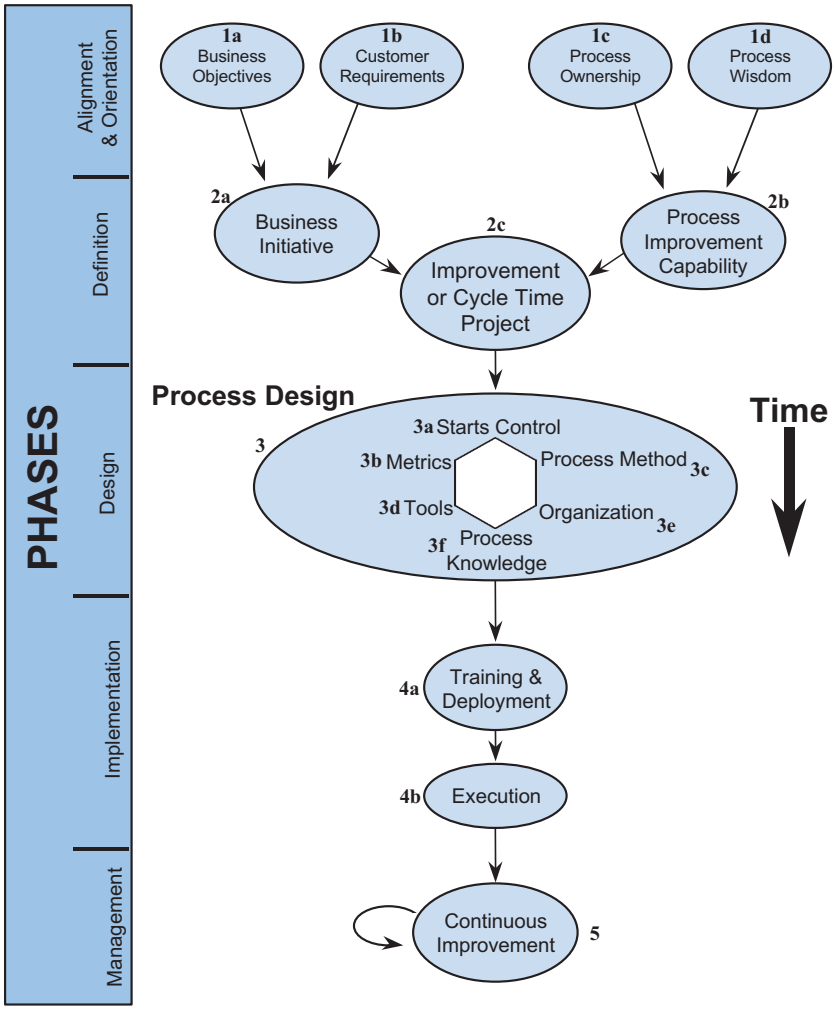

Figure 4 illustrates the APC Model for Cycle Time Reduction.

In this section we mention all phases and sub-phases of the APC model. The model was developed to manage any kind of process improvement project, but for the purposes of this paper discussion of the model will focus on cycle time reduction. Here in Section 4 we only provide detail on

those portions of the model that are more or less independent of the type of improvement being undertaken (that is, independent of cycle time reduction). These are Phase 1 except sub-phase 1d, Phase 2, Phase 4 and Phase 5. In the Section 5 we provide more detail on cycle time specific issues for sub-phase 1d; in Section 6 we provide more detail on cycle time specific issues for sub-phases 3a to 3f.

Phase 1. Alignment and Organization

The top management of an organization needs to be sure that the Phase 1 elements (Alignment and Orientation) are in place. Otherwise, cycle time reduction efforts are likely to fail.

Business Objectives (1a in Figure 4)

There needs to be a business reason for undertaking improvement efforts. In other words, there needs to be more than just a simple declaration that "we need to do cycle time reduction."

It is particularly important that the improvement work to be undertaken is aligned with important business objectives, or else the improvement work will be starved for resources. Resources are not given to projects that do not address important business objectives.

Cycle time reduction may or may not be a primary business priority. Defect reduction, cost reduction, or better understanding of customer needs are types of improvement work that might have higher priority than cycle time reduction.

Also, one must always beware of improving a process that should instead be out-sourced.

For more on this topic, see Rummler and Brache's book (listed in Appendix D) which includes an excellent discussion of Business objectives.

Customer Requirements (1b)

One also needs to have alignment with customer requirements. However, one must know the difference between what looks like a cycle time reduction requirement and what actually is one. For instance, it is very common that a company thinks reduced cycle time is needed when a customer complains about late deliveries. However, a careful examination of a customer's situation often reveals that what the customer actually wants is for the promised delivery date to be met exactly, and the customer is less concerned about whether the promised date is a little later or not. The customer often has other activities that have to be carefully synchronized with the promised delivery, including site preparation, hookups to related systems and equipment, scheduling downtime for normal activities, and so on.

Finding the appropriate customer requirements does more than just let an organization do what the customers want. Almost more important is the alignment and motivation an organization can gain from customer input and pressure.

Every business has two fundamental problems: figuring out what the market wants, and aligning and mobilizing the organization to actually accomplish it. Both are hard. However the latter is often more difficult. There are numerous internal agendas and opinions about what customers *should* want, making internal agreement impossible. By collecting data from the market and customers and working together to analyze it (for instance using a method such as Concept Engineering), the relevant people in the organization are more likely to develop similar views about what is needed and the integrated vision of what the customers want can be used to keep activities on course.[5]

[5] Gary Burchill's PhD thesis — *Concept Engineering: An Investigation of TIME vs. MARKET Orientation in Product Concept Development*, MIT, 1993 — provides a theoretical basis for how emphasizing the market portion of time-to-**market** tends to result in more internal alignment and thus quicker time-to-market than does emphasizing the time portion of **time**-to-market.

Process Ownership (1c)

A company typically has somewhere between a half-dozen and a dozen major business processes, such as product development, order acquisition, production, and customer support. Each of these spans various functional organizations and bits of organizational hierarchy. Thus, anyone trying to work on one of these typically is doomed to failure because no one owns the whole business process. The attempt to improve it runs into numerous political and structural roadblocks. A primary message from Michael Hammer (of Business Process Reengineering fame) is to have process owners for these key processes, so there is someone who can make the necessary trade-offs.

For instance, the president of High Voltage Engineering, based on his study of the methods of Rummler-Brache and the methods of Michael Hammer, required that each of his half-dozen business units organize around and have a person in charge of at least the following four key business processes: integrated product development, customer acquisition, order fulfillment, and customer service.

Process Wisdom (1d)

Many people work full time in an area and can execute reasonably well, but that doesn't necessarily mean that they have process wisdom. For instance, each of us walks fluidly every day. However, that doesn't mean that we actually understand the physiology and mechanics of walking and are qualified to undertake a project to improve the efficiency of how we ourselves or someone else walks. Similarly, we all spend much of our life in queues, but that doesn't mean we necessarily understand queuing theory and are qualified to try to change a process to speed up throughput through a network of queues. Wisdom is needed beyond conventional or common sense wisdom, which all too often is flawed or completely wrong. Later in this paper we will provide several examples where common sense about cycle time reduction gives inferior results.

Wisdom comes in two forms:

- About improvement processes in general, for example, making processes visible, having metrics with which to measure what is happening in a process and its results, PDCA, running controlled experiments, and so on.
- About the specific type of process you are trying to improve, for example, cycle time reduction.

Section 5 contains extensive discussion of process wisdom relating specifically to cycle time reduction.[6]

[6] The content of Section 5 might fit more logically at this point in this paper; however, it is so extensive that it might swamp the rest of the description of the model, which is why we present it in the next section. The reader optionally might read Section 5 now.

Phase 2. Definition

The top management of an organization needs to be sure that the Phase 2 elements (Definition) are in place. Otherwise, cycle time reduction efforts are likely to fail.

Business Initiative (2a)

For sufficient business reasons, a company or other organization should undertake a business initiative. What the initiative will be is driven by business objectives and customer requirements.

An "initiative" can be thought of as being different than a "project." An initiative is a broad effort to improve a company's overall capability in a certain area. For instance, a company might undertake an initiative

throughout the company to listen better to customers, decrease product defects, or decrease cycle times. Motorola's six-sigma program, which has now spread to many other companies, is an example of an initiative.

Initiatives serve to help create alignment in an organization. They are in essence the clearly stated strategies by which an organization achieves its business objectives. Initiatives provide the impetus for process improvement projects, and help provide a priority order among an organization's alternative improvement projects.

Communication of the initiative around the organization helps ensure that resources will be aligned in support of the process improvement effort. Complete alignment of the management team in support of the initiative is essential.

As mentioned under sub-phases 1a and 1b, it is important that this initiative be the right one. Cycle time reduction is not automatically the initiative an organization needs.

Process Improvement Capability (2b)

One needs a capable team of people to carry out the improvement. A capable team consists of people who are sufficiently engaged to get the job done, have the skill and wisdom to do the job, and have the authority to do the job. (This is a fairly good operating definition of being "empowered.")

