Theory-W Software Project Management: Principles and Examples

BARRY W. BOEHM, SENIOR MEMBER, IEEE, AND RONY ROSS

Abstract—**A good software project management theory should be simultaneously simple, general, and specific. To date, those objectives have been difficult to satisfy. This paper presents a candidate software management theory and shows that it satisfies those objectives reasonably well. Reflecting various alphabetical management theories (X, Y, Z), it is called the Theory W approach to software project management.**

Theory W: Make Everyone a Winner

The paper explains the key steps and guidelines underlying the Theory W statement and its two subsidiary principles: *plan the flight and fly the plan*; and, *identify and manage your risks*.

Several examples illustrate the application of Theory W, and an extensive case study is presented and analyzed: the attempt to introduce new information systems to a large industrial corporation in an emerging nation. The case may seem unique, yet it is typical. The analysis shows that Theory W and its subsidiary principles do an effective job both in explaining why the project encountered problems, and in prescribing ways in which the problems could have been avoided.

Index Terms—**Project management, software case studies, software development, software maintenance, software management, software personnel management, software planning and control.**

I. Introduction

SOFTWARE project management today is an art. The skillful integration of software technology, economics and human relations in the specific context of a software project is not an easy task. The software project is a highly people-intensive effort that spans a very lengthy period, with fundamental implications on the work and performance of many different classes of people.

A. The Software Project Manager's Problem

The software project manager's primary problem is that a software project needs to simultaneously satisfy a variety of constituencies: the users, the customers, the development team, the maintainance team, the management. As seen in Fig. 1, each of these constituencies has its own desires with respect to the software project. The *users*—sometimes too enthusiastic, sometimes too skeptical—desire a robust, user-friendly system with many functions supporting their mission. The *customers* desire a product delivered reliably to a short schedule and low budget. The *bosses* of the project manager desire a project with ambitious goals, no overruns, and no surprises. The *maintainers* of the product desire a well-documented, easy-to-modify system with no bugs. The *development team* members—often brilliant, sometimes unmanageable—desire interesting technical challenges and fast career paths, generally with a preference for design and an inclination to defer documentation.

These desires create fundamental conflicts when taken together (e.g., many functions versus a low budget and no overruns). These conflicts are at the root of most software project management difficulties—both at the strategic level (setting goals, establishing major milestones and responsibilities) and at the tactical level (resolving day-to-day conflicts, prioritizing assignments, adapting to changes).

B. The Software Management Theory Problem

A good software management theory should help the project manager navigate through these difficulties. As seen in Fig. 2, a software management theory has a similar challenging set of simultaneous objectives to satisfy. It should be simple to understand and apply; general enough to cover all classes of projects and classes of concerns (procedural, technical, economic, people-oriented); yet specific enough to provide useful, situation-specific advice.

Several attempts have been made to provide a relatively small set of software project management principles which can be easily recalled and applied, and which cover all of the important aspects. Thayer *et al.* [21] and Reifer [18] provide sets of principles largely organized around the five overall management principles in Koontz–O'Donnell [12] of planning, staffing, organizing, controlling, and directing. Boehm [3] provides a set of seven fundamental principles of software development. Although these have been very useful in many situations, none of these to date have produced a sufficient combination of simplicity, generality and specificity to have stimulated widespread use.

This paper presents a candidate fundamental principle for software project management developed by one of the authors (Boehm), and shows how it would apply in avoiding the software project management problems encountered in a case study analyzed by the other author (Ross).

The fundamental principle is called the Theory W approach to software project management.

Theory W: Make Everyone a Winner.

Manuscript received October 30, 1987; revised February 29, 1988.

B. W. Boehm is with TRW Defense Systems Group, One Space Park, Redondo Beach, CA 90278, and the Department of Computer Science, University of California, Los Angeles, CA 90024.

R. Ross is with the Department of Computer Science, University of California, Los Angeles, CA 90024.

IEEE Log Number 8928293.

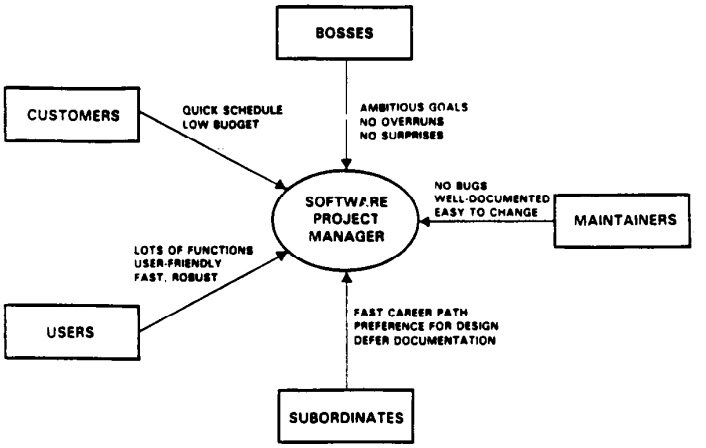

Fig. 1. The software project manager's problem.

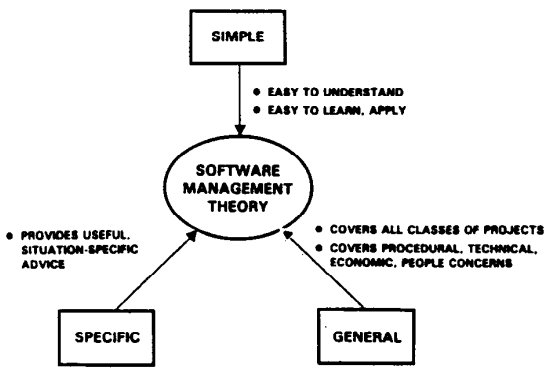

Fig. 2. The software management theory problem.

It holds that the primary job of the software project manager is to make winners of each of the parties involved in the software process: the project manager's subordinates and managers; the customers; the users and maintainers of the resulting product; and any other significantly affected people, such as the developers or users of interfacing products.

Making everyone a winner has a number of implications which will be discussed below, including the use of two subsidiary principles:

- Plan the flight and fly the plan.
- Identify and manage your risks.

Section II of this paper elaborates on the overall Theory W approach and the software project implications of making everyone a winner. Section III elaborates on the two subsidiary principles. Section IV provides the history of the system involved in the case study. Section V analyzes the case study with respect to Theory W and the subsidiary principles, and Section VI presents the resulting conclusions.

II. Theory W: Make Everyone a Winner

This section elaborates on Theory W's major principle. We begin in Section II-A by placing Theory W in the context of other management theories, particularly Theories X, Y, and Z. Section II-B presents the key concept involved in Theory W: the distinction between win–win, win–lose, and lose–lose situations. Section II-C summarizes the three primary steps suggested to achieve the desired goal of making everyone a winner, and the nine substeps involved in implementing Theory W. Section II-C also elaborates on the first three substeps: those that deal with creating win–win situations, the strongest distinguishing feature of Theory W as a management approach. Section II-D elaborates on all of the substeps, and shows how a set of strategic principles for software project management can be generated by applying each of the substeps to each of the project manager's constituencies identified in Fig. 1 above. Section II-E shows via an example how the Theory W steps can be used to solve day-to-day tactical project management problems as well as strategic problems.

A. Comparison to Theories X, Y and Z

The Theory X approach to management built largely on the "scientific management" ideas of Frederick Taylor [20]. It held that the most efficient way to get a job done was to do more and more precise time and motion studies, and to organize jobs into well-orchestrated sequences of tasks in which people were as efficient and predictable as machines. Management consisted of keeping the system running smoothly, largely through coercion.

Theory Y, introduced in [8], held that Theory X was a poor long-term strategy because it stunted people's creativity, adaptiveness, and self esteem, making the people and their organizations unable to cope with change. Theory Y held that management should stimulate creativity and individual initiative. This led to organizations which were much more adaptive and personally satisfying, but created difficulties in dealing with conflict. This was not a problem in Theory X, but became a major concern in Theory Y organizations, with many individual initiatives competing for resources and creating problems of coordination.

