WHAT DO YOU MEAN YOU CAN'T TELL ME IF MY PROJECT IS IN TROUBLE?

by

Joseph Kasser, DSc. , CM., CEng.
University of Maryland University College
University Boulevard at Adelphi Rd.
College Park, MD 20742-1614
Phone 301-985-4616, fax 301-985-4611
E-mail: jkasser@polaris.umuc.edu

Victoria R. Williams
Keane Federal Systems, Inc.
1375 Piccard Drive, Suite 200
Rockville, MD 20850
Phone 301-548-4450, fax 301-548-1047
E-mail: vwilliam@keane.com, vrwcbw@erols.com

ABSTRACT

All the measurements made during the Software Development Life Cycle (SDLC) do not provide an accurate answer to the question. Anecdotal evidence suggests that most projects do not fail due to the non mitigation of technical risks. Rather, they fail as a result of poor management of the human element. This paper describes the development of a set of risk-indicators based on the human element. These risk-indicators can be further refined into metrics to provide an answer to the question posed in the title of the paper.

INTRODUCTION

The SDLC for large systems can take several years to complete. During this time, the:

- $ customer makes periodic progress payments to the supplier (contractor). In this situation, since the acceptance tests are often only made at the end of the SDLC, the suitability of the product for its mission is unknown for the time in which the bulk of the payments are made.
- $ supplier provides the customer with information to reassure the customer that the product is being produced on schedule and within budget. The information is provided in the form of reports and other documents, detailing budget compliance (estimated and actual), GANTT and PERT Charts, conformance to Abest practices,@ estimated function points, lines of code, defects, number of requirements satisfied, degree of compliance to the Software Capability Maturity Model (CMM) and to the International Organization of Standards (ISO) models, etc.

The reports containing this intermediate information are produced to demonstrate a low risk of non delivery and non compliance to the requirements. (Drucker, 1973, 509) wrote that throughout management science, in the literature as well as in the work in progress, the emphasis is on techniques rather than on principles, on mechanics rather than on decisions, on tools rather than on results, and, above all, on the efficiency of the part rather than on the performance of the

whole. Nothing seems to have changed in 25 years. While the SDLC has evolved from the waterfall method through various iterative approaches (e.g., Incremental, Spiral, and Rapid Prototyping), the focus of measurements being made in today's paradigm are still in the process and product dimensions of the activities (Kasser, 1997). These measurements provide post facto information, namely they report on what has already happened. This causes management to be reactive instead of being proactive. In addition, in spite of all the measurements being made, the supplier is often unable to tell the customer:

- the exact percentage of completeness of the system under construction anytime during the SDLC, and;
- the probability of successful completion within budget and according to schedule.

Thus, there is little wonder that software projects tend to fail (exceed original estimates for cost and schedule, or terminate prematurely). For example, in the United States alone, in 1995:

- the Standish Group estimated that companies and government spent $81 billion for canceled software projects (Chaos, 1995).
- almost 80,000 software projects were canceled (Voyages, 1996).
- approximately 80% of the medium to large software projects executed within the Department of Defense (DoD) were 100% over budget and 90% were at least one year behind schedule (Cuppan, 1995).

The growing international dependency on the ISO standards for the SDLC indicates that this phenomenon of software project failure is not limited to the United States. Anecdotal evidence suggests that most projects do not fail due to technical reasons. Rather, the failure tends to be due to the human element. In addition, while the Standish Group identified ten major causes for project failure along with their solutions, they also stated that it was unclear if those solutions could be implemented (Voyages, 1966). This paper describes the development of several indicators that can be used to identify metrics to predict that a project is at risk of failure.

A METHODOLOGY FOR DEVELOPING METRICS FOR PREDICTING RISKS OF PROJECT FAILURES

The methodology is based on Case Studies written by students in the Graduate School of Management and Technology at the University of Maryland University College[1]. These students wrote and presented term papers describing their experiences in projects that were in trouble. The papers adhered to the following instructions:

- document a Case Study. Students had to write a scenario for the paper based on personal experience.
- analyze the scenario.
- document the reasons the project succeeded or ran into trouble.

[1] These students are employed in the workforce and are working towards their degree in the evening. Their employment positions range from programmers to project managers. Some also have up to 20 years of experience in their respective fields.

- list and comment on the lessons learned from the analysis.
- identify a better way with 20/20 hindsight.
- list a **number of situational indicators** that can be used to **identify a project in trouble** or a successful project **while the project is in progress**.

The methodology:

- summarized the student papers to identify common elements.
- surveyed systems and software development personnel via the Internet to determine if they agreed or disagreed with the indicators.
- summarized and analyzed the results.

