RMRP–2002-01, Version 1.0

Universal Risk Project

Final Report

February 2002

Risk Management Research and Development Program Collaboration

[Formal Collaboration:
INCOSE Risk Management Working Group; Project Management Institute Risk Management Specific Interest Group]

Table of Contents

Universal Risk Project
Final Report

"If you can't afford to mitigate the risk now, be absolutely sure you can afford to resolve the problem later when it happens."

Project Leaders:
INCOSE RMWG – Dave Hall, SRS Information Services,dhall@pop300.gsfc.nasa.gov
PMI RiskSIG - Dr. David Hulett, Hulett and Associates, dthulett@lainet.com

Major Project Contributors:
(1) Roger Graves, Davion Systems, Ltd. Rgraves@davion.com
(2) Wes Calhoun, Systems Engineering Manager, Raytheon
(3) Steve Waddell, Newport News Shipbuilding (waddell_js@naptheon.com)
(4) Barney B. Roberts, Futron Corporation (broberts@futron.com)
(5) John A. Cloward, (john_a_cloward@groton.pfizer.com)
(6) Richard W. Kitterman [r.w.kitterman@larc.nasa.gov]
(7) Tom Davis, AMC/SCT [Tom.Davis@scott.af.mil]
(8) Judy A. Lewis, (judy.lewis@eds.com]
(9) Ron Kohl, NASA OMNIBUS Chief Systems Engineer, Titan Systems Co
(10) Blaise Burgman [bburgman@acm.org]
(11) Ralph Simon, [ralph.simon@lmco.com]
(12) David Whitmoyer, [dave.whitmoyer@cotelligent.com]
(13) Mary Dadone Irish, (DRMMDI@aol.com)
(14) Noel Dickover, Department of the Navy, OASN(RD&A)Acquisition Reform Office, (NDickover@ar.navy.mil)
(15) David Hillson, [dhillson@pmprofessional.com]
(16) Paul Callender, Partner, CMRS-Group [paulcallender@hotmail.com]
(17) Rick Dondo, Gestolande Inc. [Rick_Dondo@videotron.ca]
(18) Robert D. Jones, Technical, Education and Management Services, Inc (RJonesAssn@aol.com)
(19) Barbaro V. Moya, Climate and Air Pollution (meteoro@atenas.inf.cu)

Abstract

The Risk Management Research and Development Program is a multi-association program designed to accomplish the following: ***Research, develop and promulgate suggested Project[1] Risk Management practices, tools and knowledge to enable and accelerate the transition of Project Risk Management from an art into a science.*** Within the overarching Program, there are several projects covering each of the major Risk Management Process areas[2]. The Universal Risk Project is designed to develop a list and definitions of "universal risk areas" that can assist risk management in any type of project in any industrial/government/commercial sector. This list of risk areas is intended to be illustrative and serve only as an aid or a guide to those starting the risk identification process. The list is NOT intended to serve as a complete, comprehensive checklist of all risk areas or risks applicable to any specific project. It should add to, but not substitute for, an organization's own list of typical risks. During this project, the collaborators had to accomplish several specific actions. These were to define what is meant by "universal risk area", develop a list of risk areas, develop a list of risks under each risk area and define each risk to include examples from several sectors. This is the Final Report for the Universal Risk Project.

[1] For this report, the term Project is defined to encompass projects to develop specific product(s), operation of a system or systems, or conducting an analysis, etc. Essentially any endeavor being carried out to accomplish a specific purpose.

[2] Each organization uses slightly different terms to describe the Risk management process. This report will concentrate on the INCOSE process terms - Planning, Assessment, Handling and Monitoring. The PMI terms for the same steps are Risk Management Planning, Risk Identification, Qualitative and Quantitative Risk Analysis, Risk Response Planning and Risk Monitoring and Control.

Section 1.0 Introduction

1.1 What is Project Risk?

Project Risk (for the remainder of this report, we will use the term "risk" only) can be very subjective depending upon who looks at it. What consequences there are changes for each specific stakeholder. A fairly common definition of Risk is as follows: *An uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives.*[3] However, there are three characteristics commonly found in all definitions of risk:

- A risk is a **future** event that may or may not occur. We sometimes categorize past events as causing or being problems, issues, or maybe a crisis. Since these events were in the past, they are not examples of risk. In addition to taking place in the future, a risk must also be an event. Therefore, terms like cost, schedule and performance do not meet the definition of risk we use because they are not events. However, a future event in the program schedule or a future event that measures system performance could be a risk. Also, neither a person nor a place is an event, so neither meets our definition of risk. However, an event that a person executes or an event that occurs at a particular place could be a risk. When considering risk, ensure that you are considering a future event.