Theory Z, described in [10], holds that much of the conflict resolution problem can be eliminated by up-front investment in developing shared values and arriving at major decisions by consensus. It focuses largely on doing this within an organization, and does not say much about how to deal with other organizations with different objectives and cultures—a particularly common situation with software managers and their diverse constituencies (developers, customers, users, etc.). Overall, Theory Z's primary emphasis is at the corporate-culture level rather than at the intercompany level or the individual project level.

Theory W's fundamental principle is well-matched to the problems of software project management. It holds that software project managers will be fully successful if and only if they make winners of all the other participants in the software process: superiors, subordinates, customers, users, maintainers, etc. This principle is particularly relevant in the software field, which is a highly people-intensive area whose products are largely services or decision aids, and whose performers are often unfamiliar with user and management concerns. However, Theory W can be applied to other fields as well.

Rather than characterizing a manager as an autocrat (Theory X), a coach (Theory Y), or a facilitator (Theory Z), Theory W characterizes a manager's primary role as a negotiator between his various constituencies, and a packager of project solutions with win conditions for all parties. Beyond this, the manager is also a goal-setter, a monitor of progress towards goals, and an activist in seeking out day-to-day win-lose or lose-lose project conflicts, confronting them, and changing them into win-win situations.

B. Win-Win, Win-Lose, and Lose-Lose Situations

Making everyone a winner may seem like an unachievable objective. Most situations tend to be zero-sum, win-lose situations. Building a quick and sloppy product may be a low-cost, near-term "win" for the software developer and customer, but it will be a "lose" for the user and the maintainer. Adding lots of marginally useful software "bells and whistles" to a product on a cost-plus contract may be a win for the developer and some users, but a lose for the customer.

At worst, software projects can be lose-lose situations. Setting unrealistic schedule expectations; staffing with incompatible people; poor planning; or trying to catch up on a schedule by adding more people will generally make losers of all the participants.

Nonetheless, win-win situations exist, and often they can be created by careful attention to people's interests and expectations. Creating a profit-sharing arrangement for a software subcontractor provides the subcontractor with a motivation to develop a high-quality, widely-sold product, thus increasing the size of the profit pie for both the subcontractor and the top-level product developer. Using better software technology such as structured programming, early error detection techniques, or information hiding will also create wins for all parties.

C. Creating Win-Win Situations

The best work on creating win-win situations has been done in the field of negotiation. The book *Getting to Yes* [9] is a classic in the area. Its primary thesis is that successful negotiations are not achieved by haggling from preset negotiating positions, but by following a four-step approach whose goal is basically to create a win-win situation for the negotiating parties:

1) Separate the people from the problem.
2) Focus on interests, not positions.
3) Invent options for mutual gain.
4) Insist on using objective criteria.

The Theory W approach to software project management expands on these four steps to establish a set of win-win preconditions, and some further conditions for structuring the software process and the resulting software product, as shown in Table I.

The remainder of this section elaborates on the first three substeps in Table I which deal primarily with the process of creating win-win situations.

1) Understand How People Want to Win: One important subprinciple here is to *make sure you identify the key people*. Often, software projects have failed because a key constituency (users' bosses, hardware procurement personnel, subcontractors) has not been included in the win-win scheme.

Another important subprinciple is to *project yourself into others' win situations*. This is often difficult for people to do because it runs counter to strongly implanted notions of goodness such as the Golden Rule: "Do unto others as you would have others do unto you." But, others may not want what you want as win conditions. Some frequent examples:

- Managers frequently assume that software professionals win by getting "promoted" to management. However, the motivating-factors studies done by Couger and Zawacki [6] indicate that the typical data processing professional has a much stronger need for professional growth than for social interaction, while the average manager has the opposite profile. Thus, promotions to management can be quite harmful to software people's careers, and dual-track (technical and managerial) career-path ladders can be much more successful in software organizations.
- Computer-science majors brought up on canonical applications such as compilers and operating systems, where users are programmers, implicitly build up a set of assumptions about software users: that software users like to program, and prefer powerful and terse (but perhaps obscure) command languages and users' manuals. Well-meaning attempts to apply those assumptions to such software users as nurses, doctors, pilots and bank tellers have led to numerous software disasters.

Thus, Theory W suggests a modified form of the Golden Rule: "Do unto others as you would have others do unto you—if you were like them."

TABLE I
THEORY W WIN-WIN STEPS

1. Establish a set of win-win preconditions
a. Understand how people want to win; b. Establish reasonable expectations; c. Match people's tasks to their win conditions; d. Provide a supportive environment.
2. Structure a win-win software process.
a. Establish a realistic process plan; b. Use the plan to control the project; c. Identify and manage your win-lose or lose-lose risks; d. Keep people involved;
3. Structure a win-win software product.
a. Match product to users', maintainers' win conditions.

Another key subprinciple is the Peters-Waterman [17] maxim to *get close to the customer*. This involves getting software people to operate more like marketing personnel than like people who wait around to code up whatever specification is provided. It involves much more proactive use of interviews, surveys, tours of duty, prototypes, scenarios, operations analysis, user-culture analyses, and understanding of users' previous experiences with automation (scars, bruises, traumas, triumphs).

Overall, the field of motivational analysis provides the most comprehensive set of insights on understanding how people want to win. Gellerman [10] provides a good early survey of the field; more recently, Couger and Zawacki [6] have provided a good set of insights related specifically to data processing people.

2) Establish Reasonable Expectations: Many software problems stem from the fact that software customers and users frequently have little feel for what is easy and what is hard to achieve with computers and software. This leads to a set of unrealistic expectations: either thinking things are too hard to implement (complex scheduling or file management) or too easy (pattern recognition or building 150 man-months worth of software in 6 months). Similarly, software people often have unrealistic expectations of what is easy and what is hard for users to do.

Some important subprinciples here are:

- Bring your constituencies together to identify and resolve expectation mismatches.
- Have people look at issues from the other constituents' viewpoints.
- Have people look for objective, mutually relevant solution criteria.
- Relate people's expectations to experience: benchmarks, reference checks, expert judgment.
- Relate people's expectations to well-calibrated models: computer-performance models, software project cost and schedule estimation models.

A related management insight is that "hard-soft works better than soft-hard." A manager who overpromises to his various constituencies and then has to deflate their expectations has an easier time initially, but a much rougher time in the long run, than a manager who deflates initial expectations and provides some management reserve to soften his position later where necessary.

A good recent example of establishing reasonable software project expectations involved the need for improvements in the on-board software of the F-16 aircraft. The aircraft users expected a long list of additional software capabilities to be delivered in 12 months. The developers' expectations were in terms of previous software productivity rates, and indicated a much longer development period. Rather than conduct a positional bargaining exercise resulting in unsatisfied expectations on both sides, the users and developers decided to explore their options using COCOMO, a software cost and schedule estimation model calibrated to experience in similar projects [2].

As a result, both groups developed a much better understanding of the relationships between software functionality, cost, and schedule. The developers found options to increase their software productivity capabilities and expectations. The users were able to establish a series of prioritized annual software increments whose achievability was keyed to their developer-shared productivity expectations. After two years of software deliveries, both groups have experienced satisfactory results relative to their revised expectations.

Overall, the process of reconciling people's expectations is dealt with in the fields of conflict resolution and teambuilding. Walton [22], Kirchof and Adams [11], and Dyer [7] are good sources of additional insight.

3) Match People's Tasks to Their Win Conditions: The key principles here involve *searching out win-win situations* and *expanding the option space to create win-win situations.*

Some effective techniques available to the software project manager for searching out win-win situations include:

- Breaking options into parts (functions, activities, increments, phases), and configuring combinations of suboptions into win packages for each participant. For example, under some conditions, establishing a separate leader for successive software increments has worked well, particularly if the increments are large, with different technical and/or organizational centers of gravity.
- Realigning options along win-win axes. For example, some projects have successfully shifted the authority and responsibility for software quality assurance from the developer (who may consider it a bore) to the maintainer, who has considered it a major win-leverage opportunity.

