

**Multi-Project Scheduling and Control: A Process-Based Comparative Study of the  
Critical Chain Methodology and Some Alternatives.**

**Izack Cohen\*#, Avishai Mandelbaum\*, Avraham Shtub\***

**16/10/03**

\*Industrial Engineering and Management  
Technion Israel Institute of Technology  
Haifa, Israel 32000

#Israeli Air-Force

Corresponding author:  
Professor Avraham Shtub  
Industrial Engineering and Management  
Technion Israel Institute of Technology  
Haifa, Israel 32000  
Voice 972-4-8294454  
fax 972-4-8293172  
email: [shtub@ie.technion.ac.il](mailto:shtub@ie.technion.ac.il)

## **Multi-Project Scheduling and Control: A Process-Based Comparative Study of the Critical Chain Methodology and Some Alternatives.**

### **Abstract**

Critical-Chain (CC) is a popular project management technique in many multi-project organizations. It applies the Theory of Constraints (TOC) to offer a practical and easy method for planning, scheduling and control of multi-project systems. While some prior studies examined CC performance for single project management, little attention has been given to its performance in a multi-project environment. In this paper we examine the control mechanisms of CC and some alternatives. We demonstrate that, when CC is not enough to prevent projects' lateness, such alternatives may give rise to similar and sometimes better, possibly much better performance.

**Keywords:** multi-project management; critical chain; project scheduling and control

## **Multi-Project Scheduling and Control: A Process-Based Comparative Study of the Critical Chain Methodology and Some Alternatives.**

### **Introduction**

Critical-Chain (CC) methodology for project management (Goldratt, 1997) applies the Theory of Constraints (TOC) to multi-project scheduling and control. Specific software packages, based on CC methodology, have been developed (Speed to Market's Concerto 2002 and ProChain Solutions Inc. 1999). In parallel, a growing number of articles relating to CC have been published – some are criticizing the approach (Herroelen and Leus, 2001; Herroelen et al., 2002; Shou and Yeo, 2000) and others are praising it (Steyn, 2000; Leach, 1999). None of these papers has thoroughly examined the performance of CC methodology in a multi-project environment. Yet, a growing number of multi-project organizations have chosen the CC methodology for planning, scheduling and control of their projects, which has motivated our efforts to understand it better.

It is thus our intention to obtain new insights on the CC methodology in a multi-project environment, while comparing its performance with alternative methodologies. To this end, we consider an environment of multiple concurrent projects in which projects compete for the same set of *scarce resources*. The environment is *random* (stochastic) in that uncertainty plays in it a significant role. Projects are *unique* in that their operational requirements and activity durations differ. Yet, projects are also “non-unique” in that they share common characteristics that enable their classification into classes; for example, within a class, precedence relations between projects' activities can be identical and, for

each activity, the historical realizations of activity times fit a common distribution function. According to the first authors' experience, such an environment prevails in organizations that process maintenance or retrofit projects, which are common in the aircraft industries (for a description of maintenance projects in the aircraft industries, see also Gemmill and Edwards, 1999). Here, the realization of each project in the *project portfolio* is unique (e.g., due to its unexpected delays, different technical findings or even differences in aircraft structures). However, despite the differences between the projects, one can model such an organization as one that processes several classes of projects with, for example, each different type of aircraft giving rise to a different project class. This approach of classifying projects in a multi-project organization into different classes has been found useful in past research (e.g., Adler et al., 1995, for product development projects in the chemical industry, Leung, 2002, for software maintenance projects, and Griffin, 2002, who suggested a general classification of all projects in an organization to four project classes).

A natural modeling framework for non-unique multi-projects was described by Adler et al., 1995, in terms of stochastic *processing networks*. The building blocks of such a network are interdependent resources that process project activities according to some pre-specified discipline. At any given moment, each activity is either receiving service from a resource, queuing up for access to a resource or waiting to join a prerequisite activity that is being processed or delayed elsewhere. The network model is stochastic, or random; for example, activity durations are modeled by random variables. Randomness here captures unpredictable variability that is prevalent and significant in most project environments.

The stochastic network paradigm is also natural for the analysis of *buffer management*, as defined by the CC methodology. A buffer stores "time-units", specifically time by which the project activities corresponding to this buffer could be delayed without causing a delay to the planned project due-date. A buffer, thus, stores slack time that is added during planning. Buffer management then amounts to the dynamic management of resources according to buffer contents, or equivalently buffer consumption levels. For example, amongst several competing activities, top priority in resource allocation is given to the activity whose buffer consumption is the highest, namely its slack time is the least.

Through the comparison of CC to other alternative methodologies, we in fact address two aspects of broad significance to project management. The first is the *trade-off* between resource *utilization* and project *throughput*: project throughput times get longer as resources' utilization become higher. We quantitatively demonstrate this trade-off for the different management methodologies. The second aspect is implementation *costs*. Here we demonstrate that some simple management methodologies, requiring low implementation costs, can achieve very good performance compared to the CC methodology. We note that the implementation of CC methodology in an organization usually requires some organizational changes and considerable implementation costs (mainly training costs for both management and workers, and purchase costs for special software packages). Buffer management gives rise to additional ongoing costs.

The rest of the paper is organized as follows: In the next section, we discuss the fundamentals of CC methodology. Then we introduce the process management approach for modeling dynamic multi-project environments. Next, we elaborate on our experimental

design, accompanied by a section with notation and formulas. In the following two sections, we present our main results: a comparative simulation analysis of the CC methodology for the management of non-unique multi-projects. We then conclude with a summary and some suggestions for further research.

### **Fundamentals of CC methodology**

CC methodology (Goldratt, 1997) aims at developing a sound schedule, using buffer management, in order to avoid project overruns. The methodology is not well defined in the sense that it leaves open precise definitions for some project entities and scenarios. Rather, it gives a heuristic framework and guidelines for project managers on how to plan, schedule and control their projects, and it is up to the user of the method to complete the details. (For further discussion of the merits and pitfalls in CC methodology, readers are referred to Herroelen and Leus, 2001.) We now review the steps of CC methodology as they apply to the models discussed in the sequel. We start with a single-project environment, and then generalize to a multi-project environment.

*The CC steps for **single** project planning, scheduling and control are as follows:*

Step S1: Reduce activity durations by eliminating safety margins.

Estimates of activity durations include safety times. Indeed, based on their experience, managers tend to quote late due dates so that they can meet them with a high degree of certainty. The result: inflated activity durations which become self-fulfilling, or even overrun due to the combined effects of stochastic variability and Parkinson's Law (Work

expands to fill the time given for execution, Parkinson, 1957). CC methodology, hence, and right at the outset, reduces predicted activity times to their median (which ensures 50% probability of on-time completion, Goldratt, 1997) or to their average duration (Product Development Institute, 1999; Herroelen & Leus, 2001). .

Step S2: Identify the *critical chain*.

A critical chain is a sequence of activities that determines the project duration, taking into consideration both precedence dependencies and resource constraints. Such a critical chain arises from a *project plan* that assumes deterministic estimations for the reduced activity durations and resource requirements, as in Step 1. (When a critical chain is non-unique or difficult to identify, the guide is to pick one up arbitrarily.) The project plan is then further revised according to the "late starts" of the project activities.

Step S3: Create a *project buffer*.