A process owner needs to ensure the team has the engagement, skill and wisdom, and appropriate authority to do the improvement work successfully.

Improvement or Cycle Time Project (2c)

Once the elements of Phase 1 (business objectives, customer requirements, process wisdom, and process owner) have been addressed and in turn the Business Initiative (2a) and Process Improvement Capability (2b) elements are in place, we are ready to clearly define the project (or projects) required to improve something, and that something may be cycle time reduction.

For instance, we might decide to improve the cycle time of our new product development process, or we might decide to improve the cycle time of our order acquisition process. The definition of the project would likely include: schedule, staffing, metrics, current values of metrics (if available), expected improvements, budget, customer requirements, outline of approach, and so on.

The improvement effort is going to result in a changed process design, for example for new product development; and that process design needs to deal with several sub-elements, such as those enumerated in Phase 3.

Phase 3. Process Design

The CQM Study Group on Cycle Time Reduction initially concluded that the relevant subcomponents of a good process were the four elements — metrics, process method, tools, and organization — shown in the Initial Study Group Model in Figure 2 and shown at the sides of the Phase 3 hexagon in the APC Model (labeled 3b—3e in Figure 4). However, from the CQM study and his own work, Neil Rasmussen concluded that two other important elements were missing from the study group's first formulation — starts control and process knowledge (3a and 3f in the APC Model shown in Figure 4).

Once the design for the improved process (for instance, a process for reduced cycle time in new product development) has addressed (at least) these six areas, then the process being improved will be applied repeatedly (for instance, to each product development project).

Here in Section 4, we give a one-sentence explanation of each of the six areas of process design which must be assessed, documented, and, as necessary, modified or completely redesigned. An effective process design should address all these areas. Section 6 includes detailed discussions of each of the six design elements.[7]

[7] The detailed description might fit more logically at this point in this paper; however, the examples are so extensive they might swamp the rest of the description of the model, which is why we present them in Section 6. The reader optionally might read Section 7 now.

Starts Control (3a)
Methods to prevent starting more projects than the organization has resources for, or can coordinate and manage efficiently.

Metrics (3b)
Ways to measure what is happening, what drives what, and what works and doesn't work.

Process Method (3c)
The process that is going to be followed, for instance, the new product development process.

Tools (3d)
Design aids, monitoring equipment, software, templates, forms, and other devices and methods that can be used in the process.

Organization (3e)
Reporting structures, roles, how teams are formed, etc.

Process Knowledge (3f)
Mechanisms for capturing new insight about the process and making it accessible to future users and improvers of the process.

Phase 4. Implementation

Training and Deployment (4a)

Once a new or improved process has been created (for instance for new product development) then people have to be trained in the method and it has to be deployed throughout the primary organization concerned with it (R&D in the case of new product development) as well as throughout the rest of the organization as appropriate (e.g., manufacturing, sales and customer support). There is a dearth of description of good practice in this area.

Execution (4b)

Finally, one is ready to use the process.

Phase 5. Continuous Improvement

Continuous Improvement recognizes that no process is ever right the first time, and, even if it is right, circumstances change. The initial process design, in fact, is only a theory about what should work. The real learning about what actually does work comes from trying the process and then adapting it.

Phase 5 typically applies to Phases 3 and 4 (although it would seem that the initial Phase 1 and 2 work, particularly relating to having a process improvement capability, should also have enabled the on-going Phase 5 continuous improvement work). The Phase 3 process design and the Phase 4 training and deployment system should also have been

designed to enable on-going improvements and their deployment.

In this continuous improvement phase of the model, the purpose of the process improvement team is to focus on business results of the newly updated process. Changes to the process in this phase are reactive in nature based on real data coming in from the execution of the process. CQM's 7-Step reactive improvement process is typical of a method used to respond to defects or roadblocks in the process.

Every company that undertakes improvement of a key business process reports that they should have deployed it sooner, and thus gotten real world feedback sooner, and thus improved the process sooner. However, there is a natural human tendency to want to be sure that such an important process is exactly right before deploying it.

There is also a natural human tendency to want to put lots of controls on the change of such an important business process, so that it cannot be changed casually. However, since the process won't be exactly right upon first release and will need fixing, having lots of controls on changing it also convinces those that we want to use the process that the process exists in some sort of dream world that doesn't match real life issues. Thus, they may dismiss the process as irrelevant.

At APC there is a set of "process coordinators" who have been responsible for the design and deployment of their new product development process. Each of these individuals has the authority to change the process immediately. This helps convince the users of the process that it is relevant. When a user points out a problem with the process, if the coordinator sees the validity of the user's point, the coordinator can say, "You're right — let me change it." These process coordinators report to the senior engineering managers (who aren't authorized to change the process). Incidentally, the APC engineers spontaneously started a "users group" for their new product development process as a forum to discuss the process and provide feedback on it.

Application of this model more globally

As suggested earlier in this paper, sub-phase 2c of the model could be parameterized to apply to "Improvement of *any capability* in *any business process*, and the rest of the model would still apply. For instance, oval 2c could say "Improvement of Cost in Acquisition of Orders" or "Improvement of Quality in Production of Product."

5. Details on CTR Wisdom

This section provides details relating more specifically to cycle time reduction for sub-phase 1d of the APC model shown in Figure 4.

Neil Rasmussen has observed that significant understanding of the application area of improvement, in this case cycle time reduction, is needed. Common sense understanding is often misleading. In this section we present some insights into cycle time reduction that experts know but that many other people do not. An excellent source for this information is Reinertsen's *Managing the Design Factory: A Developer's Tool Kit*,[8] from which many of the concepts presented below are drawn.

[8] Donald G. Reinertsen, *Managing the Design Factory*, The Free Press, 1997.

CTR wisdom: Complete utilization and high throughput are incompatible

Optimization of utilization of the components of a system is incompatible with overall system throughput.

One of the most common business measures is resource utilization, that is, cost per unit; and many people try to drive that as far down as possible. This is shown in Figure 5.

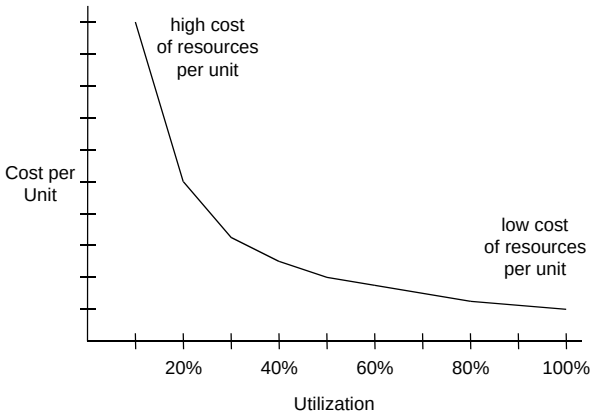

Figure 5 shows that Cost per Unit goes down as Utilization goes up.

However, queuing theory (and practical experience) tells us that as utilization goes to 100%, delay goes up rapidly. This is shown in Figure 6.

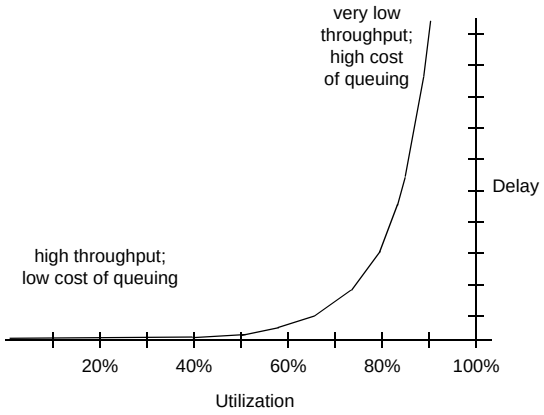

Figure 6 shows that Delay goes up non-linearly as Utilization goes up.

As the delay goes to infinity, the throughput goes to zero. The increase in delay toward infinity (and drop in throughput to zero) may begin with utilizations as low as 20% or as high as 65% to 80%, depending on the pattern of "arrivals" of work to be done, time distribution to do the work, whether new requests for work disrupt work already in progress, etc.

It may be counter-intuitive, upon first consideration, that delay does not rise linearly with utilization (or indeed that delay can get great before a system is overutilized). Consider the schematic diagram (in Figure 7, at right) of cars traveling in three lanes going in one direction on a highway and passing through an electronic toll booth that does not require them to slow up to deposit the toll.

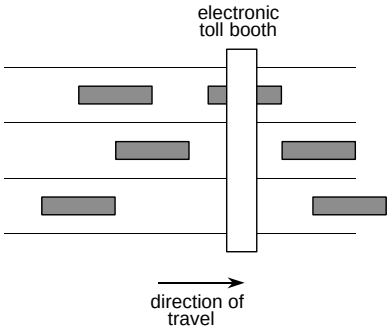

Figure 7 is adapted from Managing the Design Factory *by Donald G. Reinertsen.*

Assume the road can be used at 65 m.p.h. at 1/3 utilization, and we illustrate the average utilization by showing three cars in nine car lengths of roadway (3 lanes times 3 car lengths), as follows in Figure 8:

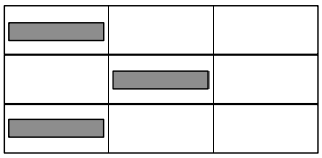

Figure 8 shows three cars in nine car lengths of roadway.