Summary of Student Papers

Nineteen students produced papers that identified 34 different indicators. Each indicator identified was a risk or a symptom of a risk that can lead to project failure. Several indicators showed up in more than one student paper; *Apoor requirements@* showed up in all of the papers.

The Survey

A survey questionnaire was constructed based on the student provided risk-indicators[2] and sent to systems and software development personnel via the Internet. The survey asked respondents to state if they agreed or disagreed that the student provided indicators were causes of project failure.[3] One hundred and forty-eight responses were received.

The findings are summarized in Table 1. The first column contains a number identifying the risk-indicator described in the second column. The third column lists the number of students that identified the risk. The fourth column contains the percentage of agreement. The fifth column contains the percentage of disagreement. The sixth column is the ranking of the risk-indicator.

Survey Results

The survey results were surprising. Modern Total Quality Management (TQM) theory holds that the Quality Assurance Department is not responsible for the quality of the software. Everybody shares that responsibility. Thus, while it was expected that most respondents would disagree with this risk-indicator, only 60% of the respondents disagreed. It was also anticipated that most respondents would agree with the other risk-indicators, yet the overall degree of agreement was:

0.7% (one respondent) agreed with all 34 risk-indicators.
8.1% agreed with at least 30 risk-indicators.

---

[2] The students tended to state ›problems using the semantics of ›solutions or ›symptoms rather than ›causes .

[3] The authors recognized that there are other causes of (risks) project failure and added an Aother@ category to the survey questionnaire for Awrite-in@ risks.

51% agreed with at least 20 risk-indicators.
93% agreed with at least 10 risk-indicators.

As for the degree of disagreement:

0.7% (one respondent) disagreed with 25 risk-indicators.
4.7% disagreed with at least 20 risk-indicators.

Table 1 Initial Findings

Risk	Risk-Indicators	Students	Survey agree	Survey disagree	Rank
1	Poor requirements	19	97	3	1
2	Failure to use experienced people	7	79	21	13
3	Failure to use Independent Verification and Validation (IV&V) [Note 1]	6	38	62	31
4	Lack of process and standards	5	84	16	11
5	Lack of, or, poor plans	4	95	5	2
6	Failure to validate original specification and requirements	3	91	9	3
7	Lack of Configuration Management	3	66	34	19
8	Low morale	2	51	49	24
9	Management does not understand SDLC	2	59	41	22
10	Management that does not understand technical issues	2	56	44	23
11	No single person accountable/responsible for project	2	69	31	18
12	Client and development staff fail to attend scheduled meetings	1	42	58	28
13	Coding from high level requirements without design	1	75	25	14
14	Documentation is not produced	1	63	38	21
15	Failure to collect performance & process metrics and report them to management	1	48	52	25
16	Failure to communicate with the customer	1	88	12	5
17	Failure to consider existing relationships when replacing systems	1	85	15	10
18	Failure to reuse code	1	27	73	34
19	Failure to stress test the software	1	75	25	15
20	Failure to use problem language	1	34	66	30
21	High staff turnover	1	71	29	16
22	Key activities are discontinued	1	74	26	17
23	Lack of Requirements Traceability Matrix	1	67	33	19
24	Lack of clearly defined organizational (responsibility and accountability) structure	1	82	18	11
25	Lack of management support	1	87	13	6
26	Lack of priorities	1	85	15	8
27	Lack of understanding that demo software is only good for demos	1	47	53	26
28	Management expects a CASE Tool to be a silver bullet	1	45	55	27
29	Political considerations outweigh technical factors	1	86	14	9
30	Resources are not allocated well	1	92	8	4
31	The Quality Assurance Team is not responsible for the quality of the software	1	40	60	29

Risk	Risk-Indicators	Students	Survey agree	Survey disagree	Rank
32	There are too many people working on the project	1	36	64	32
33	Unrealistic deadlines - hence schedule slips	1	86	14	7
34	Hostility between developer and IV&V	1	33	67	33

Note 1 The papers were written for a class on IV&V, hence the emphasis on IV&V. However, if the descriptions of tasks that IV&V should have performed (in the papers) are examined, the word AIV&V@ could easily be replaced with the word Asystems engineering,@ and the papers would be equally valid.

52% disagreed with at least 10 risk-indicators.
88% disagreed with at least one risk-indicator.