Some of the safety times that were eliminated, in Step 1, from activity durations are shifted to the end of the critical chain and "stored" in a time buffer. This buffer, called project-buffer, is used dynamically to protect the project due-date against variations in critical chain activities. A standard guide is to set the project-buffer capacity to 50% of the total duration of the critical chain (Leach, 1999).

Step S4: Create *feeding buffers*.

Delays in *non-critical activity chains* (activity chains merging into the critical chain) could cause undesirable delays of the critical chain. Consequently, feeding buffers are added at the end of each non-critical activity chain ("pushing" the latter, which was scheduled by "late start", back in time). The feeding buffers thus protect the critical chain from variations

of non-critical chains and allow critical chain activities to start early, when possible.

According to Leach (1999), a feeding-buffer capacity is set to 50% of the duration of its non-critical activity chain.

#### Step S5: Control

Buffer monitoring provides a quick grasp of project status, which, in turn, enables adaptive control. Specifically, buffer consumption that reaches a predefined threshold (e.g., two-thirds of the buffer size or, equivalently, one-third of the slack time remains unused; Leach, 1999) triggers an early warning towards taking some preventive managerial action. More details will be provided later.

Multi-projects are accommodated by combining single-project scheduling with TOC (Goldratt, 1984) and CC principles, notably the emphasis on reducing multi-tasking (Herroelen and Leus, 2001; Leach, 1999). To this end, project start-times are staggered, which turns the multi-project system into a "pull" system with newly determined release/start times. Here are the relevant details.

#### *Scheduling and control of a **multi-project** system:*

Step M1: Treat each project as a single project.

Individually schedule each of the multi-projects, using the four steps for scheduling a single project, as described in Steps S1-S4.

Step M2: Stagger projects according to the *bottleneck* resource.



First identify the bottleneck, namely the most constraining resource (often by using merely managerial experience). Then release projects sequentially, by staggering them, so that the bottleneck works continuously, that is without idle time.

Step M3: Create a *capacity buffer*.

A time buffer, called the capacity buffer, is associated with the bottleneck, and its role is to ensure bottleneck availability. The capacity buffer decouples between bottleneck activities that belong to successive projects, thus determining projects' start times. Since there is no standard way in the literature to set the size of this capacity buffer, we set its base-case size at 50% of the duration of the bottleneck activity. We then analyze the effect of alternative sizes by varying the values through 8.3%, 16.7%, 83.3% and 116.7%.

Step M4: *Control*

As with single projects, scheduling control of multi-projects is buffer-based: when allocating an idle resource, top priority is given to critical chain activities over non-critical chain activities; secondary priority is given to activities of projects with the highest level of project buffer utilization, or equivalently the least slack time. Least priority is given to activities of projects with the highest feeding buffer consumption.

### **From project to process management**

Following Adler et al., 1995, we model a multi-project organization as a stochastic processing network. Adler et al., 1995 validated the model based on an actual R&D organization, showing that the model simulated quite accurately its performance.

In the model of a stochastic processing network, each network node represents a group of (one or more) statistically-identical resources, who perform the same type of activities and who are able to do so in parallel. When several activities of a project can start being processed at the same time, we refer to the phenomenon as a "*fork*"; when an activity cannot begin until its predecessor activities have been completed, we call it a "*join*".

(Consequently, such models are often referred to as Fork-Join queues, for example see Nelson and Tantawi, 1988.) The time required to complete an activity is called its *processing time* (duration) and the intervals between successive project releases are "*inter-arrival times*". The reciprocal of the mean inter-arrival time is the projects *input rate* (expressed in number of projects per unit of time).

We use network diagrams, as in Figure 1, to illustrate activities' precedence requirements. The network is stochastic since inter-arrival times, processing times and precedence requirements are subject to random (stochastic) variability. Projects are of the same *type* if they are characterized by the same set of probability distributions (precedence, inter-arrival times and processing times). This representation can thus model a multi-project system with different project types. When a project activity "arrives" to a node/resource, it either starts its processing immediately or it joins a queue and waits there till its processing starts. Such queues are called *resource-queues* – they are managed according to priority rules and

are subject to resource availability. Another type of waiting takes place in *synchronization queues*, where activities are delayed due to precedence constraints. We shall now make these abstract notions concrete via a simple example of a processing network.

### **Model construction**

Consider the multi-project system depicted in Figure 1. The system has four resource types, numbered 1- 4, which process projects of a single type. Each project consists of four activities, denoted A-D: Resources 1 are *dedicated* to processing Activities A, 2 dedicated to B, etc. (The more general model could have a resource type processing several activities.) The Start and Finish activities are milestones - they have neither a duration nor a resource requirement. Figure 1 could be viewed as representing a simple multi-project organization, for example an aircraft maintenance company or a chemical product development process. The system characteristics are given in Table 1. From the table we read that release times between successive projects have an exponential distribution with mean 3.25 units of time; that 3 resources of type 1 are dedicated to processing activities of type A, and that the processing durations of such type A activities have an exponential distribution with an average of 6 units of time. (The dependence of performance on the distribution of the processing-duration will be discussed in our concluding section.) The bottleneck resource, namely the resource that determines the system's processing capacity (Step M2), is Resource 4. This choice finds ample support in our subsequent analysis (see Table 2). But it can be roughly justified, by observing that a mere single Resource 4 is

dedicated to activity D, with an anticipated utilization level of about  $3/3.25 = 92\%$  in steady state, which is by far the highest among all the resources.

Figure 2 illustrates the buffered single- and multi-project systems, according to the CC scheduling methodology described in Steps S1-S4 and M1-M3 respectively; as already noted in Step M3, we set the size of the capacity-buffer to 50% of the duration of the bottleneck activity D.

\*\*\* Figure 1 preferred location (see figure's appendix) \*\*\*

\*\*\* Table 1 preferred location (see table's appendix) \*\*\*

\*\*\* Figure 2 preferred location (see figure's appendix) \*\*\*

### **Experimental design**

Our experimental tool is a simulation model, written in Visual Basic on a personal computer. Each simulation run started with a warm-up period that leads to steady state. This transient phase was discarded from the analysis. The system was then simulated in steady-state for a predefined time interval that was chosen according to Welch's moving average procedure, as described in Law and Kelton, 1991. Each simulation was replicated for 50 times.

The experiment aims at comparing system performance under CC with performance under alternative control methodologies. Performance measures are mean project duration, its standard deviation, and throughput rate, i.e. the number of completed projects per unit of time.

We analyze two types of controls: *open* control, under which all candidate projects are actually initiated, and *closed* control where projects must adhere to some predefined criteria in order to be started. Our open controls are termed No-Control, CC and MinSLK; the closed group consists of ConPIP and QSC. Here are their details:

*Open Controls:*

1. **No Control** - A push system with FCFS (first come first served) queues priority rules. (A thorough analysis of this specific model is carried out in Barron and Mandelbaum, 2003, and Baron, 1996.)
2. **CC** - New projects are initiated according the CC methodology, as was described in Step M2. Guided by Step M3, buffer management determines the priority of the next activity to be processed by a resource. When viewing CC methodology as a process management model, we make the following observations. Buffer consumption determines the priority of its corresponding activity in a resource queue: the higher the consumption (less slack) the higher the priority. While an activity is delayed in queue (either synchronization or resource), its buffer consumption increases until it reaches a predefined threshold. This threshold is here taken as two thirds of the associated buffer size, for both feeding and

project buffer, and when it is reached, a resource is allocated to the activity by preempting activities with lower buffer consumption.