Now suppose one lane at the toll booth closes, and the three cars in the above figure now must fit into only two lanes:

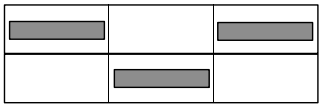

Figure 9 shows the same three cars, but now they must fit into only two lanes.

Utilization is now 1/2. However, even though 50% of the highway capacity is still unused, delay goes up and throughput goes down. We can try to intuit by how much: perhaps car speeds drop to 55 m.p.h., maybe less. Now, suppose another lane closes, and the three cars must fit into one lane:

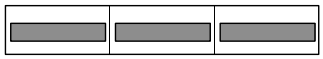

Figure 10 shows the same three cars, which must now fit into one lane.

The utilization is now 1. But what happens to delay? We all know the answer. Even though in theory the system is not overutilized (utilization is 1, not greater than 1), delay becomes very long and we are likely to have a multi-mile backup at the toll booth. Throughput on the highway goes practically to zero. The issue at hand is variation: car arrivals and passing through the tool booth cannot be synchronized closely enough to maintain full throughput (or anything close to it).

Combining our previous utilization and delay curves, Figure 11 shows what happens.[9]

[9] Figure 11 comes from page 48 of Reinertsen's book, ibid., although we have long understood it and could have drawn the figure ourselves.

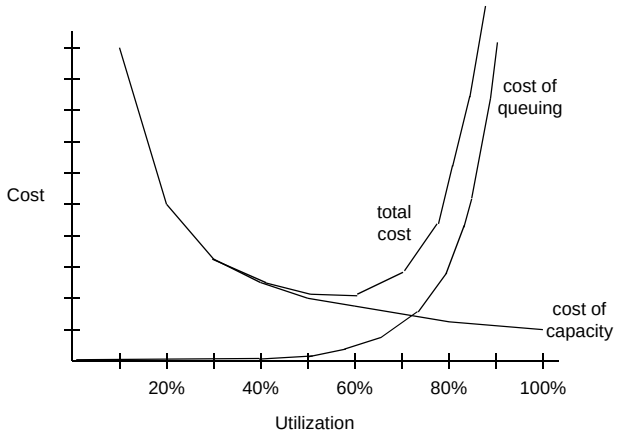

Figure 11 illustrates that the Total Cost is the Sum of the Costs of Capacity and Queuing.

As utilization rises, the cost of per unit or cost of capacity decreases. However, as utilization rises up, the cost of queuing rises. The total cost is the sum of the two component costs. In particular, full utilization is incompatible with high throughput and with low cost per unit processed. Therefore, the process designer needs to think about the best mix of cost

and throughput for the particular situation, e.g., somewhere between 60% and 80% utilization.

Queues are not always visible, but they always have a cost. Local optimizations (e.g., full utilization of components of the system) damage the cycle time performance of the overall system. Queues, not task performance, are the dominant factor influencing cycle time in most slow cycle time processes. Clearly, understanding of queuing theory should be a requirement for those attempting cycle time reduction.

There are methods of controlling the queues. Reinertsen's book and any queuing theory book list the following:

Increasing capacity relative to demand. Permanent increases in human or nonhuman resources may be possible either by simply adding capacity or by becoming more productive. There also may be ways of temporarily shifting such resources to a process when they are needed. Reinertsen lists the following general strategies for improving productivity: tools, support staff, reducing non-value added work, training, and cross-training.

Managing demand relative to capacity. Reinertsen lists limiting projects, limiting features, dropping requirements, and reuse as ways to manage demand in the area of new product development. More generally and dynamically, rules, charges or other mechanisms may be put in place to reduce demand during peak periods and possibly transfer it to periods of less demand (this must be done in a way that maximizes economic benefit).

Reducing variation in the components of the system (ultimately variation is why we have a hard time building systems without queues). One may attempt to "clock" requests for service and service times more rigidly. Reinertsen lists the following possible elements of attempting to reduce variation: collecting data on what is happening, reuse, process standards (such as narrow time standards for doing process sub-tasks), and reducing the batch size. For instance, if the airlines used more planes that were smaller, then there might well be less variation in the number of passengers queuing to get their luggage at a given time. At least, the big batches of service requests should not come before small batches of service requests. For example, landing a large plane immediately before a small plane results in all the passengers from both planes being in a long queue for luggage; landing the small plane first could result in at least the passengers from the small plane seeing a shorter queue.

Putting control systems of various types in place. One may install systems that attempt to react dynamically to changes in demand or service times, perhaps using combinations of the above methods. Reinertsen also mentions such strategies as monitoring queue lengths, moving queues off the critical path, reservation systems, planning more predictable queues, and reducing batch size.

If there is one message that anyone working on cycle time reduction should hear, it is: *study queuing theory*.

An understanding of queuing theory will provide insight into why we have many of the problems we have with long cycle times and thus will provide insight for what to do to correct the situation. Furthermore, monitoring queue lengths provides a *predictive* metric of long cycle times.

To repeat, study queuing theory! Queues are present not only in business process, but in everything we do. As the noted queuing theory expert Professor Leonard Kleinrock of UCLA observes, we basically spend all our lives in queues: "we arrive some place, hang around for a while, and then depart."

We have referred to Reinertsen's book extensively in this subsection, and will refer to his book and his ideas more in the rest of this paper. His book, of course, touches on many more topics and in much more detail than are mentioned in this paper, and the interested readers will do well to buy and study Reinertsen's book.

CTR wisdom: Project tasks seem inevitably to be finished later than planned

Projects (and subprojects) are never finished before their planned due dates, no matter how far out the due dates. This is shown in the pair of diagrams in Figure 12.[10]

[10] From page 18 of Reinerston's book, ibid.

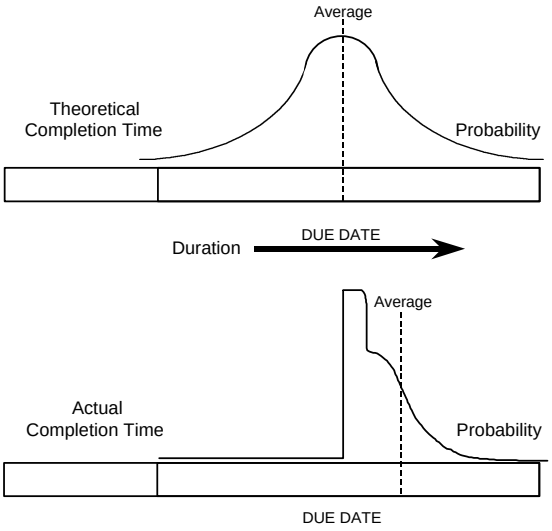

Figure 12 shows different viewpoints of the distribution of project completion time.

The top diagram in the figure shows the dreams people have about the due date of a project, i.e., that there is some sort of a distribution around the planned due date with some probability of finishing early and some probability of finishing late. However, we all know this is not reality. The bottom diagram in the figure is more realistic. There is almost no chance the project will be done early, and there is some distribution after the due date of actual completion. In fact, people usually schedule some slack time in their estimate of completion times for each component task to protect against trouble that will cause time delays, but that slack is typically used up and then some.

The slack time we put into a project schedule is used up in a variety of ways: multi-tasking as discussed in the next subsection, waiting until the last minute to start, not telling anyone in those rare instances when something is getting done early, etc. The later-than-plan completion time provides justification for the slack time we put into the schedule to guard against a time overrun, and the next time we remember the later-than-plan completion time from last time and may add more slack to that. There is pressure for the estimated completion dates of projects and their parts to get longer and longer.

The slack needed to protect against unexpected delays is usually put in each of the sub-tasks, as shown in Figure 13.

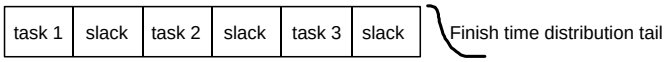

Figure 13 shows that slack is needed to protect against unexpected delays.

However, we have already pointed out that each bit of slack time usually gets used up, so the actual finish time is longer than the total of all of the estimated sub-task times and the slack time allowed for each. Slack needs to be eliminated from sub-task planning and aggregated in a few key places, and then be allocated to sub-tasks only as needed, as shown in the top half of Figure 14, below:

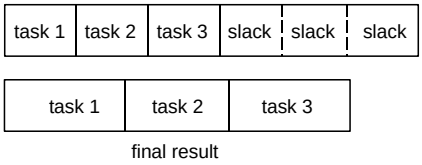

Figure 14 shows that slack needs to be eliminated from sub-task planning and put, instead, in a few key places; to be allocated to sub-tasks only as needed.

This might result in actual sub-task times as shown in the bottom half of the figure. Thus, we need mechanisms to help ourselves take slack out of sub-task planning and to clump it where it is really needed.