Further Analysis

The top seven (high priority) risk-indicators were identified using the following approaches:

$ **The Tally:** An Aagree@ was allocated a value of +1, a Adisagree@ a value of -1. The answers to each survey statement were then tallied. The raw results are shown in Table 1. The following risk-indicators received the highest positive values (most agreement) as causes of project failure:

Risk	Risk-indicator	Responses
1	Poor requirements	134
5	Lack of, or, poor plans	125
6	Failure to validate original specification and requirements	113
30	Resources are not allocated well	109
16	Failure to communicate with the customer	106
25	Lack of management support	98
33	Unrealistic deadlines - hence schedule slips	97

$ **Priorities:** The survey asked respondents to rank the risk-indicators in order of priority. The weighted results are as follows (top priority first):

Risk	Risk-indicator	Weight
1	Poor requirements	864
16	Failure to communicate with the customer	683
5	Lack of, or, poor plans	574
4	Lack of process and standards	361
25	Lack of management support	350
6	Failure to validate original specification and requirements	329
29	Political considerations outweigh technical factors	304

$ **Top Seven List:** Since the actual position may be subjective, the number of times a risk-indicator showed up in the priority list was also tallied. The results are as follows:

Risk	Risk-indicator	Count
1	Poor requirements	99

Risk	Risk-indicator	Count
16	Failure to communicate with the customer	86
5	Lack of, or, poor plans	77
4	Lack of process and standards	51
25	Lack of management support	51
29	Political considerations outweigh technical factors	45
6	Failure to validate original specification and requirements	44

These results show a high degree of consensus on these risk-indicators as causes of project failures.

Sensitivity Analysis

The sample size for respondents without management experience was 99. The raw tallies for the risk-indicators listed below were examined to see if there was a difference between non managers and managers with various years of experience. No differences of more than 10% were noted.

Risk	Risk-indicator
5	Lack of, or, poor plans
8	Low morale
15	Failure to collect performance & process metrics and report them to management
25	Lack of management support
27	Lack of understanding that demo software is only good for demos
29	Political considerations outweigh technical factors
32	There are too many people working on the project
33	Unrealistic deadlines - hence schedule slips

The AOther@ Category

Several respondents added a small number of risk-indicators in the Aother@ category of the questionnaire. These included:

- *failure to control change.*
- *rapid rate of change of technology.*
- *low bidding.*
- *poor management.*
- *lack of a technical leader.*

Thus, the small student sample size of 19 seems to have identified most of the important risk-indicators.

The Risk-Indicators Most People Disagreed With

Part of the analysis of the survey results was to determine which risk-indicators received the most amounts of disagreement as causes of project failure. This was done by determining the:

- largest number of disagreements by the recipients.

$ least number of agreements by the recipients.

The risk-indicators receiving the largest number of disagreements were:

Risk	Risk-indicator	Responses
18	Failure to reuse code	88
3	Failure to use Independent Verification and Validation (IV&V)	80
32	There are too many people working on the project	75
12	Client and development staff fail to attend scheduled meetings	74
34	Hostility between developer and IV&V	70
31	The Quality Assurance Team is not responsible for the quality of the software	68
15	Failure to collect performance & process metrics and report them to management	67

The following risk-indicators received the least number of agreements as causes of project failure:

Risk	Risk-indicator	Responses
20	Failure to use problem language	30
18	Failure to reuse code	32
34	Hostility between developer and IV&V	34
32	There are too many people working on the project	43
31	The Quality Assurance Team is not responsible for the quality of the software	45
3	Failure to use Independent Verification and Validation (IV&V)	49
28	Management expects a CASE Tool to be a silver bullet	53
12	Client and development staff fail to attend scheduled meetings	54
27	Lack of understanding that demo software is only good for demos	55

In each method of analysis, six risk-indicators showed up in the group receiving the most amount of disagreement. Consider the risk-indicators most of the respondents disagreed with, namely:

$ *Failure to reuse code:* A major advantage of Object Oriented Technology is the ability to lower costs by reusing code. Yet 73% of those surveyed did not agree with this risk-indicator.
$ *Hostility between developer and IV&V:* This risk-indicator shows a team problem and results in less than optimal costs due to the lack of cooperation.
$ *There are too many people working on the project:* This risk-indicator is based on (Brooks, 1982) which describes the problems associated with assigning additional people to projects.
$ *Failure to use problem language:* The use of problem language was promoted as one of the major advantages by (Ward & Mellor, 1985). Yet, only 34% of the respondents agreed that it was a risk. Several did not know what the term meant.
$ *The Quality Assurance Team is not responsible for the quality of the software:* As discussed above, this was the only indicator that should have shown disagreement.
$ *Client and development staff fail to attend scheduled meetings:* This is a symptom of poor communication between the client and the developer. In addition, while there are other communication techniques available, if meetings are scheduled, and not attended, negative messages are sent to the project personnel.
$ *Failure to collect performance & process metrics and report them to management:* If measurements are not made and acted upon, how can the process be improved? Yet 52% of the respondents disagreed that this was a risk-indicator.