3. **Highest priority in queue to a Minimum Slack activity (MinSLK)** - When an activity is completed, the prevalent critical-path is re-evaluated and slack times are updated for the rest of the projects' activities. (Here *slack-time* is defined as the difference between the late start time and early start time.) MinSLK (also called MinSLK[DD] by Bock and Patterson, 1990) employs the following priority rule for allocating resources to activities: the lower the slack time the higher the priority. Thus, as a project is delayed, the priorities of its activities increase. (Note that slack times can turn negative, specifically when the start time of an activity is later than the late start time that is needed in order to complete the project on its due-date.)

*Closed / Semi-Closed Controls:*

4. **Constant Number of Projects In Process (ConPIP)** - new projects are started based on a predetermined number of projects in process, call it NPIP (Adler et al., 1995 and Anavi-Isakow & Golany, 2002). Specifically, an arriving project starts its processing immediately if the number of projects concurrently in process within the system is below NPIP; otherwise, it is placed in an external queue and waits till it can be processed. We define the *throughput time* of a project as the time span from its start-time (S), when it leaves the external queue, until its finishing time (F), when it leaves the system. For varying values of project input rates, we determined NPIP values as the maximal number of projects allowed concurrently in the system so that the mean waiting time in the external queue does not

exceed half of the average throughput time. (The latter condition is plausible for the aircraft maintenance industry, according to the experience of one of the authors.)

**5. Queue Size Control (QSC)** - A predetermined maximal number of activities is allowed, at any given time, within the resource queue of the bottleneck. An arriving project is then allowed into the system to be processed if the length of the bottleneck's resource queue is below this maximal number; otherwise, the arriving project is discarded, never to return.

### Notation and formulae

Consider  $n$  replications, each of length  $m$ ;  $Y_{ij}$  is the duration of the  $i^{\text{th}}$  project from the  $j^{\text{th}}$  replication ( $i = 1, 2, \dots, m; j = 1, 2, \dots, n$ ). The average project duration for replication  $j$  is taken to be (Law & Kelton, 1991):

$$X_j = \frac{\sum_{i=l+1}^m Y_{ij}}{m-l},$$

where  $l$  is the length of the transient period, counted in number of projects.

Assuming that replications are independent, the  $X_j$ 's are independent random variables

with  $E(X_j) \approx \mu$  (the true steady-state mean value). The overall mean duration  $\bar{X}$ , where

$$\bar{X} = \sum_{j=1}^n \frac{X_j}{n}$$

is approximately an unbiased point estimator for  $\mu$ . An alpha-level confidence interval is

given by:

$$\bar{X} \pm t_{n-1, 1-\alpha/2} \frac{S}{\sqrt{n}},$$

where  $S^2$ , the estimator for the variance equals

$$S^2 = \frac{\sum_{j=1}^n (X_j - \bar{X})^2}{n-1}.$$

The mean standard deviation SD of a replication is

$$SD = \frac{\sum_{j=1}^n \sqrt{\frac{\sum_{i=l+1}^m (Y_{ij} - X_j)^2}{m-l-1}}}{n}.$$

The arrival rate of projects,  $\lambda$ , is a given parameter, while the effective throughput rate,  $\lambda_{\text{eff}}$ , is a measured output of the model/simulation. For each resource type, we estimate its mean *traffic intensity*  $\rho_k$  by

$$\rho_k = \frac{\sum_{j=1}^n \frac{\sum_{i=l+1}^m T_{jik}}{U_j}}{n * N_k}.$$

Here  $T_{jik}$  is the processing-time of project  $i$  by resource type  $k$  in replication  $j$ ;  $U_j$  is the simulation duration in steady-state for replication  $j$ ; and  $N_k$  is the number of resource units for resource type  $k$ .

### Simulation results

Table 2 summarizes the performance measures of our simulation experiments, in steady state. Applying CC methodology, specifically Step S2, to the multi-project system described by Figure 1 and Table 1, yields a project plan with throughput time of 17.5 time



units and a throughput rate of 0.22 projects per unit of time. Our simulation analysis reveals a mean project throughput time of 15.50, which is below the planned 17.5 value. Nevertheless, 34% of the projects are expected to have a throughput time that exceeds the plan:  $P(T > 17.5) = 0.340$ , where  $T$  denotes a random variable that models project duration in steady state, and its distribution is here determined by the histogram in Figure 3a.

A reduction of the capacity buffer (from the 50% that the CC methodology often guides as default) results in an increase of both project throughput time and its standard deviation. Reduction of buffer capacity to 16.67% resulted in an increase of mean throughput time to 21.87 and standard deviation of 11.15 (51.0% of the mean). Further reduction to 8.33% buffer size resulted in mean throughput time of 32.65 and standard deviation of 18.92 (57.9% of the mean).

We compared the performance of the different control methodologies to CC. For each throughput rate, we performed a modified t-test confidence interval test, at the 0.95 confidence level (using Welch's approach for comparing performance measures without assuming equal variances; see Law and Kelton, 1991, pp. 588-594).

When the system operated with No Control, the mean and standard deviation were significantly higher than in CC, for all throughput rates.

Performance of the MinSLK priority rule was not significantly different from CC at throughput rates of 0.29, 0.22 and 0.18 and slightly better for throughput rate 0.31. The throughput time distribution for throughput rate 0.22 (see Figure 3b) is similar to that of CC, with  $P(T > 17.5) = 0.341$ .

Queue Size Control gave rise to lower values of both mean throughput time and standard deviation. The improvement is most noticeable at high utilization levels and hence high throughput rates (0.31, 0.29 and 0.22). However, there is some decrease in the effective throughput rate, as described below in Table 2. For example, in arrival rate of 0.31 project per time unit, reducing the mean traffic intensity from about 90% (in CC) to 85% (in QSC(6)) resulted in a 38% decrease in mean throughput time, while the effective throughput rate reduction was 6.4%. Figure 3c presents the throughput time distribution for an effective throughput rate of 0.21 (max queue size of 3 projects). In this case, the throughput time for 30% of the projects is over 17.5 time units:  $P(T > 17.5) = 0.300$ . In Figure 4a-4d we display the throughput-time distribution for different controls, under a relatively heavy load (high throughput rate of 0.31). CC and MinSLK exhibit a very similar distribution, with over 70% of the projects expected to exceed throughput time of 17.5 time units, and the medians equal 32.44 and 30.40 respectively. The performance of the uncontrolled system is no less than catastrophic: for about 90% of the projects; the throughput time is expected to be more than 17.5 and, indeed, the median increases to 51.42. In contrast, the closed control QSC(3) exhibits fairly good performance with only 40% of the projects expected to be late for their planned due date, and the median is 17.33. Note that this is at the cost of an effective throughput rate of 0.26, which amounts to giving up about 16% of the projects.

ConPIP control has not shown a significant difference in mean throughput time relative to CC. The standard deviation and confidence interval for the mean throughput time were lower for high throughput rates (0.31,0.29).