Some effective techniques available to the software project manager for expanding the option space to create win-win situations are:

- Linking tasks to future options and career paths ("Quality assurance may be a bore, but it's a ticket to a fast-track career path").
- Expanding the scope of a task ("Quality Assurance should not be a bore. I think you could lead the way in helping us make quality assurance a more proactive function in getting us quality products. That would be a real achievement").
- Linking tasks to extra rewards ("Rescuing this integration and test mess will be a killer, but I'll make sure you get a good bonus and lots of kudos if you succeed").
- Providing extra support ("This schedule is very am-

bitious, but I'll provide your team with the first-class workstations and facilities you'll need to meet it").

- Surfacing new options ("We can't develop all the functions in 12 months, but if we do an incremental development, we can satisfy your top-priority needs in 12 months").

Overall, the field of negotiation provides the best additional sources of insight in matching tasks to win conditions. Some good books are Fisher and Ury [9] and Nierenberg [15].

D. Deriving Strategic Project Guidelines from Theory W Win-Win Steps

Most current software management directives, and many of the textbooks, present strategic software management guidelines as a series of relatively unconnected what-to-do lists of activities to perform (e.g., prototype the user interface, configuration-manage the baselined items, set up and follow a set of programming standards).

The power of Theory W becomes evident in Tables II and III, which show that one can derive most of the apparently unconnected what-to-do activities by applying the Theory W win-win steps in Table I to the various constituencies involved in the software process. Prototyping is a way of understanding the users' win conditions (Table II). Configuration management is partly establishing a supportive environment for the developers and maintainers, and partly participation in change control by all parties impacted by a proposed change (Table II). Programming standards contribute to structuring a software product so that its maintainers will be winners (Table III).

Further, Tables II and III provide stronger guidance than usual for allocating life-cycle responsibilities to the various software parties. An example is the allocation of the quality assurance responsibility to the maintainers, as their win conditions are most strongly affected by product quality.

Tables II and III also show that Theory W provides not just a "what" for the process activities, but also the underlying "why." This is very important in the frequent situations of tailoring the process activities to special circumstances, and determining how much of a given process activity is enough. For example, if the inclusion of machine-generated flowcharts in the maintainance documentation does not help the maintainers become winners, it is not necessary to require their delivery.

E. Theory W: A Tactical Management Example

Theory W provides specific useful guidance in tactical as well as strategic project management situations. The resulting solutions are often preferable to those derived from previous management theories. Consider the following example:

> XYZ Corp. has been developing a large financial system for a Boston bank. A new position on the project is being created to lead a system analysis effort. George and Ann are the two primary candidates for the job. They are equally well qualified: George has somewhat more overall experience, while Ann has more experience specific to this type of application. The project manager must decide whom to chose.

Using Theory X, the manager would make a choice, based on some arbitrary criterion such as seniority. Using Theory Y, the manager would likely ask George and Ann for proposals on how they would do the job, and pick the most ambitious one. Using Theory Z, the manager would likely concentrate on prebuilding a consensus on team objectives, and make a choice based on team priorities.

TABLE II
STRATEGIC GUIDELINES DERIVED FROM WIN-WIN PRECONDITIONS

Win-Win Precondition	Users	Maintainers	Customers	Developer Team
Understand win conditions	Mission anal. Ops. concept Prototyping Rqts. spec Early users' manual	Ops. concept Ops. procedures	Cost-benefit analysis	Career path develop.
Reasonable expectations	*Teambuilding, Negotiating, Conflict resolution*			
	Rqts. scrub		Resource allocation	
Match tasks to win conditions	*Change control participation*			
	User-spec reviews Prototype exercise	Quality assurance	Status tracking	Staffing, organizing
			Early Error Detection	
Supportive environment preparation	User training Cutover preparation	Maint. training Conversion Deliverable support envir. Config. mgmt.	Customer training	Developer training Support envir. Config. mgmt.
		Modern programming practices		

TABLE III
STRATEGIC GUIDELINES DERIVED FROM PRODUCT, PROCESS GUIDELINES

Guideline	Users	Maintainers	Customers	Developers
Process planning	Operational plan Installation & training plans	Life-cycle support plan	Development plans	
Process control	*Teambuilding, Negotiating, Communicating*			
	Reviews	Reviews	Status tracking, Controlling	Perform. feedback
Risk management	*Sensitivity analysis* *Risk management plans*			
	User rqts. validation, stability	Quality assurance	Budget, schedule Validation	staffing
Process involvement	Sys. engr. plan participation Review participation Prototype exercise	Sys. engr. plan participation Review participation Quality assurance	Cost-benefit reviews, approvals	Delegation Planning particip.
Product structuring	Service oriented Efficient Easy to learn Easy to use Tailorable	Easy to modify Prog. standards	Efficient Correct Feasible	Easy to Modify Balanced Correct

Theory W would try to avoid the above situations, each of which creates a win-lose situation between George and Ann. By following the Theory W steps in Table I, the manager would try to create a win-win situation as follows:

1) Understand how people want to win. In talking with George and Ann, the manager finds that George greatly wants the job because of the extensive travel to Boston, where he has a daughter in college. Ann greatly wants the job because it would provide a career path toward marketing.

2) Match people's tasks to their win conditions. The manager expands the option space by considering comparable jobs with Boston travel for George and comparable marketing-oriented jobs for Ann.

Frequently, the Theory W approach will help the manager to find and establish such win-win solutions, creating more satisfaction and personal commitment among the participants, fewer disaffected and uncooperative participants, and more satisfactory all-around outcomes.

F. Connections between Theory W and Game Theory

Theory W also has fruitful connections to game theory. For example, the case of George and Ann can be formulated as a nonzero-sum game involving three players: George, Ann, and the customer. By using the concept of Rational Offer Groups formulated by Rosenschein and Genesereth [19], one can analyze the conditions under which the expansion of George's and Ann's option spaces will produce a win-win-win situation for George, Ann, and the customer. An example result is that if the project manager is too successful in finding alternate jobs for George and Ann, neither will take the systems analysis job, and the customer will become a loser.

III. Theory W Subsidiary Principles

Because of their particular importance to the management of the software process, the first three Theory W win-win process substeps in Table I are highlighted and combined into two key Theory W subsidiary principles. These are:

- Plan the flight and fly the plan (steps 2a, 2b).
- Identify and manage your risks (step 2c).

A. Planning the Flight

Establishing a realistic process plan is crucial to the success of the project. As indicated in Table III, there are several types of plans involved in making everyone a winner: operational plans, installation and training plans, lifecycle support plans, and development plans. Each of these may have a number of subsidiary plans: configuration management plans, quality assurance plans, test plans, conversion plans, etc.

Plans are important in Theory W because:

- They record the mutual commitment of the project participants to a set of win-win conditions and their implications.
- They provide a framework for detecting deviations from the win-win conditions which require corrective action.

Frequently, each software subplan is organized around a totally different outline, making the various plans more difficult to develop, assimilate, and query. Each Theory W plan is organized around a common outline, reflecting a small number of universal interrogatives (why, what, when, who, where, how, and how much):

1) Objectives (*Why* is the activity being pursued?)
2) Products and Milestones (*What* is being produced by when?)
3) Responsibilities (*Who* is responsible for each result? *Where* are they located organizationally?)
4) Approach (*How* is each result being achieved?)
5) Resources (*How much* of each scarce resource is required to achieve the results?)

Fig. 3 presents the outline for one of the key software management plans: the software development plan. It shows that the subsections of the plan are particular to software development issues (requirements, product design, programming, configuration management, quality assurance, etc.), but that the major sections of the plan follow the common Theory W outline.