What we are discussing now is very closely related to Goldratt's concepts of Theory of Constraints as described in his books *The Goal* and *Critical Chain*. In *The Goal* Goldratt shows systems such as the following:

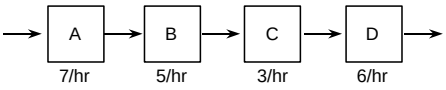

Figure 15 illustrates a system described by Goldratt in The Goal.

Missing a beat at A, B or D has little effect. C is the constraint. Thus, there is little benefit in running A, B or D faster than 3/hr. It only creates inventory. If C, however, misses a beat, the whole system slows down. Thus, a buffer is needed before C so C doesn't get starved for input:

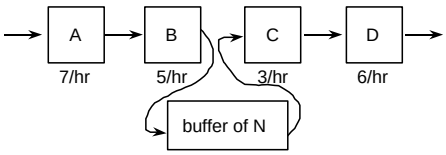

Figure 16 shows where a buffer is needed.

The buffer size N is determined by the statistical variation of the sub-processes in the system. Suppose we now double the capacity of C:

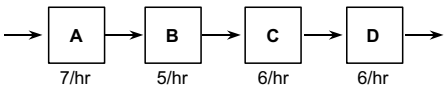

Figure 17 shows that we have doubled the capacity of C.

C is no longer the constraint. B becomes the constraint. Now we need a buffer before B:

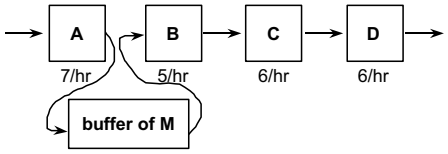

Figure 18 shows that we now need a buffer before B.

Of course, real life is more complicated:

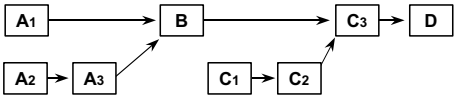

Figure 19 is an example of how this might look in real life.

Nonetheless, we can analyze the system and put buffering only in front of the few constraints in the system.

A parallel set of examples can be given where time, not inventory, is what must be buffered against. Goldratt describes this in *Critical Chain*. We must eliminate slack time (i.e., time buffers) everywhere in the system (because such time typically is irrevocably lost) and only put slack time at the few constraints in the system and then carefully control these time buffers. (Unfortunately, there is no good current way, politically and motivationally acceptable, to put all the needed slack time at the end or the few critical path points in the project where it does not inflate the time required for the sub-tasks but is still available if needed. To date each company, division or department that succeeds in doing this uses its own ad hoc methods — see the discussion of starts control below.)

CTR wisdom: Avoid multi-tasking

Timesharing resources among projects — having the resources, particularly human resources try to do multiple projects in parallel — is very often incompatible with fast program cycle time. Consider Figure 20 which shows different orders for completing three projects, A, B and C, each of which consumes 10 units of time to carry out.[11]

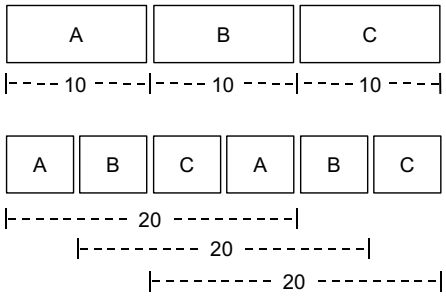

Figure 20 shows different orders for completing multiple projects.

In the top example in the Figure 20, the average time per project is 10 and the average finish time of the project is 20. In the bottom example in the figure, the average time per project is 20, and the average finish time of the projects is 25. As the time slice becomes smaller, the limit of each of these metrics moves toward the total time to finish all three projects. This does not take into account the added inefficiency of switching among and coordinating across projects which typically is explosively large.[12]

In many companies there is a lot of multi-tasking and sharing of people among projects. In some companies on average (or maybe even absolutely) there is no one working full time on many projects. In most companies, people are scheduled well over full capacity. As a result, project times extend drastically and throughput (completion of projects) goes down, down, down.[13]

As many of the most successful managers will tell you, the first step in controlling an out-of-control project completion situation is to stop over scheduling people and to stop sharing them among projects. Projects will start to get done, and overall project completion throughput will go up even though at any one time you are trying to do less.[14]

[11] Figure 20 comes from page 126 of Goldratt's *Critical Chain*. Although anyone could derive this figure, we credit it to Goldratt to call the reader's attention to his excellent book.

[12] A more general version of Goldratt's argument sketched here may be found in *Product Development in Half the Time*, Preston G. Smith and Donald G. Reinertsen, 1st edition, John Wiley & Sons, 1991, on page 200. It argues that any time you dilute resources you cause economic damage. This dilution of resources can come from multi-tasking the same resources or from supporting too many projects at the same time, even when they are fully staffed.

[13] Note, we are not arguing against doing projects or subprojects in parallel as in concurrent engineering. We are worried about time-sharing the same resources across several projects. Each project or subproject needs its resources to be immediately available when needed and not to be switched away from it at inappropriate times.

[14] Reinertsen points out (personal communication, October 1999), "There are certain cases where multi-tasking a centralized resource may give better performance than decentralizing the same quantity of resources to separate teams. When team demands for a skill are highly variable, aggregating demand streams from multiple teams produce a less variable overall work stream for a larger centralized resource. Therefore, delays can be larger with fragmented resources than with properly managed centralized resources. Thus, multi-tasking is not always a bad practice." However, we hasten to add that in our experience not dedicating resources causes problems much more often than maintaining a central resource causes problems. In particular, in our experience a shared central resource often is not "properly managed" — too often it over utilizes its resources in its attempts to satisfy competing demands.

Jeff Swift example from Analog Devices (ADI)

In an article entitled "Core Team Success at Analog Devices,"[15] Jeff Swift describes a situation in a division at Analog Devices.

The basic problem was that they were not completing new product development work as fast as they desired. Furthermore, when finally developed, the products were not meeting their market expectations. In trying to figure out what the problem was, they noticed that there was *much* sharing of people among different projects. In fact, on average no one was working on a project full time. Projects would be started as soon as anyone was available, people added later to a project would question the assumptions made by those who joined the project earlier, etc. All of these lengthened product development cycles.

Therefore, in this division they adopted a new, team-based, resourcing strategy. They adopted so-called "core teams" that consist of people (such as designers and test engineers) whose top priority is to the team's goals, and they are with the project from start to finish. (Extended team members, such as CAD or layout people, are with a project part time, providing service to a team or perhaps several teams and leaving a team when their services are no longer needed.)

With the core team approach, projects were completed faster and, in many cases, resulting revenue was better (indicating perhaps that getting to market faster improves revenue potential).

[15] *CQM Voice*, Volume 8, Number 2, Fall 1997 (the article is available on CQM's website at http://cqmextra.cqm.org/voice.nsf/issues/vol8no2swift).

6. Details on CTR Process Design

This section provides details relating specifically to cycle time reduction for Phase 3 of the APC Model shown in Figure 4.

Starts Control (3a in Figure 4)

The most important missing component from the original study group model (Figure 2) that is included in the APC model (Figure 4) is "starts control." Starts control is the system and criteria that allows a new project to be started — the system and criteria that limit project starts below a utilization level and avoiding too much switching among projects which in turn will result in maximum throughput. In other words, starts control addresses the process wisdom issues listed in the subsection immediately above.

For example, if we improve our new product development process, then the improved process should include a way of controlling when instances of developing a new product get started. Without such starts control, gridlock is possible: too many projects get started, which have to share resources too much, and thus get in each other's way, delaying all of them.

Unfortunately, the natural human tendency is to want to start more projects than can be successfully done at the same time. We can't bring ourselves to make choices. We convince ourselves that everything must be done, soon! We can't choose which projects are less important and thus can be delayed. We dream that timesharing among the available resources will get all the projects done in the same time period (and we then frequently dramatically over schedule resources, e.g., to the 200% level).

In some domains (such as new product development) starts control can be the single most important issue. Without starts control too many projects compete for available resources, there is too much switching among and coordination of resources, and throughput goes down.

Some consultants live off of this pattern of behavior, and make a big

difference in a company's performance simply by helping the company implement starts control for their key business processes.

Devices must be created for limiting starts. However, as soon as we say that, someone will say, "but in real life . . ." (implying that we are some sort of ivory tower theoreticians) ". . . we don't have the option of delaying key projects" (ignoring the fact that there will be more delay on average by trying to do too many projects than there will be in not over utilizing resources). As John Goodhue from Cisco (who spoke to the study group) says, "It is emotionally hard to think about deliberately making someone un-busy some of the time. Also, [in new product development] scheduling an engineer at less than 100% could cause good ones to feel unchallenged and to move on."

To repeat, devices are needed for limiting starts and avoiding overutilization and counterproductive multitasking. Seven detailed real-life examples follow:

Example 1

As part of the Analog Devices "core teams" approach (mentioned at the end of the previous section), Jeff Swift reported to a CQM study group a method they used for starts control in product development: a project could not be started until it was appropriately staffed with full-time people who have committed that this project will have first priority in their lives (thus avoiding counterproductive multitasking). This can delay a project start, sometimes for several weeks, while the project team sorts out its collective commitment and possibly replaces uncommitted people with appropriately committed people. It can also prevent a project from being started at all if the people needed by the project are not available.