The experience of projects, as they progress through the system, is summarized in Table 3. We are using, what we call, *time-profiles*: these are the fractions of time that projects spend either waiting for a resource, waiting for activity synchronization, or actually being worked on. Specifically, Table 3 displays time-fractions for the following performance measures: *Waiting* time in resource queues; *Synchronization* time, where activities wait until their precedence constraints are fulfilled; and *Processing* time of activities by the resources. From Table 3, we learn that a significant fraction of projects' throughput time is spent waiting, either in synchronization or resource queues. For example, the time profile for CC with throughput rate 0.22 is as follows: processing time 52%, waiting for resources 13%, and waiting in synchronization queues 35%. As the throughput rate gets higher, the difference between the time profiles in an open system and in a closed system gets larger. For example, CC with throughput rate 0.31 exhibits a time profile consisting of: processing time 26%, waiting for resources 36%, and waiting in synchronization queues 38%. Under QSC(3), with the same throughput rate, this profile changes to processing time 48%, waiting for resources 18%, and waiting in synchronization queues 34%.

\*\*\* Table 2 preferred location (see table's appendix) \*\*\*

\*\*\* Figure 3 preferred location (see figure's appendix) \*\*\*

\*\*\* Figure 4 preferred location (see figure's appendix) \*\*\*

\*\*\* Table 3 preferred location (see table's appendix) \*\*\*

## Discussion

The most important trade-off that an organization's management should consider is that between resource utilization and project throughput time: typically, the higher the former the longer the latter, and vice versa. The first-order question is thus whether to work at high traffic intensity levels and have long throughput times or, alternatively, lower traffic intensity to gain lower throughput times and lower standard deviation, hence also more predictability.

Research on multi-project planning based on CC (Leach, 1999) claims that resource competitions are resolved via buffer management, which protects against schedule variations. The claim is based on a small-scale example of a system that processes few projects; hence it is in a transient phase of operation. We, on the other hand, are analyzing *steady-state*: buffers and queues have filled up to their steady-state levels and, consequently, they are prone to being incapable of absorbing all stochastic variations. Figure 5 demonstrates variation in throughput time during the transient period until the system reaches its steady state. It is clear that analyzing only the transient period would

bias the estimate of throughput time to be far over-optimistic. Indeed, the buffers are "unchallenged" during this transient period. In contrast, some steady state scenarios are such that the buffers fail to protect the planned due date for a significant fraction of the projects.

\*\*\* Figure 5 preferred location (see figure's appendix) \*\*\*

A comparison of alternative control methodologies to the CC methodology shows that, for a given throughput rate, we can get the same or better performance using the priority rule MinSLK. This is done through dynamic control of the project, determination of its current critical path and prioritizing the activities with minimum slack. This MinSLK priority rule and buffer management are almost identical - both give priority to the critical activity (critical in the sense that it is the latest activity, or the activity with least slack). We believe that MinSLK outperforms CC in higher loads because of its adaptive determination of the critical path - it helps determine the current "most critical" activities to allocate resources to. Buffer management, in contrast, always gives priority to activities belonging to a *pre-determined* critical chain which, as throughput rates increase, is likely to have changed.

When turning the system into a closed system, as done in Queue Size Control, one can achieve a dramatic reduction of the mean throughput time - in high throughput rates -by reducing the mean traffic intensity of the bottleneck resource. For example, in an arrival rate of 0.31 projects per time unit, reducing the mean traffic intensity from about 90% (in

CC) to 85% (in QSC(6)) resulted in a 38% decrease in mean throughput time, while the effective throughput rate reduction was 6.4%. This conclusion, which is supported also by *Queueing Theory*, suggests that a *modest reduction* in the traffic intensity of a highly loaded bottleneck is likely to result in a *meaningful reduction* of throughput time: its mean and in fact the percentiles – see Figure 6. This reduction can be implemented in many different ways: for example, increasing resource allocation of the bottleneck or turning the system into a closed system by not bidding on a small fraction of the potential projects.

\*\*\* Figure 6 preferred location (see figure's appendix) \*\*\*

As we stated in the introduction, our model applies to maintenance lines in the aircraft industry (both civilian and military). Here, ConPIP control turns out natural since such organizations typically have *some flexibility* in altering the preplanned project start time. This is achieved without incurring costs to the customer, who continues to utilize the aircraft. It is especially true, and can in fact become a necessity, when the organization is doing fleet maintenance for a customer. In that situation, usually, the customer cannot release an aircraft to maintenance until another aircraft returns to service.

ConPIP based control gives rise to a performance that is similar to the other open control policies. For a high throughput rate, ConPIP has had a stabilizing affect that resulted in a lower standard deviation and tighter confidence intervals of mean throughput time.

Applying ConPIP control to an organization of the kind we are discussing involves a very small effort compared to CC or MinSLK. The key is in identifying the desired NPIP,

namely the limit to the number of projects that are allowed to reside concurrently within the system: large NPIP yields high throughput rates and possibly excessive throughput times; reducing NPIP will improve the latter but at a cost to the former.

It is worth noting that Anavi-Isakow and Golany, 2002, improve ConPIP performance further by applying more sophisticated priority rules for queue management, rather than the standard FCFS that we have used.

## **Conclusions**

CC methodology is a popular project management technique in many multi-project organizations. It offers an intuitive method for planning, scheduling and control of multi-project systems. CC acknowledges correctly the interaction between activities' precedence relations and resource constraints. Time buffers (Feeding, Project and Capacity buffers) are introduced as a systematic method for dealing with stochastic (unpredictable) variability.

The methodology offers some loose guidelines for choosing the buffer sizes. Buffer management is applied to control projects' progress, spot schedule deviations and act correctively when a buffer is consumed beyond a certain predefined threshold.

Our study examines control mechanisms in a multi-project environment, which is typical to organizations in the aircraft industries. We demonstrate that buffer management may not be enough to meet the planned schedule. Some control methodologies such as QSC, ConPIP and MinSLK can give similar and sometimes better performance. In particular, turning the system into a closed system (for example, QSC) enables one to work in higher throughput rates than those recommended by CC, while still maintaining desired delays. (Here, the

comparison is against CC with capacity buffer size set at 50% of the duration of the bottleneck activity - our base-case size)

We also demonstrate that ConPIP (Adler et al., 1995, Anavi-Isakow and Golany, 2002), which is a simple control policy that requires a minimal organizational investment, provides an effective alternative when there is flexibility in project start times. As already indicated, such is the case with maintenance or retrofit projects, which one of us has had experience with in the aircraft industries

We conclude with four observations that provide insight into the performance of multi-project systems:

- 1) Mean project throughput time is monotone increasing in throughput rate, and the better the control the milder the increase. (Refer to Figure 6: a control is superior to another one if it gives rise to a shorter mean throughput time and lower stochastic variability for the same throughput rate; for example, QSC (6) controls the system better than CC.)
- 2) Some empirical evidence has suggested that the Exponential distribution provides a reasonable fit to project duration times. But in case it does not, our framework can easily accommodate any other distribution, including actual empirical distributions. Moreover, our insights and conclusions remain intact.
- 3) An accurate inference of statistical characteristics is highly advisable. Indeed, changing the distribution of processing times or project arrivals could change, sometimes dramatically, system performance. Roughly speaking, performance deteriorates as stochastic variability increases in either processing times or arrivals (see Hopp and Spearman pp. 282-310). Consequently and practically speaking, standardization in either



activity processing or project starts would reduce, under equal throughput rates, the throughput times; or alternatively, a higher throughput rate will be achievable at equal throughput times.