Space limitations preclude further discussion of software project planning here; some good references are [8] and [14]. Also, some similar concepts are being developed in the draft IEEE Standard for Software Project Management Plans.

B. Flying the Plan

Developing a plan which satisfies everyone's win conditions is not enough to make everyone a winner. You also need to use the plan to manage the project.

This involves making a particular effort to monitor the project's progress with respect to the plan. The nature of this effort should be specified in the plan; see section 5.3 of the plan outline in Fig. 3. If the project's progress continues to match its plans, the project is in good shape. But usually, there will be some mismatches between the progress and the plans. If so, the manager needs to assess the reasons for the mismatches. It may be that the plans are flawed or out of date, in which case the plans need to be modified. Or the project's progress may be deficient, in which case the project manager needs to apply corrective action.

Applying corrective action is one of the most critical situations for using the "make everyone a winner" principle. It is all too easy to apply snap-judgment corrective actions with win-lose or lose-lose outcomes, or to heap public blame on people so that they feel like losers rather than winners. But it is generally possible to follow the Theory W win-win steps in Table I to find a corrective action strategy which either preserves everyone as winners, or convinces them that their losses are minimal with respect to other strategies. (An example is provided in the case study analysis in Section V-A.) And it is generally possible to reprimand people's behavior without making

1. Objectives (the "why")
   - 1.1. Software Product Objectives
   - 1.2. Development Plan Objectives
2. Milestones and Products (the "what" and "when")
   - 2.1. Overall Development Strategy
   - 2.2. Detailed Schedule of Deliverables
   - 2.3. Detailed Development Milestones and Schedules
3. Responsibilities (the "who" and "where")
   - 3.1. Organizational Responsibilities
     - 3.1.1. Global Organization Charts
     - 3.1.2. Organizational Commitment Responsibilities
   - 3.2. Development Responsibilities
     - 3.2.1. Development Organization Charts
     - 3.2.2. Staffing
     - 3.2.3. Training
4. Approach (the "how")
   - 4.1. Risk Management
   - 4.2. Development Phases
     - 4.2.1. Plans and Requirements Phase
     - 4.2.2. Product Design Phase
     - 4.2.3. Programming Phase
     - 4.2.4. Integration and test Phase
     - 4.2.5. Implementation Phase
   - 4.3. Reviews
   - 4.4. Documentation
   - 4.5. Configuration Management
   - 4.6. Quality Assurance
   - 4.7. Facilities and Related Concerns
5. Resources (the "how much")
   - 5.1. Work Breakdown Structure
   - 5.2. Budgets
   - 5.3. Status Monitoring and Control

Fig. 3. Theory W outline for the software development plan.

them feel like losers. A good example is the "one-minute reprimand" in the book *The One-Minute Manager* [1].

C. Risk Management

Planning the flight and flying the plan will make everyone a winner if the plans reflect the participants' win conditions and if the plans are realistic. Ensuring that the plans are realistic is the province of risk management.

Risk management focuses the project manager's attention on those portions of the project most likely to cause trouble and to compromise the participants' win conditions. Risk management considerations can also help the project manager to determine the appropriate sequence of performing project activities. The spiral model of software development [4] discusses risk-driven sequencing of project activities in more detail.

Webster defines "risk" as "the possibility of loss or injury." The magnitude of a risk item is generally defined as a quantity called Risk Exposure *RE*:

$$RE = (LP) * (LM).$$

The Loss Probability factor *LP* represents the probability of an unsatisfactory outcome. The Loss Magnitude factor *LM* represents the magnitude of the loss if the outcome is unsatisfactory. The magnitude of the loss is best expressed in terms of the participants' utility functions, which measure the degree to which the participants become losers rather than winners.

There are two primary classes of project risk:

1) *Generic risks*, which are common to all projects, and which are covered by standard development plan techniques.

2) *Project-specific risks*, which reflect a particular aspect of a given project, and which are addressed by project-specific risk management plans. The most common project-specific risks are personnel shortfalls, unrealistic schedules and budgets, inappropriate requirements, shortfalls in external components and tasks, and technology shortfalls or unknowns.

D. Risk Management Steps

The practice of risk management involves two primary steps, Risk Assessment and Risk Handling, each with three subsidiary steps. Risk Assessment involves risk identification, risk analysis, and risk prioritization. Risk Handling involves risk management planning, risk management execution, and risk monitoring and control.

Risk Identification produces lists of the project-specific items likely to compromise a project's win-win conditions. Typical risk identification techniques include checklists, decomposition, comparison with experience, and examination of decision drivers.

Risk Analysis produces assessments of the loss-probability and loss-magnitude associated with each of the identified risk items, and assessments of compound risks involved in risk-item interactions. Typical techniques include network analysis, decision trees, cost models, and performance models.

Risk Prioritization produces a prioritized ordering of the risk items identified and analyzed. Typical techniques include risk leverage analysis and Delphi or group-consensus techniques.

Risk Management Planning produces plans for addressing each risk item, including the coordination of the individual risk-item plans with each other and with the overall project plan (e.g., to ensure that enough up-front schedule is provided to properly develop, exercise, and learn from a prototype). Typical techniques include risk-resolution checklists such as the one in Table IV, showing the top 10 primary sources of software project risk and the most effective approaches for resolving them. Other techniques include cost-benefit analysis and statistical decision analysis of the relative cost and effectiveness of alternative risk-resolution approaches. The best form for a risk management plan is the general "why, what, when, who, where, how, how much" plan template discussed above.

Risk Management Execution produces a resolution of the risk items. Typical techniques are the ones shown in Table IV.

Risk Monitoring and Control completes the "flying the plan" counterpart of risk management planning. It involves tracking the progress toward resolving high-risk items and taking corrective action where appropriate. A most effective technique is a Top Ten Risk Item list which

TABLE IV
A TOP TEN LIST OF SOFTWARE RISK ITEMS

A Top Ten List of Software Risk Items	
RISK ITEM	RISK MANAGEMENT TECHNIQUES
1. Personnel shortfalls	-Staffing with top talent; job matching; teambuilding; key-personnel agreements; cross-training; prescheduling key people
2. Unrealistic schedules and budgets	-Detailed multisource cost & schedule estimation; design to cost; incremental development; software reuse; requirements scrubbing
3. Developing the wrong software functions	-Organization analysis; mission analysis; ops-concept formulation; user surveys; prototyping; early users' manuals
4. Developing the wrong user interface	-Prototyping; scenarios; task analysis; user characterization (functionality, style, workload)
5. Gold plating	-Requirements scrubbing; prototyping; cost-benefit analysis; design to cost
6. Continuing stream of requirements changes	-High change threshold; information hiding; incremental development (defer changes to later increments)
7. Shortfalls in externally furnished components	-Benchmarking; inspections; reference checking; compatibility analysis
8. Shortfalls in externally performed tasks	-Reference checking; pre-award audits; award-fee contracts; competitive design or prototyping; teambuilding
9. Real-time performance shortfalls	-Simulation; benchmarking; modeling; prototyping; instrumentation; tuning
10. Straining computer science capabilities	-Technical analysis; cost-benefit analysis; prototyping; reference checking

is highlighted at each weekly, monthly, or milestone project review.

These steps are supported by a variety of techniques. Space limitations preclude further discussion of the issues here. Further details on each of the software risk management steps are given in [5].

IV. THE CASE STUDY

A. Corporate Background

BBB Industries is one of the largest manufacturers in the small, yet advanced emerging nation named Optimia. The company started out in the 1950's as a privately owned workshop, and has gone through periods of prosperity and periods of recession. During one of the recession periods in the early seventies, the owners sold their shares to MMM corporation, one of Optimia's largest investment corporations.

In 1983, BBB Industries' sales volume reached $100 million a year, with over 3000 employees. The manufacturing was carried out in several factories while the Marketing, Production Planning, and Financial Services functions were all concentrated at the company's headquarters. BBB Industries manufactured various consumer products that were marketed through diverse distribution channels, including the company's own store. Over half of the sales were directed to export markets in the USA and Europe.