Example 2

One business executive that John Goodhue and David Walden know "derated" head count by .8 for "overhead" when assessing the overall capacity of his organization across several projects over a span of 6—9 months. He then staffed projects in terms of the factored head count, rather than by individual names, until he had applied all his head count. By doing this, he guarded against overutilization of available resources.

Example 3 (and commentary)

John Goodhue informed the study group of a clever method Cisco uses as part of its starts control mechanism for product development. If a project cannot get itself fully staffed and approved within six weeks of conception, then it is insufficiently interesting to be undertaken at that time. Perhaps it is dropped permanently, or it is shelved "until it can be staffed or another way of accomplishing the purpose can be found (e.g., meeting a market need with a different platform or perhaps acquiring a company that already has the desired capability)."

Goodhue also notes,[16]

> "In his book *Managing the Software Process*, Watts Humphrey emphasizes making sure that all stakeholders in the project understand and are committed to the objective, so people don't slow things down by forcing unnecessary reexamination or rework to the objective later on. This is an essential component of the starts control approach used at Cisco, along with the resource considerations."

[16] Personal communication, October 1999.

Example 4 (and commentary)

John Goodhue also made the following suggestions about avoiding overutilization:[17]

> "Simplify the puzzle by focusing on critical path tasks. If one has a bottleneck on a noncritical path, one can manage around it most of the time. Also, avoid the appearance of scheduling someone to be "un-busy" by assigning to bottleneck resources a mixture of must-do and ok-to-slip tasks. For example, if one has a set of must-have and nice-to-have features in a software release, use the nice-to-have features to create slack [when needed] as any one [of the nice-to-have tasks] can be deferred to a later release if the person working on [the release] gets overloaded."

[17] Personal communication, January 1999.

Donald Reinertsen response. Donald Reinertsen expresses the following thoughts on John Goodhue's suggestions:[18]

[18] Personal communication, January 1999.

> "I think the big problem with allowing [engineers] slack time is that they will consume it polishing the design and you will not receive anything early. You need slack to prevent unexpectedly long tasks from unraveling the schedule but you can't allow unexpectedly short tasks to expand to fill available time.
>
> "To accomplish this you must do two things. First, you must establish a 'cultural norm' that an adequate solution delivered early is better than a perfect solution delivered late. One of the most important tools for influencing the trade off between performance and design elegance is to ensure the entire team understands the four decision rules for the project (especially the trade-off between performance and time on the critical path). Second, it definitely helps to have some background nice-to-do work. Such work needs to be important enough not to be viewed as busy work, but unimportant enough to be deferrable and easily dropped. I would suggest using activities that are inherently attractive to the individual concerned, because this helps them let go of their work product. For example, it is less likely that the engineer will continue polishing their design if the background task is to investigate a new technology than if it is to rearrange the engineering supplies cabinet.
>
> "You are, of course, precisely right about noncritical path tasks. You should only be willing to pay cost of delay for time on the critical path. Therefore a queue off the critical path is a 'free' queue. Of course, this illustrates why you need to know where the critical path is on every program. A more subtle problem is that of 'near critical path activities.' If you treat cost of delay as zero on near critical path activities they will be deferred until they are on the critical path and then you will have the opportunity to pay cost of delay for them. Clearly, they should be valued above zero but below the critical path cost of delay.
>
> "There is a rather important implication of this queueing discussion. If you cannot anticipate exactly when the queues will form when you plan the project, then you must dynamically monitor the size of queues on the project's critical path. Most development organizations monitor cycle times instead of queues and it is a much less effective measure because of its lagging nature. A good analogy is why Lucky's supermarket monitors the size of the checkout line instead of trying to forecast demand in advance.
>
> "A second important implication is that as long as you have queues you need a queueing discipline. The most naive sequence with which to handle work in the queue is FIFO. If you are smart enough to know

the cost of delay on individual projects you can use the most sensible queueing discipline: work on high cost of delay tasks first, make low cost of delay tasks wait longest."

Goodhue follow-up. Regarding Reinertsen's idea of a cultural norm that an adequate solution delivered early is better than a perfect solution delivered late, John Goodhue notes:[19]

> "... 'early not elegant' is the term that people use at Cisco to reinforce [this cultural norm], accompanied by a bundle of anecdotes that people tell about past products that were huge market successes in spite of competing products that had better technical specifications at the beginning. It is worth noting that this bundle of anecdotes includes stories with the opposite twist — competitors who beat Cisco by introducing products that initially had technical specifications that were inferior to what Cisco was willing to ship."

John goes on to speculate that the cultural norms and stories have more effect on the behavior of managers and senior engineers, and that "the urge to perfect takes precedence with most individual contributors (not entirely a bad thing — engineers are trained to get things 'right' for a reason). Enabling the desired approach via queue management and nice-to-do work is therefore critically important."

[19] Personal communication, January and October 1999.

Example 5

One division at Bolt Beranek and Newman Inc. (BBN) dealt with the issue of avoiding overutilization as follows. Resources were carefully allocated to projects so that no one was more than 100% allocated, in terms of 40 hour weeks. Then, unexpectedly high peak loads on a project could be handled with night and weekend work, and the project would still stay on schedule. If people are scheduled at substantially over 100% of capacity, then the nights and weekends to handle unexpected events are already allocated and the crush of also handling unexpected events will lower throughput.

Another variation of the last idea was used for the construction of a factory. A meeting of all subcontractors was held at the end of the day, every Thursday. If any subcontractor was not going to complete its scheduled work for the week, then it had to work over the weekend to catch up, so none of the work in the following week depending on that subcontractor's work would fall behind. The factory was completed on schedule.

Example 6

In another BBN example, one division was having trouble completing projects. Like all BBN divisions, this division used annual planning budgets that included detailed estimates of the costs and time to do approved projects. During the year, an elaborate hour-by-hour cost accounting system was used to allocate costs to each project. As in most companies, there was a certain amount of gamesmanship in the creation of the project budgets — low-balling estimates to avoid overruns and to protect against shifts in overhead allocations that were based on prorating actual project costs. And, as in most companies, there were many "budget vs. actual" variances throughout the year. Forecasting of actuals was really quite poor.

Therefore, this division switched to a system where the annual division budget was based on the level of effort available and, since these costs would be accrued on one project or another, overall end-of-year actual was close to budget. Projects were estimated simply in terms of expected level of effort to do the project (not low-balled, without concern for overhead, etc.). Projects were

undertaken based on benefit to the business or benefit vs. cost. The highest priority project was fully staffed first. The second priority project was fully staffed next. If there was not sufficient capacity to start another project, it was not started, and so they practiced starts control. When a project finished and capacity was available again, the next priority project was staffed and started. Detailed cost accounting was eliminated (which was a significant cost saving in its own right), as were overhead allocations; and an employee's entire cost during a quarter or month was assigned to the project he or she spent most of his or her time on.

This system had two results. First, a starts control mechanism was in place because new projects were not started until they could actually be staffed with specific people, and thus projects began to completed on time. Second, because there was no need under this new system to "game" the budgeting process, project forecasts vs. actual became more accurate despite using more approximate methods of cost accounting.

Example 7 (and commentary)

One of the compilers of this paper mentioned to John Goodhue of Cisco a method of *personal time management* that he derived based on insight from his work as a developer of data communication network systems (in which queuing is a paramount consideration).

> "I also used these methods in my person scheduling, for avoiding personal overutilization and thus avoiding becoming a bottleneck myself. On average I never scheduled more than one-half my time (my secretary who maintained my calendar enforced this ratio for me on a daily or at least weekly basis). Thus I had 50% of my time left to absorb unexpected things that came up during the day, and still had nights and weekends to get the original work done that took longer than its original halftime schedule. If I had scheduled 100%, nights and weekends would have been used up compensating for the unexpected, and then the originally scheduled work never would have gotten done.
>
> "Demands for scheduled activities beyond half of work week time were simply delegated to other people, who inevitably were happy to handle the tasks themselves without my involvement, just as I was when my boss left me to do something on my own."

This stimulated an interesting series of comments that follow.

John Goodhue responds. John Goodhue had the following remark on this method:[20]

> "I am beginning to realize that the most powerful bottleneck is the senior manager or management team. Your technique instinctively recognizes this by keeping a large amount of slack time to minimize the size of your queue. At the opposite end of the spectrum, the manager who uses the number or people waiting outside the door as a measure of self-worth is a destructive force.
>
> "A more damaging (and more common) situation is one where a well-intentioned manager focuses on throughput rather than queue depth. If I solve 100 problems but still have 10 on my queue at the end of the day I might feel like I accomplished a lot, but in fact I have constricted the organization. If I find ways to reduce the number of things that land on my queue in the first place and always clear my

[20] Personal communication, March 1999.

queue, then I have delivered more value to the organization even though I have done 'less work.'