- The **probability** of the future event occurring must be greater than 0% but less than 100%. There are countless future events – but not all of them are risks. Future events that have a zero or 100 % chance of occurrence are not risks. Once we know with 100% certainty that a future event will occur, we have a problem, a crisis or an issue, not a risk. When considering risk, remember that future events are risk candidates if the probability of occurrence lies between 0% and 100%.

- The **consequence** of the future event must be unexpected or unplanned for. There are many future events that could possibly occur (with a probability between zero and 100 percent), with unexpected consequences. Some of those future events can be categorized as opportunity (positive risk) if their consequences are favorable. Others can be categorized as negative risks if their consequences are unfavorable. Once we determine an event is in the future and it has a probability of occurrence between 0% and 100%, we then look to determine the probable consequence of the event

.

1.2 Why Risk Management?

A few reasons why effective Risk Management is both important and beneficial include:

• Environment - A changing, more constrained project environment including downsizing, tighter budgets, and the desire to develop and deliver systems faster. Thus, the development and delivery of many systems is now higher risk than the historical

[3] *PMBOK® 2000 Guide*, p. 207. Other definitions can be found in "What Is Risk?", a Risk Management Research Program paper by Dave Hall.

norm for the same level of relative technology gain. For operation of systems, the requirement to accomplish the same (or more) level of operations with fewer personnel and less maintenance means that these operations are now higher risk than the historical norm.

• Added Value - A single averted risk on a project or operation can pay for many, if not all, Risk Management activities. Recent examples exist where customers have used the contractor’s proposed Risk Management process and evaluation of candidate risks as a key source selection discriminator in competitive procurements; in one case the procurement totaled several billion dollars and Risk Management and a related process were the sole source selection discriminators. Thus, if performed effectively, Risk Management may not only help a project avert potential performance, cost, and schedule impacts, but it may also help win a contract.

• Forward-Looking - Risk Management provides leverage on the front-end of a project and helps to avoid costly performance, cost, and schedule problems downstream.

• Formal Methodology - Risk Management is a structured tool for day-to-day decision-making. As one project manager said: ”It helps me in my daily decision making."

• Formal Procedures - When unanticipated problems occur (such as external events that were not foreseeable), Risk Management can help you keep on top of problem issues and efficiently contribute to their solution.

Risk Management provides a greater opportunity to enable relatively high-risk acquisition approaches (such as NASA’s faster, cheaper, better) to be successful. The ultimate success of a project within the ever-tightening triple constraints of time, cost and scope depends heavily on how the project deals with the ever-present risks.

Risk Management is concerned with future events, whose exact outcome is unknown, and with how to deal with these uncertainties (e.g., a range of possible outcomes) in advance. In general, outcomes are categorized as ranging from favorable to unfavorable, and Risk Management is the art and science of planning, assessing (identifying and analyzing), handling, and monitoring actions leading to future events to ensure favorable outcomes. Thus, a good Risk Management process is proactive in nature, and is fundamentally different from crisis management (or problem solving), which is reactive. In addition, crisis management is a resource-intensive process that is normally constrained by a restricted set of available options. This is in part because typically problem-solving options decrease as the time to discover problems increases. The adverse cost, schedule and/or performance impacts associated with those options are likely to be substantially greater than if the issues had been identified much earlier in the project. This is in part because trade-offs between the triple constraints (cost, schedule and performance) cannot be traded perfectly in the short-run.

Definition of Insanity: Doing the same things over and over again and expecting the results to be different each time. (Paraphrasing Einstein)

1.3 Risk Management Research and Development Program Tasks

There are numerous Risk Management research and development projects ongoing concurrently. These projects are all designed to facilitate the accomplishment of the overall goal of our collaboration:

> ***Research, develop and promulgate suggested Risk Management practices, tools and knowledge to enable and accelerate the transition of Risk Management from an art into a science.***

The projects that are complete, ongoing or planned under the Program are as follows:
(**Bold** indicates ongoing activities)

1.3.1. Risk Planning Activities

a. Develop Risk Management Return on Investment data base.
b. Develop a Risk Management Glossary and FAQ.
c. Develop a Risk Management Reference list with links to documents, books, articles, presentations, etc. Include reviews with as many as possible.
d. Develop suggested Risk Management surveys and feedback forms.
e. Develop Risk Management practitioner certification requirements.
f. Develop Risk Management certification track training courses and workshops.
g. Set up Risk Management Maturity Levels and requirements for attaining each level.
h. Develop a listing of Risk Management tools and provide comments on usefulness.