The profitability of the company was very unstable: the world demand for BBB's product line is subject to frequent ups and downs, and BBB Industries was unable to adjust in time to these dynamic changes. This inability was attributed mainly to BBB's old-fashioned production and organizational methods.

BBB's Information Systems in 1983 were of the most archaic type. In the early 1970's a major effort was made to computerize the production and control systems by using a card-operated computer. This effort failed, and a decision was made to transfer the information processing to a service bureau. For technical and political reasons, the various departments adopted different service bureaus, so that in 1983 each of the General-Ledger, Accounts-Receivables, Payroll and Inventory systems used the services of a different service bureau.

B. The New Management's Attitude

In 1984, a new General Manager was appointed to BBB Industries. The business results of 1984 were good, and the General Manager decided that the time had come to do something about BBB's Information Systems. To achieve that result, he hired a new manager for the Data Processing department, Mr. Smith.

"It's not going to be an easy job," he told Mr. Smith, "But this is a big challenge. I know this company cannot go on without proper information systems. However, my middle management does not understand information systems concepts. It is up to you to show us the way, and to help me convince the other managers in this company to give a hand to this effort. However—you should not forget that BBB's budget is limited, and that 1985 is not going to be as profitable as 1984. So, we shall have to do our best with a minimal budget. And, of course, since I am trying to cut down on all personnel, you cannot hire any more people to the data processing department right now. First, I want to see some results, and then—the sky is the limit."

C. The Initial Survey

The initial survey was done by Mr. Smith himself. The survey consisted of two parts:

a) A study of BBB's existing systems.

b) An outline of BBB's requirements for new Information Systems.

The survey's findings can be summed up as follows:

- Except for the Payroll system, all the existing data-processing systems of BBB did not serve their purposes. These systems were not used in the day-to-day operations, their accuracy was very low, and they therefore required a lot of manual processing.
- The vital Production Design and Control operation could not benefit at all from any of the computer systems, and therefore was slow, inflexible, and inefficient.
- There was practically no integration between the different systems, and each served the specific, limited needs of the department that was in charge of it.
- BBB's productivity, manageability, and profitability depended on the replacement of these systems by new, better ones.
- The potential users of the systems were quire ignorant of what modern information systems concepts are, and how they could be of use for them in their daily ac-

tivities. Furthermore, the factory workers had little faith in BBB's ability to adopt new, modern methods.

The survey's recommendations were:

- There is immediate need to replace the existing systems by on-line, interactive systems, based on in-house computers, that will supply the information by both operational and management levels in a timely, accurate, and comprehensive fashion. This effort can be done in stages, and the first system to be implemented should be a relatively simple, low-risk system. The success of this implementation will improve the ability to continue with other, more complex systems.
- The development of the first system should be done by an outside contractor, preferably a software house that already has a package for that purpose.
- BBB's middle management personnel should receive special training that will enable them to better understand the potential of on-line computer systems and their applicability to their own problems.
- The problems of the factories are complex, and require more detailed research to analyze and define the information systems requirements of the factories and to evaluate the various modes of operations that are amenable for this problem (distributed processing versus centralized processing, interactive versus autonomous, data collection techniques, etc.).
- Even though the task of computerizing BBB is complex, such projects are common nowadays, and the overall timetable should not exceed three years.

The survey was presented to BBB's management, and its conclusions were approved enthusiastically. The Finished-Goods Sales and Marketing system (FGSM) was chosen for first implementation, primarily because it was the easiest to implement, and because the FGSM managers were the strongest in expressing their need for and support of a new system. Mr. Smith was charged with preparing a Request for Proposal that would be presented to potential suppliers of software and hardware. There was no discussion of the required budget, nor additional personnel.

D. The Request for Proposal (RFP)

The RFP was based on the initial survey and on the findings of a subsequent two-week survey of the Finished-Goods Sales and Marketing organization. It consisted of the following parts:

a) A general description of BBB, its organization, operations, and goals.

b) A thorough, although not detailed, description of the Finished-Goods Marketing and Sales Organization.

c) A list of the requirements for the new system for FGSM:

- The system should be an on-line, interactive system.
- The system shall handle all the different types of items and incorporate all the different types of Catalog Codes that are in current use.
- The system shall handle the Finished Goods inventory in various levels of detail.
- The system shall handle the various types of clients (retailers, wholesalers, department stores, company-owned stores).
- The system shall produce automatic billings to the various clients (some of the department stores required predefined forms).
- The system shall be able to produce different sales and inventory reports.
- The system shall be able to integrate in the future into the General Ledger and Accounts Receivable Systems

d) A four-page outline of the requirements for the new Financial Systems for BBB.

The RFP was presented to the three leading hardware suppliers in Optimia, and to five software companies that had previous experience in similar systems.

E. The Proposals

After the first elimination process, three proposals were left in the game. Since the RFP was rather open-ended, the proposals varied in their scopes and in the extent to which they covered the requirements mentioned. The price quotations ranged from $70,000 to $450,000. The final competitors were as follows.

1) Colossal Computers: The leading hardware distributor in Optimia. Colossal Computers proposed their popular System C computer, and recommended the software packages of SW1 Software as the basis for the implementation. (Colossal refused to take full commitment for both hardware and software.)

2) Big Computing Computers: The second largest hardware distributor in Optimia, distributors of Big computers, with their own Financial and Marketing packages.

3) Fast Computing Computers: The distributors of world renowned Fast computers. There were only few installations of Fast computers in Optimia, even though the equipment was excellent. As a result, there were no software packages available on Fast Computers. The owners of Fast Computing Computers was MMM Corp., the owners of BBB Industries. MMM Corp. was deliberating at the time how to increase the sales of Fast Computers.

Table V summarizes the results of the evaluation process among the three competitors, as presented to BBB's management.

Mr. Smith's recommendation was to buy Colossal's equipment and to engage SW1 Software as subcontractor for the Marketing and Financial Systems, relying on SW1's existing Financial package. Mr. Smith had met with two of SW1's executives and was very impressed with their familiarity with Sales and Marketing Systems. It turned out that SW1 had considerable previous experience in developing Marketing systems similar to that required by BBB.

BBB's management informed the three competitors of BBB's choice, and started final negotiations with Colossal Computers.

TABLE V
PROPOSALS EVALUATION—THE FGSM SYSTEM FOR BBB INDUSTRIES

	Colossal	Big Computing	Fast Computing
HARDWARE EVALUATION			
Speed Factor	Average	Average	V. Good
Memory Factor	Average	Low	V. Good
# of installations (Optimia)	200	50	5
Growth Factor	Average	Low	High
PROPOSED SW SOLUTION			
Financial Package	SW1's package	Own Package	To be developed
Marketing System	SW1	Own devlp.	BBB devlp.
SOFTWARE EVALUATION			
Financial Package	Good	Good	?
Marketing Solution	Good	Average	None
Addt'l Packages	Many	A few	None
GENERAL FACTORS			
Familiarity with Equip.	High	Low	Low
Compatibility with BBB's Inventory Sys.	None	None	High
# of SW houses	15	5	2
COMPANY FACTORS			
Company Stability	High	Average	Average
Maintenance Organization	High	Low	Average
Company Commitment	Average	Average	High
ESTIMATED COSTS			
Hardware	$170K	$130K	$140K
Marketing System	$50K	$40K	?
Financial Package	$30K	$30K	$40K
Estimated Modifications to Financial Package	$20K-$40K	$30K-$50K	?
TOTAL COSTS	$270K-$290K	$240K-$260K	$180K+?

The next day, BBB's General Manager got a call from Fast Computing Computers' General Manager, and a meeting was set where BBB was asked to clarify why Colossal was chosen. Fast Computing's General Manager explained that the BBB account had a crucial significance to Fast Computing's future. "If in-house companies (that is—MMM owned) won't buy our equipment, who will? Colossal will use this fact as a weapon to beat us even in places where they don't have such an advantage," he said.