4) As an overall observation, we feel safe to conclude that most reasonable controls would improve the performance of an uncontrolled system, typically significantly in heavy traffic. But, to achieve further improvement would require more, for example additional resources and/or less projects admitted.

Further research is recommended for developing more robust scheduling and control mechanisms for multi-project stochastic environments. This research could examine different multi-project environments by starting with "Mini-Systems," as done here, and then implementing the findings in realistic environments. The hope is also that such simulation analysis will provide sufficient insight and stimulus for further theoretical research on planning, scheduling and control.

## References

Adler, P.S., Mandelbaum, A., Nguyen, V., & Schwerer, E. (1995). From project to process management: an empirically-based framework for analyzing product development time. *Management Science*, 41 (3), 458-484.

Adler, P.S., Mandelbaum, A., Nguyen, V., & Schwerer, E. (1996). Getting the most out of your product development process. *Harvard Business Review*, March-April, 134-152.

Anavi-Isakow, S., & Golany, B. (2002). Managing multi-project environments through constant work-in-process. *International Journal of Project Management*, 21 (1), 9-18.

Barron, Y., & Mandelbaum, A. (2003). Performance Analysis of Dynamic Stochastic PERT/CPM Networks, Technical Report, Industrial and Engineering and Management, Technion, Israel. (<http://iew3.technion.ac.il/serveng2003/Lectures/DSPERT.pdf>).

Bock, D.B., & Patterson, J.H. (1990). A comparison of due date setting, resource assignment, and job preemption heuristics for the multi-project scheduling problem. *Decision Sciences*, 21, 387-402.

Gemmill, D.D., & Edwards, M.L. (1999). Improving resource-constrained project schedules with look-ahead techniques. *Project Management Journal*, 30 (3), 44-55.

Goldratt, E.M. (1984). *The goal*. Great Barrington, MA: The North River Press.

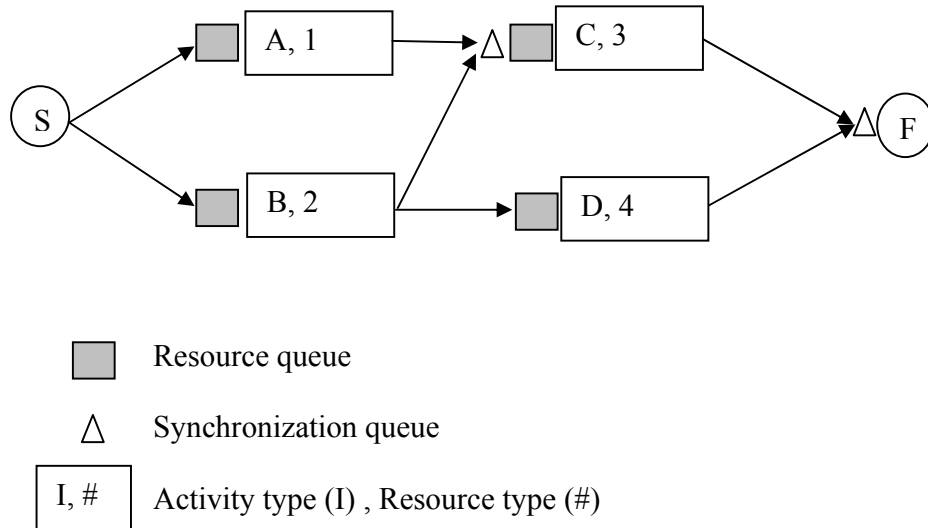
Goldratt, E.M. (1997). *Critical chain*. Great Barrington, MA: The North River Press.

Griffin, A. (2002). Product development cycle time for business-to-business products. *Industrial Marketing Management*, 31, 291-304.

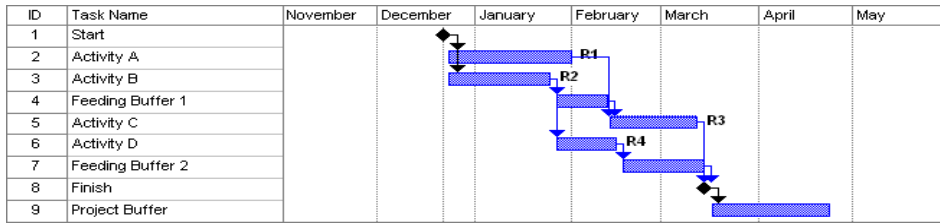
Herroelen, W., & Leus, R. (2001). On the merits and pitfalls of critical chain scheduling. *Journal of Operations Management*, 19 (5), 559-577.

- Herroelen, W., Leus, R., & Demeulemeester, E. (2002). Critical chain project scheduling: Do not oversimplify. *Project Management Journal*, 33 (4), 48-60.
- Hopp, W.J., & Spearman M.L. (1996). *Factory physics: foundations of manufacturing management*, USA: IRWIN.
- Law, A.M., & Kelton, W.D. (1991). *Simulation modeling and analysis*. USA: McGraw-Hill.
- Leach, P.L. (1999). Critical chain project management improves project performance. *Project Management Journal*, 30 (2), 39-51.
- Leung, H.K.N. (2002). Estimating maintenance effort by analogy. *Empirical Software Engineering*, 7, 157-175.
- Nelson, R., & Tantawi A.N. (1988). Approximate analysis of fork/join synchronization in parallel queues. *IEEE Trans. on Computers*, 37, 739-743.
- Parkinson, C.N. (1957). *Parkinson's Law*. Cambridge: The Riverside Press.
- Product Development Institute. (1999). Tutorial: Goldratt's critical chain method, a one-project solution. <http://www.pdinstitute.com>.
- Shou, Y., & Yeo, K.T. (2000). Estimation of project buffers in critical chain project management. *IEEE ICIMT 2000*, 162-167.
- Steyn, H. (2000). An investigation into the fundamentals of critical chain project scheduling. *International Journal of Project Management*, 19 (1), 363-369.