1.3.2. Risk Assessment Activities

a. Develop and refine a Universal Risk List to aid in Risk Identification activities.
b. Develop and provide suggested methodologies to accomplish Risk Identification tasks.
c. Conduct a survey to determine ideas for the definition of Risk. Provide a suggested definition of Risk.
d. Develop a generic scheme for assessing various types of risk qualitatively and quantitatively. Provide several case studies.
e. Develop suggested methods for prioritizing risks.
f. Develop suggested methodologies for analyzing likelihoods and impacts of risks.

1.3.3. Risk Handling Activities

a. Definitions of the four types of risk handling activities with examples and case studies.

1.3.4. Risk Monitoring Activities

a. Develop suggested Risk Management process and performance metrics
b. Develop communications suggestions and methodologies

Section 2.0 Universal Risk Project Description

The Universal Risk Project is a joint effort of the International Council on Systems Engineering (INCOSE) Risk Management Working Group (RMWG) and the Project Management Institute (PMI) Risk Management Specific Interest Group (RiskSIG). The Project is designed to develop a list and definitions of "universal risk areas" that can be applied to any type of project or operation in any industrial/ government/commercial sector. This list of risk areas is intended to be illustrative and serve only as an aid or a guide to those starting the risk identification process. The list is **NOT**[4] intended to serve as a complete, comprehensive checklist of all risk areas or risks applicable to any specific project. During this project, the collaborators had to accomplish several specific actions. These were to define what is meant by "universal risk area", develop list of risk areas, develop a list of risks under each risk area and define each risk with examples from several sectors.

2.1 Risk Identification

Risk Identification: *determining which risks might affect the project and documenting their characteristics.*[5] Risk Identification is the process of examining the project and each critical technical process to identify and document risks. To begin any sort of risk analysis or risk identification process for a project, examination the project description for any obvious sources of risk is necessary. Risks will vary with the type of project you undertake, the operation to be conducted, etc. Examining the nature of a project's product or the operation's basic processes can identify sources of risk. For example, using proven technology is less risky than developing new technology to use on your project. Projects using new or rapidly changing technology are unique in certain aspects, so there may be no history to help you identify sources of risk[6]. Risk can be associated with any aspect of a project's objectives. Risk identification can be both subjective and objective - based on expert judgment or data.

The result of the Risk Identification Process is a list of risks that pertain to a specific project. One can note these risks as one of several types of Risk Statements. Examples of two types of Risk Statement follow:

1. An IF - THEN type of risk statement.

Example 1: If this technology is not available, then we will not meet the requirement.

Example 2: If we cannot hire sufficient qualified software engineers, then we cannot meet the planned development schedule.

2. A CONDITION - CONSEQUENCE risk statement.

[4] *"Only the naive believe that any risk assessment will discover the totality of risks that may be encountered. One of the first rules of risk management is that there will always be an OSIF (Oh shucks, I forgot). One ignores that fact at one's peril"*. Robert D. Jones

[5] *PMBOK® 2000 Guide*, p. 207.

[6] *"All risk identification must be based on both history and human judgment/reasoning/creativity. The history assessment is the easy part. Conditions change, new circumstances arise, and - in the case of criminals and terrorists - creative people develop new threats. This is why the human part is difficult"*. John A. Cloward

Given the "condition", there is a likelihood that "the consequence" will occur.

Example: Given that this specific test fails, the consequence is that the planned schedule will slip.

2.2 Action 1 - Definition of a "Universal Risk"

After much discussion and debate, the Project participants agreed (as a consensus) to the following definition for a "universal risk": A Universal Risk is ***an event or condition that causes a deviation from the planned and that has a reasonable chance of affecting the conduct or execution of a project, operation of a system or conduct of an analysis and that may occur in any project, operation or system regardless of industry, organization or project/system type.***

Note: We did not try to resolve the debate over whether or not "risk" includes both positive and negative aspects. Since the idea of the Universal Risk Project was to determine universal risks, those risks to a project or operation regardless of how an organization operates, the definition as stated accepts either or both concepts. There is significant discussion on the definition of "risk" in several of the Risk Management Program areas.

The Universal Risk definition is inclusive of contractors and customers, for-profit and not-for-profit organizations, commercial, industrial and governmental sectors. It is also not specific about the likelihood of a particular risk occurring – the "reasonable chance " may be large or small, but as with all risk analyses, it cannot be zero or unity. (0% or 100%)

2.3 Action 2 - Develop List of Risk Areas

Again after much discussion and debate, a small grouping of areas that characterize a project and the environment where risks may occur was accepted (by consensus) by the Project participants. This included three major risk groups or areas as follows:

1. **Management Risk Area** - A set of risks characterizing the organization that owns the project, operation or analysis (hereafter simply called "the project") or are under its control. These include project management, system management and organizational management risk aspects. These areas include the condition of the organization (i.e., whether or not it has multiple or single projects) its resource and personnel culture, organizational tendencies, financial condition and communication or management styles.