THE CHAOS STUDY

The (Chaos, 1995) study served as a reference. It had identified some major reasons for project failure. The five risk-indicators in this study that were chosen as the most important causes for project failure also appear on the Chaos list of major reasons for project failure. The correlation between this study and the Chaos study is shown below. While *Aresources are not allocated well@* did not show up in the top seven lists of this study, it was fourth in the tally. Thus, this study supports the findings of the Chaos study.

Risk	This study	Chaos Study
1	Poor requirements	Incomplete requirements
16	Failure to communicate with the customer	Lack of user involvement
30	Resources are not allocated well	Lack of resources
	no equivalent	Unrealistic expectations
25	Lack of management support	Lack of executive management support
	no equivalent	Changing requirements and specifications
5	Lack of, or, poor plans	Lack of planning

PRESENCE OF RISK-INDICATORS IN ISO 9001 AND THE SOFTWARE-CMM

The elements of Section 4 of the ISO 9001 Standard and the five levels of the Software-CMM (CMM, 1995) were examined and interpreted to determine if the major student risk-indicators were covered in the ISO Standard and in the Software-CMM. The ISO 9001 Standard defines the minimum requirements for a quality system, while the Software-CMM tends to address the issues of continuous process improvement more explicitly than does the ISO 9001 Standard. The findings are shown below where an ›x represents the presence of the indicator. The same two major risk-indicators could not be mapped into either the elements of Section 4 of the ISO Standard, or the Software-CMM, namely:

$ *Political considerations outweigh technical factors.*
$ *Unrealistic deadlines - hence schedule slips.*

Thus, conformance to either or both Quality standards does not ensure mitigating these risks.

Risk	Section 4.x of ISO 9001																				Software-CMM				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5
1			x	x	x	x		x					x									x	x		
4		x			x			x	x	x						x	x		x	x		x	x	x	
5		x			x																	x	x		
6			x	x		x		x		x	x				x		x		x			x	x		
16			x		x									x		x						x	x		
25	X																						x		
29																									
30	X								x		x			x			x	x	x					x	
33																									

THE DEVELOPMENT OF METRICS

TO IDENTIFY THE PRESENCE OF THESE INDICATORS

The large consensus on the major reasons for project failure seems to show that if we could remove these reasons, projects would have a greater probability of success. The (Chaos, 1995) study showed that projects tended to succeed if the opposite of these risk-indicators were present (e.g., good requirements instead of poor requirements). The (Voyages, 1996) paper stated that the causes were known, but it was unclear if the solutions could be implemented. Thus it seems that the current metrics paradigm is focused on measuring the wrong things and needs to be changed to develop metrics to show the presence or absence of the major risk-indicators identified above and any other known as major causes of project failures (risk management). Consider ways metrics can be developed for the following risk-indicators:

- *Poor requirements:* Current requirements management tools do not indicate if requirements are defective. That is, if they violate the requirements for writing requirements (Kasser, 1995, 166). This risk can be mitigated by teaching personnel how to write good requirements using techniques such as a ›requirements workshop (Kasser, 1995, 259). Metrics can then be developed to measure the number of defects in requirements documents.
- *Failure to communicate with the customer:* This activity mitigates the risk of creating poor requirements and failing to cope with changes. This activity can be part of the process and is covered in ISO 9001.
- *Lack of, or, poor plans:* A project plan was recognized as a critical document as far back as the U.S. Military Standard for Systems Engineering Management (MIL-STD-499A), yet many software projects still either do not have one, or ignore it. Since it must be a living document (Kasser, 1995, 163), metrics need to be developed to monitor the use/updates of the plan during the SDLC.
- *Lack of process and standards:* These are covered by the Software-CMM the ISO Standards covering the SDLC.
- *Lack of management support:* This risk is recognized as the prime reason that TQM fails. We need to develop metrics to measure the degree of visible management support.
- *Failure to validate original specification and requirements:* Having a process and including this activity as a step in the process can mitigate this risk.
- *Political considerations outweigh technical factors:* Metrics need to be developed for this risk.
- *Resources are not allocated well:* This risk follows on from the lack of, and use of a plan. Having and using a plan will mitigate this risk.
- *Changing requirements and specifications:* (Kasser, 1997) presented a simple Categorized Requirements in Process (CRIP) method for monitoring progress and making both progress and changes in requirements readily visible to top management and any other interested parties. The CRIP approach can easily be built into requirements management tools.