"Using this vocabulary, one can recast several modern management buzzwords as ways to minimize senior management queue length:

- 'Empowerment' prevents stuff from getting onto the queue of a senior manager at all.
- Having a 'strong company culture' broadens the number of situations where a line manager or individual contributor can be confident that their actions and decisions are consistent with overall objectives, thereby keeping things off the senior management queue (complementary to empowerment).

"One can also recast various notions about decision making and time management as methods to reduce the time to process items on your queue, which in turn reduce queue depth:

- Making an 80% quality decision in 10 minutes vs. a 100% quality decision in two hours reduces queue depth.
- 'Management by wandering around' reduces queue depth by reducing the amount of time needed to communicate an idea or gather a piece of data (this notion was stimulated by 'What Effective General Mangers Really Do' by John P. Kotter in the March/April 1999 issue of *Harvard Business Review*). One can push this idea to unproductive extremes (there really is a time and place for formal meetings), but there are many cases where a two minute hallway conversation has the same impact as a half hour meeting.
- Living in a geographically distributed company, I find that phone mail and email can be leveraged in the same way (given sufficient face time and trust-building to back it up). For example, I have had fewer than 5 sit-down meetings with my marketing counterpart in the past 18 months, but an average of 5 phone mail exchanges per day, with drop-in meetings for face-to-face contact whenever we both happento be in the same city (we both arrive at the office in the early morning). Of course, one must be careful to match the medium to the individual with whom you are interactive. Some people do well with phone mail, some people do well with email, and others do best with interactive dialogue. Choosing the wrong medium and trying to make it work can be debilitating. Trust is also essential to efficient fast-paced communication. By trust I mean confidence that the person you are communicating with will behave according to the expectations set by the conversation."

Neil Rasmussen adds some thoughts. Neil Rasmussen of the CQM study group, had the following reactions to John's observations.[21]

"I have become convinced that the most fundamental problem driving queues and compounding the slack time management problem is what we at APC call 'starts control.' Starts control is the process by which project starts are created and managed.

"Starts exist overtly as explicit new product development projects, but also exist in hidden form as side projects, updates, enhancements, process improvement teams, research, systems deployment projects, facilities start-up projects, cost reductions, etc. Some of these projects get created in response to real customer satisfaction issues, but often they are created by management.

"Some of these starts are well meaning attempts at local optimization

[21] Personal communication, April 1999.

of resource utilization. However, it is difficult to get overall visibility to the total capacity requirements of these starts.

"Overdriving a system with starts ensures that there will be queues, partial people deployments, and no slack time. In such an environment, it is my assessment that attempting to manage the problem within projects is futile because even if gains are made, a broken starts control system will quickly recapture them.

"We are finding that a visible starts control process that includes the hidden starts is going to be necessary so that we can balance starts with capacity.

"Preliminary readings from our new product development process measurement system indicate that divisions that have projects with a staff-to-plan ratio of significantly less than one (indicating they are overloaded with starts) have the greatest schedule slip, even when all divisions are following the same new product development process.

"We are envisioning that a visible starts control process will allow managers to understand the consequences of the starts they create (hidden or explicit). Also, a visible system allows rational ranking and trade-offs to occur. We are developing systems to achieve this vision with some progress.

"If there is a silver bullet or any useful technique for program managers to use within a program to facilitate slack time management it is beneficial and we need an arsenal of these tools for programs managers. However, program managers are in an impossible situation when the overall system is overdriven with starts by management (or even by the customer)."

Goodhue continues. John Goodhue followed up Rasmussen's observations with "some reinforcing data and comparative notes from [his] own experience":[22]

[22] Personal communication, April 1999.

"In the business unit where I work, the vast majority of formal senior management attention (i.e. business unit GM, VP marketing and VP engineering) at the project level is focussed on starts control. When I first encountered this, I kept wanting to ask 'where's the rest of the development process?'.

"Over time I have been impressed by how effective this is at preventing trouble down the road, and the change that occurs when one institutes this approach.

"On reflection, it occurs to me that I never think about setting an operating point, only adjusting it (much like a phased lock loop). For example, with engineering groups, there might be one group where I am constantly pushing for more aggressive loading of engineers, another where I am constantly looking for and compensating for overload, and a third where we've crossed back and forth from too much load to not enough load several times.

"Though I haven't though about it this way before, I'll speculate here that it may not really be important to measure how heavily loaded people are. Instead it may be better to:

- Drive the organization to the point where there the number of schedule misses due to resource conflicts is small, but always non-zero.
- Use the schedule misses to identify queue overload situations that can be corrected.

"This has two benefits:

- Sets an overall culture where the emphasis is on setting aggressive

schedule targets and mitigating effectively when problems occur, vs. sandbagging, as a way to achieve success.

- Ties queue length assessment to data that we already collect and pay attention to (schedule slips and explicit resource conflicts), rather than more abstract measures of resource loading that are less often watched and more difficult to collect (different people work different numbers of hours per week).[23]

"Our start control meetings entail:

Concept Commit. In-depth review to ensure clear understanding of what the project is about and why it is worthwhile before any development planning starts. The concept commit presentation includes the names of the people who will work on the planning(another workload item that can clog queues).

Execution Commit. In-depth review of development plan (including marketing, manufacturing, customer support, engineering). How closely the project team operates to the plan approved here is career-affecting. One unbreakable criterion for approval is having names of the people who are assigned to every task, and clear evidence that all of them are available when needed to do the work ('to-be-hireds' are allowed only under duress and then only when there is high confidence that the necessary people will be on board and up to speed in time to support the proposed plan; progress on to-be-hireds becomes a monthly program review item until the new hires are on board).

"After that, the program lead does a 15 minute update once per month to senior management. There is also a weekly operations meeting where the program lead can (and is obliged to) raise exceptional conditions needing cross organization help, report and / or ask for help on schedule- or resource- affecting events, etc."

[23] Reinertsen has the following elaboration to Goodhue's speculation about how to measure people's loads (personal communication, October 1999): "If we think about the shape of the queueing curve, the slope rises exponentially as we raise utilization. Think of this as a transfer function relating queues and capacity utilization. If, at a heavy level of loading I can estimate capacity utilization +/-20 percent there could be massive variation in my queue time (+/-1000 percent), and therefore cycle time. On the other hand, if I can measure queue times +/-20 percent it gives me a very precise feel for my level of capacity utilization, without trying to measure it. Although some academics advise measuring capacity utilization I would suggest that a deeper understanding of product development dynamics would lead a smart developer to focus on queues."

Metrics (3b)

The benefit of process changes, tools, organizational changes and so on cannot be accurately assessed without metrics. Obviously, we need metrics, and many people start improvement efforts there. However, we must not stop with metrics — they alone will not make improvements. Thus, the model being described in this paper treats metrics as only one of several components.

Also, we must find relevant metrics that help us actually *improve* in ways that actually *help* our business.

When one starts looking at improvement of a process such as the new product development process, it is shocking how few metrics one has and how poorly defined they are. People are not able to agree on things as simple as when a project started or when it was initially scheduled to be complete. One needs appropriate and well-defined metrics. One may be able to do some retroactive construction, but at least now one needs to begin to keep them.

Reinertsen recommends creating explicit quantitative models that we can use to make trade-offs.[24] Thus, the metrics need to:

- Help us understand business trade-offs — for example, for new product development this might involve the metrics of development cost, development time, product cost, product function, and the profit to the company that results from various combinations of these parameters in the business model.
- Help us understand application trade-offs.
- Help us understand process economics, such as the effect of queues (e.g.,

[24] See chapter 2 of his book *Managing the Design Factory* for a discussion of quantitative models and tradeoff rules and pp. 201-202 for a discussion of the control function of decision rules.

work in progress) and changes of direction (e.g., engineering change orders) and their timing.

As consultant Brad Goldense noted in his presentation to the study group, such models allow predictions to be made which can be compared with actuals to try to improve the model (and its dynamics) so the predictions will better match reality going forward.

As already noted, the metrics need to address system wide effects, not just local optimizations — overall throughput of the system, not necessarily the throughput of a component of the system, except when the throughput through the component is actually correlated with increased throughput of the overall system. Thus, one must be careful about using many traditional cost accounting metrics that measure local effects, such as utilization of a machine or department, and thus when optimized can be significant source of overall system ineffectiveness and inefficiency. In general, one needs to measure cycle time both at the step level and the overall process level, and to try to find the relationships between the two.

Goldense emphasizes looking at predictive as well as reactive metrics. Predictive metrics are often found by looking at trends across many projects — trying to find the "fundamental laws" of one's system. We have already mentioned another, more limited, sense of the word predictive: for instance, looking at queue lengths as a predictor of throughputs.

The metrics also have to be monitored and used often enough to actually operate the system. For instance, Cisco monitors many new product development metrics on a weekly basis.

Process Method (3c)

The process method is the process that is going to be followed, for instance, the product development process. This needs to be made explicit. It also will vary depending on the type of business one is in: some companies (e.g., a drug company with massive investment in each step of developing a new product) may find traditional phase review processes (such as were promoted heavily as best practice in the early 1990s) as appropriate. Other companies may find other process methods more appropriate.