"The solution offered by Colossal answers most of our needs," replied BBB's General Manager, "Your equipment may be good, but you simply do not have enough software packages to attract new clients in our line of business."

The following day, BBB's General Manager got a call from MMM's Chairman: "I would hate to interfere with BBB's internal management, but will you please give Fast Computers another chance? There must be a way for them to get this account."

BBB's General Manager's reply to that was simple: "Only if we can get the same solution as is available on Colossal equipment, within no more than two months delay, and provided that the software is developed by SW1 and that we get all the required modifications to the financial package for free."

When informed by BBB's General Manager of this conversation, Mr. Smith protested: "This is an infeasible solution! It is too expensive for Fast Computing, and I don't believe we will get our system within this time frame."

"Are you sure it cannot be done?" asked BBB's manager.

"Well—It's not impossible, but it sure requires an extraordinary effort," replied Mr. Smith.

"So, we must make sure that Fast Computing does this extraordinary effort."

"If that's what you want, we can put a clause in the agreement that we will not pay unless we get satisfactory results within a predescribed timeframe. However—I still recommend that we take Colossal's proposal," said Mr. Smith.

A couple of days later BBB signed an agreement with Fast Computing Computers. One of the preconditions for payments for both Hardware and Software was that BBB must receive a software solution that satisfied its needs, within the outlined timetable. The total cost of the project to BBB (Hardware, Marketing System, Financial Package and all the required modifications to the Financial Package) was to be $230,000.

F. The Detailed Requirements Specifications for the FGSM System

Fast Computing Computers engaged SW1 Software to develop both the Marketing and the Financial Systems. The Marketing system was to be developed according to BBB's requirements, and the Financial System was to be converted from the Colossal Computer version.

Since the project was to be carried out on Fast computers, SW1 decided not to allocate the same project manager that was proposed to manage the development on Colossal computers (Mr. Brown). A new project manager was recruited to SW1—Mr. Holmes. Mr. Smith was disappointed, since his decision to choose SW1 as software developer was based partly on Mr. Brown's capabilities and familiarity with marketing systems. But, SW1 insisted (they did not want to waste Mr. Brown's familiarity with Colossal equipment).

A Technical Committee was formed: Mr. Smith, Mr. Holmes and Mr. Watson, the representative of Fast Computing Computers. The Committee agreed upon the timetable outlined in Table VI for the development of the FGSM system. It was further agreed that, if feasible, the design and development would be divided into modules (increments), thus enabling starting 1986 with the new inventory system for FGSM (the beginning of the 10th month from the start of the project).

The analysis of FGSM's requirements specifications started off on the right foot. The Specifications Document was ready in time for the Design Review scheduled for month 4. The Design Review lasted two whole days: on top of the technical and supervisory committee members, additional representatives from FGSM's organization participated and contributed their comments and clarifications. However, Mr. Holmes expressed his concern regarding the difficulty in handling the complex form required for the Catalog Number. He complained about the lack of appropriate software tools on Fast Computers: his people were having difficulties in adjusting to the new development environment. They were very hopeful that the new version of operating system, due to be released the next month, would solve these problems. When the discussion narrowed down on the format of the sales reports, it turned out that there was no easy way to develop a report-writer similar to report-writers found in Colossal applications, and SW1 refused to commit to develop a report-writer within the existing budget for the FGSM system. They were willing to commit only to 4 predefined

TABLE VI
ORIGINAL TIMETABLE FOR THE FGSM PROJECT

Months	Subject
1 - 3	Detailed System Requirements Document for FGSM
4	Requirements Review
5 - 6	Detailed Design of FGSM
7 - 9	Programming
10	Acceptance Tests
11-12	New and Old Systems running concurrently

sales reports. Mr. Smith would not agree, and the issue remained unsolved. A similar problem arose regarding the development of special reports to Department-Stores, and this issue remained unsolved as well.

The disagreements were outlined in the document that summarized the Design Review.

G. The Design and Development of the FGSM System

The real problems started at the detailed design phase. SW1's people discovered that the differences between the Fast computer and other computers were more than they had planned for. SW1 did not have people with previous experience in Fast computers, and so the original estimates, that were prepared for the Colossal computer, were not accurate. So as to enable BBB to start 1986 with a new Inventory system, the development was partitioned into 3 increments. The Inventory Module, the Operations Module, the Sales Reports Module. Mr. Holmes presented to Mr. Smith the updated timetable outlined in Table VII.

Mr. Smith pointed out that even though he understood the difficulties SW1 had run into, these problems should be addressed to Fast Computing, and they should be able to help SW1 to keep the original timetables. BBB was willing to accept only one month of delay in the delivery of the total system, and had agreed to break the system into increments so as to receive the first module sooner, not later, than the original timetable. After a couple of meetings between Mr. Smith, Mr. Holmes and Mr. Watson, the parties agreed that it was possible to improve the timetables by 6 weeks, delivering the first module to BBB before the end of the 8th month.

Meanwhile, the people of FGSM were full of enthusiasm towards the prospect of the forthcoming installation. Being aware that once the system was installed, it would be hard to request changes and improvements, they began asking for all sorts of small improvements and minor changes. Both Mr. Holmes and Mr. Smith were very satisfied with the users' attitude, and made every possible effort to please the people of FGSM, by incorporating most of these changes into the design.

H. The Installation of Module #1

Module #1 was installed in the middle of the 9th month—two weeks before the beginning of the New Year. Mr. Holmes, Mr. Smith, and the people of FGSM exerted enormous efforts to have the system up and running in time for the New Year. It turned out, however, that the

TABLE VII
UPDATED TIMETABLE FOR THE INCREMENTAL DEVELOPMENT OF THE FGSM SYSTEM

Months (From beginning of Project)	Subject
5 - 6	Module # 1 - Detailed Design
7 - 9	Module # 1 - Programming and Test
10	Module # 1 - Acceptance Tests
7 - 9	Module # 2 - Detailed Design
10 - 11	Module # 2 - Programming and Test
12	Module # 2 - Acceptance Tests
10	Module # 3 - Detailed Design
11 - 12	Module # 3 - Programming and Test
13	Module # 3 - Acceptance Test

acceptance tests were not comprehensive enough, and after the system was already installed and running, many problems and bugs would still pop up during operations. The many minor design changes that had accumulated in the last 3 months did not help the SW1 programmers to correct these bugs and problems in time, and it was hard to tell which was the latest version of every program. Though the FGSM people were pleased with having an on-line system, they began to feel pretty uneasy about the system when it went through a whole series of corrections, errors, and crashes.

By early 1986, the development of Module #2 was almost complete, but the amount of man-months invested by SW1 had already exceeded the original estimates that were presented to Fast Computing. When SW1's General Manager discussed this problem with Mr. Watson, Mr. Watson explained that there was not much they could do for the time being: Fast Computing still had not received any money from BBB, and its own investments in support and management attention to this project were very high. Mr. Watson's recommendation was to wait for the successful installation of the 2nd and 3rd module before approaching BBB's higher management.

Mr. Holmes discussed these problems with Mr. Smith. Mr. Smith expressed his opinion, that Fast Computing had misled his management into believing that an impossible effort was possible, and that now Fast Computers were not doing their very best to keep their promise. Mr. Holmes remarked that his company did not like to be in such a situation either: lagging behind timetables and exceeding cost estimates. Both felt pretty bitter about the situation they found themselves in. Mr. Holmes, who was not party to the original cost estimates, began to feel that he was going to be blamed for something that was not of his doing, and secretly began looking for another job. One month later Mr. Holmes announced his decision to resign from SW1. One of SW1's senior Systems Analysts who participated in the project was made Project Manager.