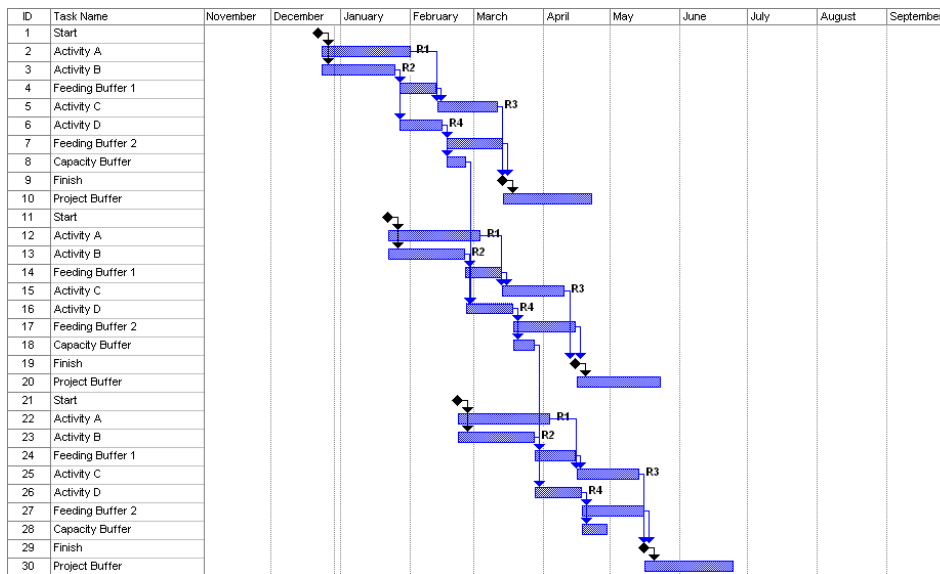
## Figures



**Figure 1.** The Stochastic Processing Network approach for representing a multi-project system.



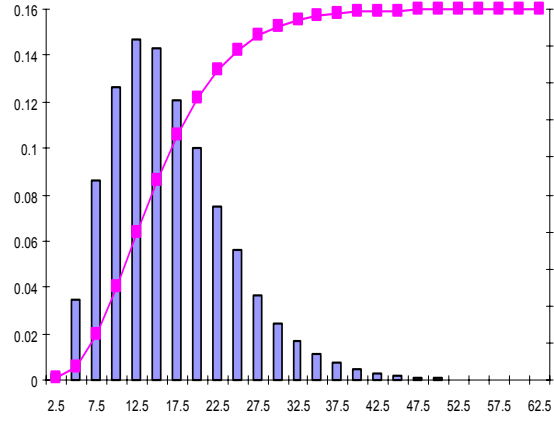
(a)



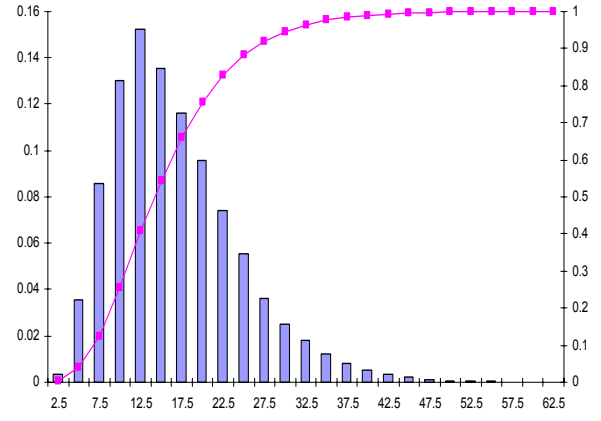
(b)

**Figure 2.** Critical Chain representation of a single project (a) and a corresponding multi-project system (b). For concreteness, one time unit is a week. The critical chain consists of Activity A and Activity C.

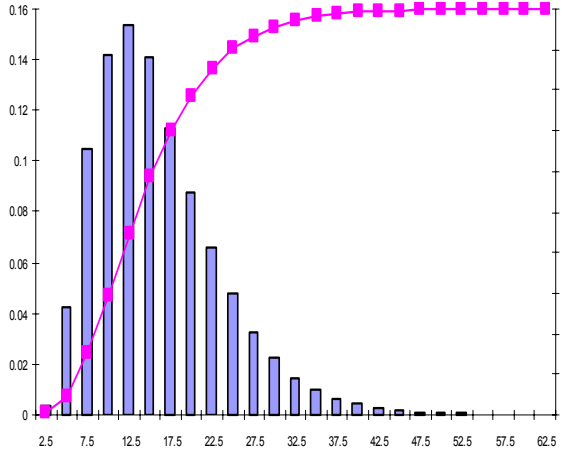
Feeding buffers are added at the end of non-critical chains: the non-critical chain that includes Activity B (Feeding Buffer 1) and one that includes activities B and D (Feeding Buffer 2). At the end of the critical chain we add a project buffer. Capacity buffers decouple successive projects: such a buffer is placed after an activity performed by Resource 4 (bottleneck resource) in one project and this activity in the following project.



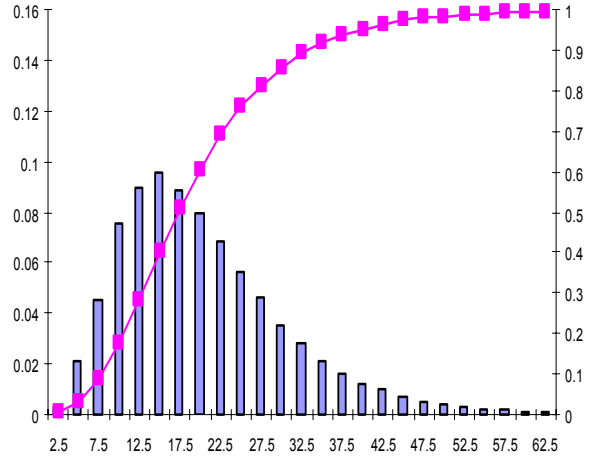
(a)



(b)



(c)

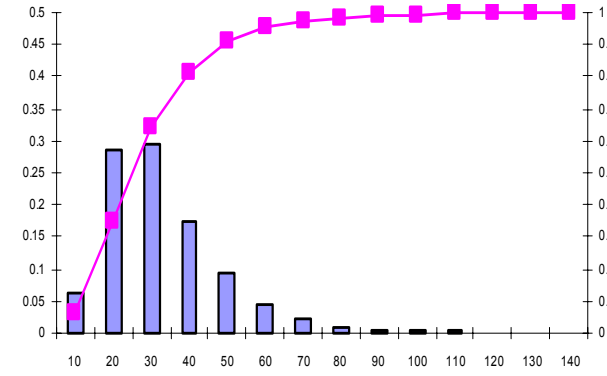


(d)

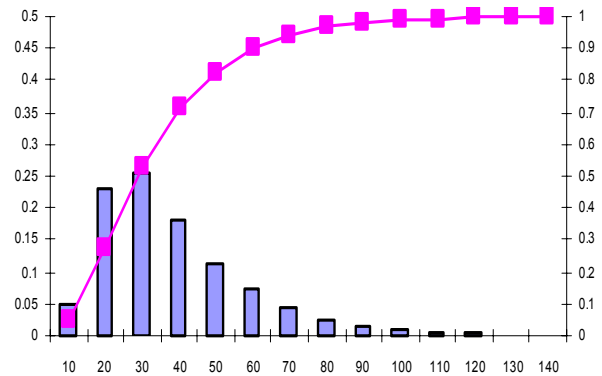
**Figure 3.** Throughput time distribution for throughput rate,  $\lambda=0.22$ , for the following controls:

(a) CC.  $\lambda_{\text{eff}}=0.22$ ,  $\bar{X}=15.34$ ,  $SD=7.66$ , (b) MinSLK.  $\lambda_{\text{eff}}=0.22$ ,  $\bar{X}=15.27$ ,  $SD=7.97$ ,

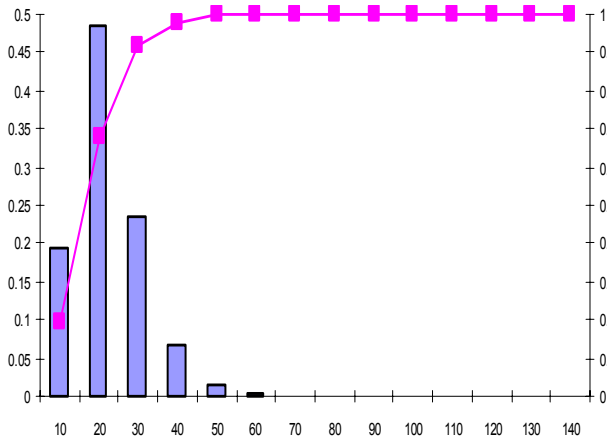
(c) QSC (3).  $\lambda_{\text{eff}}=0.21$ ,  $\bar{X}=14.60$ ,  $SD=7.52$  (d) No control  $\lambda_{\text{eff}}=0.22$ ,  $\bar{X}=18.9$ ,  $SD=10.10$