The first step is to make the process visible. This involves thinking about what the actual process is and documenting it, at least generally, whether it is an existing process or a proposed new process. It is very difficult if not impossible to improve something that is intangible.

The process method can be documented at many different levels. The level of documentation needs to be "in touch" with the way the people undertaking the process actually do their work. Finally, when thinking about and documenting a process, there are different perspectives one can have on it. One can look at the process in terms of material flow, the information flow, the dependencies of the process (see the later subsection on the Dependency Structure Matrix), in terms of who is making commitments to whom.

Of course, the process will not be perfect at first and may be improved over time. Michael Hammer and Rummler and Brache both view processes as business assets that must be maintained and improved. In CQM member companies, we can see numerous examples of integrating good ideas into improved new "best practices" (which other companies then draw on). Kiyoshi Uchimaru told a delegation of CQM member CEOs and other senior executives that he believed that one company couldn't exactly copy the "best practice" of another company — each company has to develop its own version of a "best practice" that fits its situation and culture.[25]

[25] The late Kiyoshi Uchimaru was president of NEC's Integrated Circuits and Micro-computer System division when it won the Deming Prize. He described their improvement methods in a book listed in Appendix D.

However, Uchimaru did make one suggestion that almost any company can adopt — the design plan review. The design plan review is like a design review, except it is a review of the plan for a project and done by other experienced and successful project managers before that part of the process is executed. At APC they have adopted the term "design preview" to describe this explicit part of the process, and to distinguish it more clearly from the more traditional after-the-fact "design review." This is most important in business processes which have a design or planning phase, such as construction or new product development.

At the most basic level, a process must be partitioned into phases. Give serious thought to having greater granularity of steps within phases that you typically use in processes. Greater granularity provides increased visibility (and thus increased understanding) of the process, which allows earlier feedback of problems or other process improving information. Greater granularity also dumps smaller chunks into queues.[26] For instance, the typical new product development phase review process has four or five phases, each of which is months long, with a phase review at the end of each. Such very ungranular phase review processes tend to increase cycle times.[27] (Incidentally, a phase review process should get rid of most proposed projects early. This can be an important component of start control.)

Some steps will vary widely from cycle to cycle of a process. For instance, Cisco has different versions of their new product development process for different kinds of customers. The metrics for these different situations need to be categorized appropriately.

[26] Review the discussion of bigger vs. smaller airplanes under the paragraph on reducing variation in Section 5 (page 16 in the print version) of this paper.

[27] See also Reinertsen's challenge on batch size in Section 7 of this paper (page 32 in the print version).

Tools (3d)

The real benefit of tools cannot be assessed unless a process and metrics are in place. We also need to run controlled experiments while collecting metrics data to understand which tools matter most, if at all. For instance, the noted expert on software development, Capers Jones,[28] showed data for software development that made clear that four "tools"— code inspection, design reviews, formal testing, and having a quality assurance function focused on how to improve the process — make significantly more difference in eliminating bugs from software than lots of other methods that are commonly touted.

[28] In a 1993 presentation to the CQM in Cambridge, Massachusetts.

There are many tools that might be used. Some tools are generally relevant to cycle time reduction (or are even more broadly applicable), such as automated data and metric collection, project scheduling tools, resource allocation aids, methods of doing cost vs. speed trade-offs, tools from System Dynamics, and the Dependency Structure Matrix (see next subsection). Some tools are relevant to specific application areas, for instance for new product development: voice of the customer methods, computer aided design tools, simulators, test configurations, quality function deployment, design for manufacturing, and design for assembly. The trick is to choose tools that will genuinely help increase effectiveness and efficiency in a given situation.

A corollary to choosing the right tools is to avoid using inappropriate but available tools that have potential to lead you astray. Two common examples of available but often counterproductive tools are:

- Traditional cost accounting, which often drives people to optimize irrelevant metrics (such a subprocess utilization) at the expense of over all system throughput — this is a substantial part of the message Goldratt conveys in his book *The Goal*.
- Some process scheduling and Gantt charting tools, which do not allow users to easily schedule necessary slack times at the correct points in a process, or easily and adequately calculate the probabilities of various combinations of events and thus total project results.

In many cases tools become more effective as a process becomes standardized and more repetitive, but we should not relax our attempts to try to apply relevant tools in more dynamic situations.

Dependency Structure Matrix

One tool the CQM study group looked into in much detail was the Dependency Structure Matrix (DSM). This is a tool codified by Professor Donald Steward of Sacramento State University and promoted by him and study group member Steve Denker. Another paper in this issue of the *CQM Journal* is an introduction to DSM, "Planning Concurrency and Managing Iteration in Projects" (http://cqmextra.cqm.org/cqmjournal.nsf/issues/vol8no2).

Organization(3e)

Organization includes reporting, roles, how teams are set up, whether they are "heavyweight" (teams that have the people and authority necessary to make wide-ranging changes, as appropriate) or not, and so forth. Unfortunately, the first step most companies take when they are dissatisfied with results is to reorganize, since it is relatively easy to do. However, this is usually a classic case of what Deming called "tampering" — changing the process around when you don't yet understand what the problem is. Consequently, reorganizing typically doesn't improve things, and may make them worse (since at minimum it is disruptive and may result in key talent being reorganized away). Rummler and Brache report that by the time they get called in to consult a company has often already reorganized a couple of times with no effect.

Organization should be derived from the process: form follows function. First one should work on the process, and then reorganize to match the process. Organization should probably be the last of the six subelements of process design that should be worked on.

Many organizational issues need to be considered. We have already talked about some of them: the organization needs to reduce paralleling of both people and projects, and the organization needs to stop trying to run at over 100% of capacity. We have already discussed the need to be careful about putting safety factors into tasks, and this means avoiding incentive systems that encourage people to put safety factors into tasks. We all know that rewarding getting a task done on time often lengthens the time estimates people give for a task and thus ultimately the time to do the task, since slack inevitably gets used up.

Next, management needs to be appropriately involved — making sure teams have appropriate staffing, removing obstacles to success and so on. Kiyoshi Uchimaru of NEC's Integrated Circuits and Microcomputer Systems division stated that the job of the manager is to prevent problems and to teach problem anticipation. At Cisco more senior managers are responsible for strategy and teaching skills. The job of the functional manager is to avoid having people be overloaded, to develop and grow people, and to recruit new people.

Management must also make sure methods are available for sharing of knowledge, both explicit and tacit. Explicit methods for carrying on various types of business conversations can help with knowledge sharing.[29] Job rotations would be another method of sharing knowledge. Nonaka and Takeuchi's book *The Knowledge-Creating Company* contains an extensive discussion of the ways knowledge gets shared in organizations.

[29] See Vol. 4, No. 4 of the *Center for Quality of Management Journal*, Winter 1995, for several papers on this topic (available on the Web at http://cqmextra.cqm.org/cqmjournal.nsf/issues/vol4no4).

Of course, almost everyone talks about so-called "heavyweight teams" — cross-functional teams of co-located people reporting to a team leader with broad responsibilities and authority. This often is a reaction to the traditional functionally organized business which is optimized for functional efficiency but often has lots of political, resource allocation, motivational and other problems that prevent projects from getting done rapidly. John Goodhue of Cisco told the study group about a hybrid method used within (at least parts) of Cisco where project managers and functional managers are the same people. With their project leader hat on, each manager needs the effective support of the functional managers; therefore, with their functional management hats on, managers make sure they provide appropriate resources (i.e., not over allocated or multi-tasked) to projects. This is similar to a method used at one point within HP Medical Products Group where each line manager in the top management team also had a functional management role: since line managers needed the effective support of functional managers, in their functional roles the managers were careful to provide appropriate collaboration to other senior managers.

Another technique used by some companies is involvement of customers or users early in projects. We already mentioned the practice of not beginning improvement projects unless a customer is involved, because customer input provides a counter balance to the typical internal jockeying that often prevents projects from getting done on time or at all. A specific example of this is Cisco's new product development process which requires the involvement of lead users as one qualification for going ahead.

Brad Goldense in his presentation to the study group supported the viewpoint stated above that the organization should be derived from the process and other design elements. Specifically he noted that there is lots of talk about various kinds of teams, but that process is more important. He continued saying that an appropriate process is necessary for teams to operate in a stable way.

Process Knowledge (3f)

Once a new or improved process (for instance for new product development) has been designed, it surely will not be perfect and may have some fairly severe glitches that seemed like a good idea at design time but are not in practice. Each time the process is used, new things are learned about what works well and what does not work well and about various improvements that might be made. The process design needs to include a mechanism for capturing this new insight and making is accessible to future users of the process.

Some learning from executing the process is folded back into the design of the process through continuous improvement. However there is other knowledge from past cycles of a process that don't affect the design of the process but are useful to future cycles of the process. These can be hints or known pitfalls, or they can be work that can be reused in a future similar cycle of the process. It is important to make this information available in an organized way to future users of the process.

This mechanism might be written product design standards (such as APC uses. It might be a data base of some sort (such as the study group heard that Cisco creates for each project to go with their design process). It might be some other mechanism to "socialize" improved practice. At APC they have created templates of the records that need to be kept for each project, so that people know the format well enough to access the process knowledge.