I. The Installation of Modules #2 and #3

The installation of Module #2, though two months later than scheduled, was smoother than the installation of Module #1: the acceptance tests were ready, and were carried out properly. However the integration with Module #1 was not an easy task: it was hard to locate the latest

versions of the software that were currently in use. Thus, the installation required a lot of time from SW1 programmers. It became evident that Module #3 would not be ready on time; in fact, the delay was estimated at 6 months.

All the partners to the effort were in bad shape. On one hand, the expenses of SW1 and Fast Computing exceeded even the worst projections, and it was obvious that both companies were going to lose money on this project. On the other hand, BBB was not getting the systems according to the promised timetables, and people started to compare the project to former unsuccessful attempts to introduce new systems to BBB.

The disagreements regarding the contents and form of the Sales Reports now surfaced. FGSM was not willing to settle for the 4 reports suggested by SW1. "The system is completely useless unless we get the reports we want," said Mr. Jones. "Not only that, but the Department Stores are threatening to close their account with us unless we automate the special reports they required, like all their other customers."

SW1 claimed that these reports were not part of their original agreement with Fast Computing. In fact, they blamed the Initial Survey for being vague on these points. "Heaven knows how much money we are going to lose in this project," said their General Manager to Mr. Smith, "Either BBB or Fast Computing must make it up to us."

J. The Financial Systems Design

The problems of the FGSM system were minor relative to the problems that arose during the analysis of BBB's requirements for the Financial Systems. Fast Computing's commitment was to deliver a complete system, tailored to BBB's requirements, and at the price of an "off-the-shelf" product. An initial survey of BBB's requirements, carried out by SW1's professionals, estimated the cost of this project at $150K.

The three General Managers of the three companies were summoned by Mr. Watson to a special meeting. BBB was asked to lower its level of requirements from the Financial System, so as to minimize the projected expenses. BBB's General Manager was furious: "We could have had a working system by now, had we purchased Colossal equipment," he exclaimed. "My people want nothing but the best. It took me a great effort to raise their expectations, and I am not going to let them down. Fast Computers knew exactly what they were up against when they signed the agreement with us. They cannot disregard their commitments now!"

"Our original estimates regarding the scope of the project were based upon the prices quoted by SW1 Software," replied Fast Computing's General Manager "We never intended to make money on this project, but we also never intended to lose that much."

"We based our estimates on BBB's initial survey," retorted SW1's General Manager. "As it turned out, there were too many TBD's, and the problem was that BBB's people wanted the maximum in every case, and would not settle for anything less. They kept coming with more requirements and endless modifications. One of my people has already resigned. We will not take the responsibilities that you two should have taken."

The meeting lasted for four hours, but the parties could not reach an agreement on how to proceed.

V. Case Study Analysis

Clearly, in this case, none of the parties came out a winner. BBB Industries ended up with unsatisfied users, mistrust in information systems, delays, partial systems, low morale, and major unresolved problems. Fast Computing ended up with significant unreimbursed expenditures, a poor reputation in the Sales Information Systems marketplace, and some useless partial products. SW1 also ended up with unreimbursed expenditures, and also a tarnished reputation in Sales Information Systems and poor prospects for future business in the Fast computer user community.

Below is an analysis of how these problems can be traced to lack of responsiveness to the Theory W fundamental principle (*make everyone a winner*) and to the two subsidiary principles (*identify and manage your risks*, and *plan the flight and fly the plan*). The analysis also indicates ways in which the principles could have been used to avoid the problems and to make the participants winners.

A. Make Everyone a Winner

The major source of difficulty was the win–lose contract established between BBB and Fast Computing: no payment unless BBB got everything it asked for, on schedule (Section IV-E). Fast Computing should have made a more thorough analysis of their overrun potential (risk assessment), and a thorough assessment of the benefits of entering the Sales Information System market. If the benefits were high enough, they should have approached MMM's Chairman to authorize their spending additional profit dollars to cover the added costs of software development. Otherwise, they should have dropped out. BBB's General Manager should have heeded Mr. Smith's cautions, and either required a more detailed and realistic plan and cost estimate from Fast Computers, or gone ahead with Colossal. BBB could have made a better win–win situation by not coupling system delivery and cutover to the New Year at a time when the likely development schedules were not well known.

Another major difficulty was SW1's use of Mr. Holmes. If SW1 seriously wanted to penetrate the Fast Computers market, they should have used Mr. Brown (Section IV-F). Holmes should not have accepted responsibility for making people winners until he understood the situation better (Section IV-F). SW1 management should have done more to make Holmes a winner: apprised him of the risks, done a better job of recognizing his good work in getting Module 1 running (Section IV-H), and of monitoring his frustration level and likelihood of leaving SW1 (Section IV-H).

As indicated in Section II, making people winners involves seeking out day-to-day conflicts and changing them into win-win situations. An excellent opportunity to do this occurred at the Design Review (Section IV-F), when SW1 balked at producing more than four sales reports, and at producing any Department Store reports at all. However, the conflict was not addressed, and the project continued to inflate users' expectations without any attempt to get SW1 to provide the promised capabilities.

A Theory W solution to this problem would consider the conditions necessary to make winners of each of the interested parties:

- *BBB and Its Customers:* Furnish the most important reports in the initial delivery, with the other reports as soon as possible thereafter.
- *SW1:* Provide a realistic schedule and budget for producing the desired reports (and other capabilities).
- *Fast Computing:* Develop a strong system with further sales potential, within a realistic and affordable budget and schedule.

Subsequently, a much more thorough analysis would be done to determine realistic budget and schedule estimates as functions of the amount of functionality to be delivered at each increment. These levels of functionality, their associated schedules, and Fast Computing's definition of "affordability" provide some degrees of freedom within which may be possible to define a win-win solution. If so, the project can go forward on such a basis. If not, the project should be disbanded: everyone would not be a winner, but they would minimize their losses.

A similar day-to-day problem which was deferred rather than addressed was the Fast Computing payments problem (Section IV-H). A related problem was the addition of changes and improvements to the system without changing the budget or schedule (Section IV-G). This usually leads to a lose-lose situation when the budget and schedule give out and all the original and new capabilities are not completed. A Theory W solution would involve prioritizing the proposed changes with respect to the original desired capabilities, reallocating the top priority capabilities to remain consistent with the three scheduled increments; then defining an Increment 4 and assuring the users that their remaining features would definitely be incorporated in Increment 4 if BBB's management agreed to provide the budget for them.

Some other problems were created by establishing unrealistic expectations. Issuing vague Requests for Proposal (Section IV-D) is a classical example: users tend to interpret the requirements expansively, while developers interpret them austerely, creating an inevitable lose-lose situation. The cost underestimate and specification interpretation for the Financial System is another example (Section IV-J).

On the other hand, some Theory W principles were followed well. The BBB General Manager's initial conversation with Mr. Smith (Section IV-B) established a realistic climate of expectations. The choice of FGSM as the initial system to implement (Section IV-C) was good, given that FGSM's managers were enthusiastic product champions. Had the other situations been handled in similar ways, with the participants trying harder to accommodate the others' interests, the project could have had a good chance of making the participants winners.

B. Plan the Flight and Fly the Plan

The project's planning was seriously deficient with respect to the elements of a Software Development Plan shown in Fig. 3. Some top-level milestones were established, but no attempt was made to identify dependencies and critical-path items. As discussed in the previous section, the imprecise allocation of responsibilities (e.g., SW1's responsibilities for sales reports) led to serious problems downstream. Several approach and resources problems (configuration management, verification, and validation planning, reviews, resource control) will be discussed further below.

But the major problem here was in putting the plans on a realistic basis. Budgets and schedules were determined more from optimistic target figures than from any rationale based on cost estimation techniques or task dependency analyses. Thus, although more elaborate approach plans would have avoided some problems, they would not have cured the budget-schedule-functionality mismatch problems.

For example, SW1's projected productivity for the Fast Computer development was considered to be equal to their productivity on Colossal Computer projects. Even a rough analysis using the COCOMO cost model [2] indicated a factor of 3 likely reduction in productivity due to personnel capability and experience, support system volatility, reduced tool support, and schedule compression.