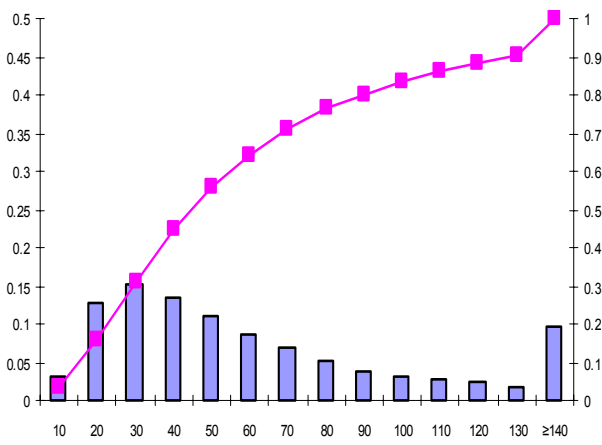
(a)



(b)



(c)

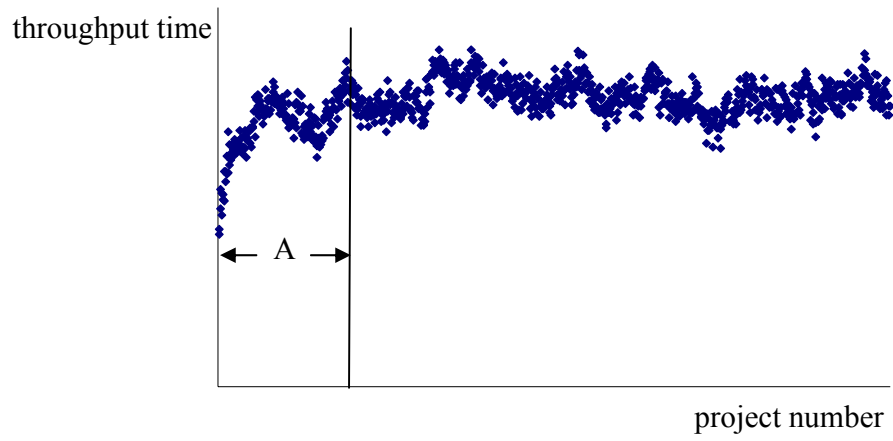


(d)

**Figure 4.** Throughput time distribution for throughput rate,  $\lambda=0.31$ , for the following controls:

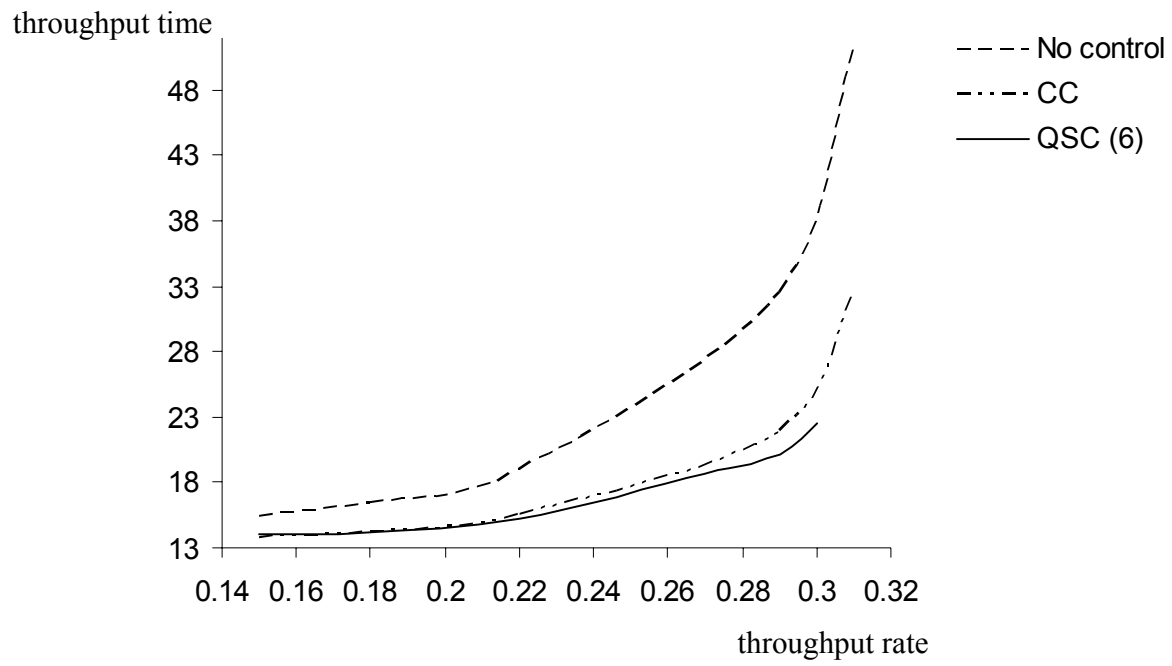
(a) CC.  $\lambda_{\text{eff}}=0.31$ ,  $\bar{X}=32.44$ ,  $SD=18.92$ , (b) MinSLK.  $\lambda_{\text{eff}}=0.31$ ,  $\bar{X}=30.40$ ,  $SD=17.67$ ,

(c) QSC (3).  $\lambda_{\text{eff}}=0.26$ ,  $\bar{X}=17.33$ ,  $SD=8.64$  (d) No control  $\lambda_{\text{eff}}=0.31$ ,  $\bar{X}=51.42$ ,  $SD=33.46$



**Figure 5.** An example of projects throughput time as it changes over a transient warm-up period (A) toward equilibrium.





**Figure 6.** Throughput time as a function of throughput rate, for three controls: No control, CC and QSC (6).

## Tables

|               | Resource type | Number of resources | Time Distribution |
|---------------|---------------|---------------------|-------------------|
| Inter-arrival |               |                     | Exp(1/3.25)       |
| Activity A    | 1             | 3                   | Exp(1/6)          |
| Activity B    | 2             | 2                   | Exp(1/5)          |
| Activity C    | 3             | 3                   | Exp(1/4)          |
| Activity D    | 4             | 1                   | Exp(1/3)          |

**Table 1.** Characteristics of our multi-project system: number of resource-units per type, processing time distribution and inter-arrival time distribution. The notation  $\text{Exp}(\lambda)$  represents an exponential distribution with probability density function  $f(t) = \lambda e^{-\lambda t}$  (and expectation  $1/\lambda$ ).