7. Opportunities for Advancing the State of the Art

From the CQM cycle time reduction study, it became clear that there are many specific opportunities for advancement of the state of the art in cycle time reduction.

Neil Rasmussen provided the following list of opportunities for advancing the state of the art as part of his interim study group presentation:

- Capacity planning tools
- Scheduling tools
- Standard, proven metrics and the tools to record and benchmark them
- Incentive systems (either tangible or intangible) that facilitate deployment of improvements and an improvement mentality
- Process mapping and documentation systems.

As we stated at the beginning of this paper, providing a general step-by-step process for cycle time reduction is also an open problem.

Finally, Donald Reinertsen, whose thinking was so important to the study group and this paper, offers the following two challenges[30] as CQM members continue to grapple with the theory and practice of cycle time reduction:

- On batch size — Is batch size important in engineering processes? What batch size is used today? What is the key to reducing it? For example, consider the process batch size used in a phase/gate process, in a precedence diagram, in a test process, in a drawing release process, and so on.
- On precedence — Traditional concepts of precedence (PERT, CPM, DSM) dictate that you should not start an activity before all tasks upon which it depends are completed. What economic parameter is being optimized by such logic? What does such logic do to overall economics when time on the critical path is expensive?

[30] Personal communication, October 1999.

Acknowledgments

We appreciate the contributions of all of the participants in the CQM study group on cycle time reduction, the experts that gave presentations to the study group, and the authors of all the papers and books studied. We particular appreciate the insights and sharing of experiences of author Donald Reinertsen and presenter John Goodhue who allowed us to quote them extensively in this paper. Kevin Young, Heather Wambach, and Eric Bergemann helped with the preparation of the manuscript.

Appendix A — Study Group Participants

- Bob Abramson, Open Market
- Mark Braun, Boston University
- Anthony Carroccio, Health Alliance
- Stephen Denker, GTE Internetworking
- Stephen Graves, Massachusetts Institute of Technology
- Tom Hennings, The Plastic Moldings Corporation
- Robert Herrick, Health Alliance
- John Hillerich III, Hillerich & Bradsby
- Verne Johnson, Portman Equipment
- Scott Jones, SerVend
- Thomas Lee, Center for Quality of Management
- Brad Nelson, Teradyne

- Richard Paynting, CTI-Cryogenics
- Neil Rasmussen, American Power Conversion
- Richard Rodriguez, Zaring National
- Carl Roos, Franciscan Health System
- Steve Rosenthal, Boston University
- Ed Starr, GTE Internetworking
- David Walden, Center for Quality of Management
- Toby Woll, Center for Quality of Management — moderator
- Terri Trespicio, Center for Quality of Management — administrative and logistics support

Appendix B — Invited Presentations by Experts and Users of Various Methods

- Dependency Structure Matrix Method, Steve Denker (study group) and Joe Killough (Bose)
- Cycle Time Reduction at Zaring Homes (a division of Zaring National), Richard Rodriguez and Mahesh Bhupalam (Zaring)
- Value Analysis: Key to Cost Reduction without Sacrificing Quality, Bob Libby (Bose)
- Clark and Wheelwright's Revolutionizing Product Development at Teradyne, Brad Nelson (Teradyne)
- Bristol-Myers Squibb Performance Improvement Process, John P. Reinhardt (Bristol-Myers Squibb)
- Measuring Product Development, Brad Goldense (consultant)
- Process Improvement and Management, Alan Ramias (Rummler-Brache Group)
- Systems Thinking and System Dynamics, Jamshid Gharajedaghi (Interact)
- Cycle Time Reduction at High Voltage Engineering, Paul Snyder (High Voltage Engineering)
- Using the Dependency Map for Information-Driven Project Management at Bose, Richard Paynting (CTI, ex-Bose)
- Product Development at Cisco, John Goodhue (Cisco)

Appendix C — Presentations and Reports of Research Done by Study Group Members

- *Developing Products in Half the Time*, by Smith and Reinertsen (reported by study group member Rasmussen)
- *Revolutionizing Product Development*, by Clark and Wheelwright (Rasmussen)
- *Fast Cycle Time*, by Meyer (Rasmussen)
- On Queuing Theory (Roos)
- *TQM for Technical Groups*, by Kiyoshi Uchimaru et al. (Rasmussen)
- *Improving Performance: Managing the White Space*, by Rummler and Brache (Braun)
- Michael Hammer's New Product Development Seminar (Rasmussen)
- Design Structure Matrix Method (several different reports) (Denker)
- *Revolutionizing Product Development*, by Clark and Wheelwright (Lee)
- *The Machine That Changed the World*, by Womack, Jones and Roos (Hillerich)

- *The Knowledge-Creating Company*, by Nonaka and Takeuchi (Johnson)
- *Fast Cycle Time*, by Meyer (Roos)
- Cycle Time Reduction Study Group Progress Update, presented at CQM Roundtable meeting (Rasmussen)
- Cycle Time Reduction at SerVend (Jones)

The study group members circulated a number of additional papers and book chapters to each other. A complete list of these is available on the CQM members-only Web site (http://cqmextra.cqm.org).

Appendix D — A Basic Literature of Cycle Time Reduction

The Machine that Changed the World: The Story of Lean Production, James Womack, Daniel Roos and Daniel Jones, paperback reprint edition, Harper Perennial Library, 1991. Shows how focusing on cycle time can change an entire industry.

Managing the Design Factory: A Product Developers Toolkit, Donald Reinertsen, Free Press, 1998. Includes an introduction to the queuing, scheduling, and multitasking problems that almost all organizations suffer from. Members of the study group were quite excited about this book. (See also the book *Developing Product in Half the Time*, Preston G. Smith and Donald G. Reinertsen, 2nd edition, John Wiley & Sons, 1997, which provides insight into how to create realistic models and take advantage of the trade-offs of various kinds between cycle times, cost and so on.)

Improving Performance: How to Manage the White Space on the Organization Chart, Geary Rummler and Alan Brache, second edition, Jossey-Bass, 1995. Describes how to think in terms of key business processes and levels of process, and how to organize in terms of key business processes.

The Goal, by Eliyahu Goldratt, 2nd revised edition, North River Press, 1994. Recommends thinking in terms of increasing throughput, lowering operating costs, and eliminating most inventory, and describes how many traditional cost measures lead to inefficient processes. (See also Goldratt's book *Critical Chain*, North River Press, 1997, which describes time in project management as analogous to inventory in manufacturing with all the same problems and requirements for careful regulation.)

TQM for Technical Groups: Total Quality Principles for Product Development, Kiyoshi Uchimaru, Susumu Okamote and Bunteru Kurahara, Productivity Press, 1993. Emphasizes building the right unique system for you and the benefits of making the process visible.

If you want to read only one book on cycle time reduction, many people recommend *Fast Cycle Time: How to Align Purpose, Strategy, and Structure for Speed*, Christopher Meyer, Free Press, 1993. It is a useful book. Nonetheless, we recommend starting with the list of five books above.

The Center for Quality of Management Journal is a forum for disseminating the experience of organizations learning to implement modern management practices. It seeks to capture experiences and ideas that may be useful to others working to create customer-driven, continuously improving organizations.

The CQM Journal is refereed. However, it is not an academic publication. Experiences and ideas will be published if they seem likely to be useful to others seeking to improve their organizations.

Send to:

The Center for Quality of Management Journal
Editorial Department
One Alewife Center, Suite 450
Cambridge, MA 02140
Tel. 617-873-8950 Fax 617-873-8980
E-mail: publications@cqm.org

If you have thoughts for a paper and you would like to discuss it with us, please write, call or submit an outline. We welcome your ideas.

Final Manuscript Requirements:

Entire manuscript should be double-spaced, including footnotes, references, etc. Text should include all the elements listed below. Generally, The CQM Journal follows the editorial principles of The Chicago Manual of Style. We strongly prefer submissions in eletronic format for all text and as many of the figures as possible. IBM-based software (particularly Microsoft Word for Windows) is preferable to Macintosh-based software if you have a choice, but is by no means a requirement.

Please include:

1. Title page, stating the type of article (e.g., 7-Step case study, research paper, short communication, letter to the editor, etc.), main title, subtitle, and authors' full name(s), affiliation(s), and the address/phone/fax of the submitting author;

2. All needed figures, tables, and photographs (see below);

3. Footnotes (if appropriate), numbered consecutively from the beginning to the end of the article;

4. Reference list, if appropriate.

Figures, Tables and Photographs:

If you can, insert each figure or table into the text where you would like it to fall. Figures should be composed to conform to one of two widths: 3 1/8 or 6 1/2 inches. The maximum height for any figure is 9 3/8 inches. Text within figures should not be smaller than 5 points and lines not less than 1/4 point at the figure's final size. Figures should labeled with the figure number underneath and title on top. Be sure that the text mentions each figure or table.

Please retain separate PICT or TIFF files of figures generated in drawing programs and a file with the text only for final submission.