1) Configuration Management: In this area, we can easily count the following shortcomings from the part of the project management:

- No change control system.
- No configuration management and control.
- No baselined master version of the specs or programs.
- No quality assurance (project standards, technical audits).

All those led to confusion, multiple bugs, problems in integration, installation, unmaintainability of the system, additional costs, and errors. There was no controlled mechanism for product changes, no track of product status, and no product integrity.

2) Verification and Validation Planning:

Most of the basic principles of V&V planning were not implemented in this case:

- No verification of the initial survey or the detailed design.
- Insufficient, late test plans (due to untimely, careless preparation).
- No acceptance criteria.
- No integration and test plans.
- Test phase and system acceptance combined.

As a result, the users got their system before it was completely verified, and were confronted with bugs and

problems. The system's reliability was undermined, and the operations forced into a haphazard process.

3) Review Plans: No product review was held, only a requirements review. However, the problems that arose in the review were not assigned, nor tracked. It is no wonder that most problems were left unattended. The results were that on one hand there were missing capabilities, and on the other that some of the requirements were not really needed. The users were not committed to the final product. Attempts to correct the problems of missing capabilities at later stages were very expensive. A proper treatment of the problem at an earlier stage would have been less costly.

4) Resources, Status Monitoring, and Control: The main problems in the area were:

- Only high-level milestone charts were available.
- No work breakdown structure was prepared.
- No budget allocations were established.

Therefore, no cost versus progress monitoring and control was possible, and only when the overall budget was exceeded were the problems surfaced. Problems of insufficient personnel and inappropriate budget were discovered only when it was too late. In short, the visibility was poor, both at the overall progress level and the individual trouble-spot level.

C. Identify and Manage Your Risks

In some cases, the participants did a good job of identifying and managing risks. In particular, Mr. Smith's recommendation in Section IV-C to start and pursue an incremental development was very good. But there were many situations in which the lack of risk management caused serious problems.

Allowing two weeks to prepare the RFP (Section IV-D) reflects a serious neglect of risk management. BBB's General Manager should have done a risk analysis on hearing Mr. Smith assess Fast Computing's need for "extraordinary effort" to succeed (Section IV-E); in particular, to carry out an independent estimate of the development cost and schedule.

BBB also did not risk assessment by looking behind the interface between Fast Computing and SW1. They did not investigate whether SW1 would use Mr. Brown on their job, and were taken by surprise when SW1 assigned the unknown Mr. Holmes. Holmes himself did very little analysis of the risks he was getting into.

BBB did not assess the risk of the highly optimistic, highly overlapped incremental development schedule proposed by SW1 (Table V, Section IV-G). They were too preoccupied with establishing an ambitious schedule for Increment 1 to meet their New Year deadline. Such overlapping increments are major sources of risk, as changes in the earlier increments usually have serious ripple effects on the later increments under development.

In one case, risk avoidance caused an "everyone a winner" problem. Mr. Smith identified several risks due to lack of user management commitment, and addressed these by a strong effort to sell the users on the advantages of information technology. This backfired when the users compared their unrealistic expectations to the project's results. A preferred Theory W solution would be to couch user benefit projections more realistically in terms of expected near-term and long-term benefits, and to involve the users more closely in analyzing and preparing for the benefits.

VI. Conclusions

When applied to a project case study, a good management theory should be able to do two things:

1)To explain why the project encountered problems.

2) To prescribe improved approaches which would have avoided the problems.

Analysis of the BBB case study indicates that the Theory W fundamental principle (*Make everyone a winner*) and its two subsidiary principles (*Plan the flight and fly the plan; identify and manage your risks*) did a good job on both counts. The case study and the other examples provided earlier also indicate that Theory W does a reasonably good job in satisfying the management theory objectives of being simultaneously simple, general, and specific.

References

[1] K. Blanchard and S. Johnson, *The One Minute Manager*. Berkeley, CA: Berkeley Books, 1982.

[2] B. W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[3] ——, "Seven basic principles of software engineering," *J. Syst. Software*, vol. 3, pp. 3-24, 1983.

[4] ——, "A spiral model of software development and enhancement," *Computer*, pp. 61-72, May 1988.

[5] ——, "Tutorial volume: Software risk management," IEEE Computer Society, 1989 (in publication).

[6] J. D. Couger and R. A. Zawacki, *Motivating and Managing Computer Personnel*. New York: Wiley, 1980.

[7] W. G. Dyer, *Team Building*. Reading, MA: Addison-Wesley, 1987.

[8] M. W. Evans, P. Piazza, and B. Dolkas, *Principles of Productive Software Management*. New York: Wiley, 1983.

[9] R. Fisher and W. Ury, *Getting to Yes*. Boston, MA: Houghton-Mifflin, 1981; also, Baltimore, MD: Penguin Books, 1983.

[10] S. W. Gellerman, *Motivation and Productivity*. New York: American Books, 1978.

[11] N. J. Kirchof and J. R. Adams, *Conflict Management for Project Managers*, Project Management Inst., Feb. 1986.

[12] H. Koontz and C. O'Donnell, *Principles of Management: An Analysis of Managerial Functions*, 5th ed. New York: McGraw-Hill, 1972.

[13] D. McGregor, *The Human Side of Enterprise*. New York: McGraw-Hill, 1960.

[14] P. W. Metzger, *Managing a Programming Project*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[15] G. I. Nierenberg, *The Art of Negotiating*, Pocket Books, 1984.

[16] W. G. Ouchi, *Theory Z*. Reading, MA: Addison-Wesley, 1981; also Avon, 1982.

[17] T. J. Peters and R. H. Waterman, *In Search of Excellence*. New York: Harper & Row, 1982.

[18] D. J. Reifer, *Tutorial: Software Management*, 3rd ed., IEEE Catalog No. EHO 189-1, 1986.

[19] J. S. Rosenschein and M. R. Genesereth, "Deals among rational agents," in *Proc. IJCAI-85*, pp. 91-99.

[20] F. W. Taylor, *The Principles of Scientific Management*. New York: Harper and Brothers, 1911.

[21] R. H. Thayer, A. Pyster, and R. C. Wood, "The challenge of software engineering project management," *Computer*, pp. 51-59, Aug. 1980.

[22] R. E. Walton, *Managing Conflict*. Reading, MA: Addison-Wesley, 1987.

Barry W. Boehm (SM'84) received the B.A. degree in mathematics from Harvard University, Cambridge, MA, in 1957, and the M.A. and Ph.D. degrees in mathematics from the University of California, Los Angeles, in 1961 and 1964, respectively.

He is an Adjunct Professor of Computer Science at UCLA, and also Chief Scientist of TRW's Defense Systems Group. He has served as the manager of several TRW software development projects, and also directed the development of TRW's software management policies and standards.

Within the IEEE Computer Society, Dr. Boehm has served as Chairman of the Technical Committee on Software Engineering, a member of the Governing Board, and an editorial board member of *Computer*, *IEEE Software*, and IEEE TRANSACTIONS ON SOFTWARE ENGINEERING.

Rony Ross was born in Tel Aviv. She received the B.Sc. degree in mathematics from Tel Aviv University in 1972, the M.Sc. degree in computer science from the Weizmann Institute of Science, Rehovot, Israel, in 1976, and the M.B.A. degree in business administration from the Graduate School of Management, Tel Aviv University, in 1980.

From 1971 to 1980, she was a Teaching Assistant at Tel Aviv University Department of Computer Science. From 1975 to 1980, she was employed by Mini-Systems Computers Ltd. as a Systems Analyst and Real-Time Programmer, working for Sci-Tex Corp. From 1980 to 1983, she was Manager of Data Processing and Information Systems Department of Kitan Ltd. From 1983 to 1986, she was in charge of new business development for Contahal Ltd. Currently, she is studying toward the Ph.D. degree in the Department of Computer Science at the University of California, Los Angeles.