However, just changing the metrics paradigm may not be the complete solution. Cobb s Paradox (Voyages, 1996) states *AWe know why projects fail, we know how to prevent their failure B so why do they still fail?@* Now **a paradox is a symptom of a flaw in the underlying paradigm.** Perhaps Juran and Deming provided the remedy. Juran as quoted by (Harrington, 1995, 198) stated that management causes 80 to 85% of all organizational problems. (Deming, 1993, 35) stated that 94% of the problems belong to the system (i.e., were the responsibility of

management). In this survey, both managers and non managers tended to disagree with the two management risk-indicators (#9 and #10). Several respondents with many years of systems or software engineering experience did not even recognize the term ASDLC.@ **It is difficult to understand how Information Technology managers can make informed decisions to mitigate technical risks if they don t understand the implications of their decisions.** The resolution of Cobb's paradox may be to develop a new paradigm for an information age organization which performs the functions of management without managers.

DEFICIENCIES IN THE STUDY

The following deficiencies are present in the study:

- the sample size is small.
- the level of expertise of the respondents is unknown.

CONCLUSIONS AND RECOMMENDATIONS

Except for *poor requirements,* none of the risk-indicators identified by this study are technical. Thus, the findings support:

- resources spent mitigating technical risks are wasted unless the major risks discussed in this paper are also mitigated. Thus, it is critical to develop and use good metrics for them.
- the need for continual training to provide managers with the skills to become capable of effective technical management.

Areas For Further Study

- which risk-indicators have greater effect on the schedule and budget.
- correlate the general agreement with the risk-indicators to specific ones. For example, did those who agreed with A*political considerations outweigh technical factors*@ agree with more risk-indicators than those who didn t agree with it.

REFERENCES

Brooks, F.P., The Mythical Man-Month Essays on Software Engineering, Addison-Wesley Publishing Company, Reprinted with corrections, 1982.

Carnegie Mellon University, The Capability Maturity Model: Guidelines for Improving The Software Process, Addison-Wesley, 1995.

Cuppan, C.D., *Capability Maturity Model (CMM) Characteristics and Benefits*, tutorial presentation at the Defense Mapping Agency, 2 June 1995.

Deming, W.E., *The New Economics for Industry, Government and Education*, MIT Center for Advanced Engineering Study, 1993.

Drucker, P.F., *Management: Tasks, Responsibilities, Practices*, New York, Harper & Roe, 1973.

Harrington, H.J., *Total Improvement Management the next generation in performance improvement*, McGraw-Hill, 1995.

Kasser, J.E., Applying Total Quality Management to Systems Engineering, *Artech House,*1995.

Kasser, J.E., AWhat Do You Mean, You Can t Tell Me How Much of My Project Has Been Completed?@, *The International Council on Systems Engineering (INCOSE) 7th International Symposium*, Los Angeles, CA., 1997.

The Standish Group, *Chaos*, **http://www.standishgroup.com/chaos.html,** accessed March 19, 1998.

The Standish Group, *Unfinished Voyages*, **http://www.standishgroup.com/voyages.html,** accessed March 19, 1998.

Ward, P.T., Mellor, S.J., *Structured Development for Real-Time Systems*, Yourdon Press Computing Series, 1985.

AUTHORS

Dr. Kasser has more than 25 years of award winning experience in management and engineering. He teaches software IV&V and software maintenance at the University of Maryland University College. He is a recipient of NASA s Manned Space Flight Awareness (Silver Snoopy) Award for quality and technical excellence. He is a Certified Manager and a recipient of the Institute of Certified Professional Manager's 1993 Distinguished Service Award. He is the author of *Applying Total Quality Management to Systems Engineering* published by Artech House and more than 30 journal articles and conference papers. His current interests lie in the areas of applying systems engineering to organizations and using technology to improve the practice of management.

Victoria R. Williams is a Senior Consultant at Keane Federal Systems, Inc. in Rockville, MD. She has 18 years of award winning experience in various aspects of systems engineering. She has received many awards and commendations from employers and customers including Computer Data Systems Inc., the Department of the Navy, Naval Air Systems Command, and the Department of the Army. She has written software in HTML, PowerBuilder, Java, FoxPro and Visual Basic. She has performed object-oriented analysis and design in an effort to reengineer an operational legacy system. Her previous experience includes project management support for the Department of the Navy, systems and software engineering for the U.S. Navy, the Royal Australian Navy, and the U.S. Military Sealift Command. She is also trained at CMM Level 2 and is currently working on her Master of Science Degree in Computer Systems Management in the Graduate School of Management and Technology at the University of Maryland University College in College Park, Maryland.