| $\lambda$ | Instance                         | No Control  | CC          | MinSLK      | QSC(6)      | QSC(3)      | ConPIP      |
|-----------|----------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0.31      | $\bar{X}$                        | 51.42       | 32.44       | 30.40       | 20.18       | 17.33       | 32.93       |
|           | $t_{n-1,1-\alpha/2} s/\sqrt{n}$  | 3.86        | 1.24        | 0.88        | 0.19        | 0.11        | 0.69        |
|           | SD                               | 33.46       | 18.92       | 17.67       | 9.42        | 8.64        | 13.45       |
|           | $\rho_1, \rho_2, \rho_3, \rho_4$ | 62,78,42,93 | 61,78,41,92 | 61,76,40,90 | 58,72,39,86 | 52,66,35,78 | 61,76,41,92 |
| 0.29      | $\bar{X}$                        | 32.44       | 21.87       | 21.59       | 18.62       | 16.62       | 22.79       |
|           | $t_{n-1,1-\alpha/2} s/\sqrt{n}$  | 1.75        | 0.41        | 0.37        | 0.13        | 0.12        | 0.20        |
|           | SD                               | 20.55       | 11.15       | 11.57       | 8.80        | 8.28        | 9.44        |
|           | $\rho_1, \rho_2, \rho_3, \rho_4$ | 57,72,38,85 | 57,72,38,85 | 57,71,38,85 | 56,69,36,82 | 51,63,33,75 | 57,72,38,86 |
| 0.22      | $\bar{X}$                        | 18.9        | 15.50       | 15.27       | 15.25       | 14.60       | 15.33       |
|           | $t_{n-1,1-\alpha/2} s/\sqrt{n}$  | 0.27        | 0.12        | 0.09        | 0.07        | 0.12        | 0.07        |
|           | SD                               | 10.10       | 7.71        | 7.97        | 7.74        | 7.52        | 7.56        |
|           | $\rho_1, \rho_2, \rho_3, \rho_4$ | 45,56,30,67 | 45,56,29,67 | 44,55,29,66 | 45,56,29,66 | 42,53,28,63 | 44,55,30,67 |
| 0.18      | $\bar{X}$                        | 16.49       | 14.17       | 14.26       | 14.26       | 14.12       | 14.81       |
|           | $t_{n-1,1-\alpha/2} s/\sqrt{n}$  | 0.17        | 0.08        | 0.06        | 0.05        | 0.09        | 0.07        |
|           | SD                               | 8.72        | 7.32        | 7.53        | 7.48        | 7.44        | 7.52        |
|           | $\rho_1, \rho_2, \rho_3, \rho_4$ | 37,46,25,55 | 36,45,24,55 | 37,46,24,55 | 37,46,24,56 | 36,45,23,55 | 36,45,24,54 |
| 0.15      | $\bar{X}$                        | 15.32       | 13.71       | 13.84       | 14.00       | 13.69       | 14.48       |
|           | $t_{n-1,1-\alpha/2} s/\sqrt{n}$  | 0.10        | 0.07        | 0.05        | 0.12        | 0.06        | 0.06        |
|           | SD                               | 7.99        | 7.18        | 7.42        | 7.23        | 7.33        | 7.51        |
|           | $\rho_1, \rho_2, \rho_3, \rho_4$ | 32,40,21,48 | 31,39,20,46 | 31,39,20,47 | 32,39,21,48 | 31,38,20,46 | 31,38,20,46 |

**Table 2.** Summary for different control methodologies. NPIP values for ConPIP control

are: 13,8,4,4,4 for throughput rates 0.31,0.29,0.22,0.18,0.15, respectively.

QSC controls'  $\lambda_{\text{eff}}$  values corresponding to  $\lambda = 0.31, 0.29, 0.22, 0.18, 0.15$  are: QSC(6)

0.29,0.27,0.22,0.18,0.15, respectively. QSC(3) 0.26,0.25,0.21,0.18,0.15, respectively.

| $\lambda$ | Time profile           | No Control | CC | MinSLK | QSC(6) | QSC(3) | ConPIP |
|-----------|------------------------|------------|----|--------|--------|--------|--------|
| 0.31      | % waiting time         | 43         | 36 | 38     | 24     | 18     | 38     |
|           | % synchronization time | 40         | 38 | 35     | 34     | 34     | 36     |
|           | % processing time      | 17         | 26 | 27     | 42     | 48     | 26     |
| 0.29      | % waiting time         | 36         | 27 | 26     | 21     | 16     | 29     |
|           | % synchronization time | 37         | 35 | 35     | 34     | 34     | 34     |
|           | % processing time      | 27         | 38 | 39     | 45     | 50     | 37     |
| 0.22      | % waiting time         | 22         | 13 | 12     | 11     | 9      | 12     |
|           | % synchronization time | 34         | 35 | 35     | 35     | 36     | 35     |
|           | % processing time      | 44         | 52 | 53     | 54     | 55     | 53     |
| 0.18      | % waiting time         | 15         | 7  | 7      | 7      | 6      | 9      |
|           | % synchronization time | 35         | 36 | 36     | 36     | 37     | 36     |
|           | % processing time      | 50         | 57 | 57     | 57     | 57     | 55     |
| 0.15      | % waiting time         | 20         | 5  | 5      | 5      | 4      | 8      |
|           | % synchronization time | 32         | 37 | 37     | 37     | 37     | 36     |
|           | % processing time      | 48         | 58 | 58     | 58     | 59     | 56     |

**Table 3.** Time profile analysis. Waiting time in resource queues, Synchronization time and Processing time.

**Izack Cohen** is serving in the Israeli Air-Force. He holds a B.Sc. in Chemical Engineering and M.Sc. in Materials Engineering from the Technion, Haifa, Israel. He is currently a Ph.D. candidate at the Faculty of Industrial Engineering and Management, Technion, Haifa, Israel. Izack has over 10 years of experience in managing engineering projects in different technological areas. His current research activities are focused on developing new methodologies for managing multi-project systems.

**Avishai Mandelbaum** is the Benjamin and Florence Free professor in Operations Research and Service Engineering, at the Faculty of Industrial Engineering and Management, Technion, Haifa, Israel. He has a B.Sc. in Mathematics and Computer Science, an M.A. in Statistics and a Ph.D. in Operations Research from Cornell University. His first academic position was at the Graduate School of Business at Stanford University, where he became interested in Project Management. At the Technion, Prof. Mandelbaum has been teaching courses in probability, stochastic processes and service engineering. His recent research activities have concentrated on queueing networks and their applications to Service Operations, with a focus on tele-services such as telephone call/contact centers. Prof. Mandelbaum was an Associate Editor of Mathematics of Operations Research (MOR) between 1991-1999. He is currently an associate editor of the journals Management Science, Queueing Systems Theory and Applications (QUESTA) and Manufacturing and Services Operations Management (MSOM). email: avim@tx.technion.ac.il

### **Avraham Shtub**

Avraham Shtub is the Sharon and Stephen Seiden professor of Project Management in the Industrial Engineering and Management faculty at the Technion – Israel Institute of Technology.

Professor Avraham Shtub has a B.Sc in Electrical Engineering from the Technion Israel Institute of Technology (1974), an MBA from Tel Aviv University (1978) and a Ph.D in Management Science and Industrial Engineering from the University of Washington (1982).

He is a senior member of the Institute of Industrial Engineering (USA) and a certified Project Management Professional (PMP) by the Project Management Institute (PMI-USA). He is the recipient of the Institute of Industrial Engineering 1995 “Book of the Year Award” for his Book “Project Management: Engineering, Technology and Implementation” (co-authored with John Bard and Shlomo Globerson), Prentice Hall, 1994. He is the recipient of the Production Operations Management Society; Wick Skinner Teaching Innovation Achievements Award for his book: “Enterprise Resource Planning (ERP): The Dynamics of Operations Management”, Kluwer, 1999.

Prof. Shtub served on the Editorial Boards of the Project Management Journal and the International Journal of Project Management. He is on the Editorial Boards of the IIE Transactions, and the International Journal of Production Research. email: shtub@ie.technion.ac.il