2. **External Risk Area -** A set of risks reasonably beyond the control of the organization that owns the project. External risk areas include the actions of others (i.e. customers, other stakeholders, suppliers, regulators, competitors, etc.), climatic forces, demographics, material markets, and economic growth.

3. **Technology Risk Area** - A set of risks inherent in the technology and processes used in the project, product, system or analysis. Technology risks are those risks that embody the capabilities and supporting technologies and processes of the development and operational environments, as well as the end product/system/analysis. This group includes applying technology (Is it more or less complex than anticipated? This is perhaps indicated by whether you have accomplished this specific thing before and how effective are you in applying this type of technology, etc.), pushing technology to or beyond state-of-the-art limits (which may be more difficult to plan for without significant deviation for the expected project plan), how well the project is defined in relation to the technology to be used, adequacy of the technology for the project/system, etc***. Note that this group is the most prone to requiring industry/project/system specific risk identification.***

2.4 Action 3 - Develop list of risks under each area:

Under each major group, it was necessary to define the areas in which specific risks (potential deviations from the planned) that an organization could use to start their risk identification process. The following are examples of the types of specific risks we defined.

1. Management Group

A. Corporate Management Risk Areas

History/Experience/Culture (culture to include organization's reaction to bad news, i.e., Do they shoot the messenger? Do they present rosiest picture to the customer? Is the organization committed to discovering the facts about the project, whether they are positive or negative? Do they commit resources to risk management? Do they have effective communications channels? Can the organization make and stick to a decision? Do they demand and maintain high standards of project discipline?)
Organization Stability (how organized, how often changed? Is an effective organization of risk analysis and management present?)
Methodology/Process (is there a commitment to best practices in their processes? Is there a commitment to establishing realistic project plans given the interaction of cost/time/scope objectives?)

1. Business support processes
2. Scoping/estimation processes
3. Communications processes
4. Project management support processes
5. Integration among the processes

Financial (in terms of financial strength, commitment to timely and adequate funding of projects)

B. Customer-Stakeholder Management Risk Areas

History/Experience/Culture (History includes if they have ever accomplished projects similar to this one before. Deviations from planned results may occur if they do not have pertinent experience with and personnel experienced in this type of project/operation. Culture includes an organization's reaction to bad news about an aspect of the project. Is there a culture of openness and sharing of information

between stakeholders, contractor, suppliers, subcontractors and joint partners on requirements, progress and problems throughout the project? *One significant problem to be noted is a misuse of risk information – using information about potential problems to punish a contractor or cancel/move the project/operation.* ***)***
Customer/User Interaction (including the proposal strategy of bidding to a predetermined cost, schedule and scope or of bidding to an anticipated cost rather than to reality. Does the customer dictate project conditions to the contractor without regard to reality?)
Contractual (including the relationship of contract type to the inherent risk of the project.)
Requirements Definition and Stability (includes clarity of communication of the requirements, inadequacy of requirements, lack of a requirements tracing process, inability to keep requirements up-to-date, volatility of requirements throughout the project, etc.)
Methodology/Process

1. Business support processes (includes human resources)
2. Scoping/estimation processes
3. Communications processes

2. External Group:

A. NATURAL RISK AREAS

Physical Environment (seismic conditions, meteorological conditions, changes over time (climate change))
Local services (electricity, water, natural gas, food, public safety, etc.)
Population (Numbers, types, disposition, stability)
Facilities/Sites – (numbers, types, schedules, requirements)
Transportation Complexity (size, safety, type)

b. CULTURAL RISK AREAS (COUNTRY ENVIRONMENT)

Political (stability, disposition)
Legal/regulatory (requirements, effects)
Interest Groups (depends on the type of project and the organization conducting it)

C. ECONOMIC RISK AREAS

Labor Market (general labor market tightness, availability of specific skills)
Financial Market (cost on money, exchange rates (current and stability))
Labor conditions (type of labor force required, historical experience with labor, labor relations)

3. Technology Group

***A.* TECHNOLOGY REQUIREMENTS RISK AREAS** (DO YOU HAVE SUFFICIENT KNOWLEDGE OF THE REQUIREMENTS AND CONDITIONS IMPOSED BY THE PROJECT THAT YOU ARE ABLE TO SELECT THE MOST APPROPRIATE TECHNOLOGY? ARE THE REQUIREMENTS WITHIN THE TECHNICAL EXPERTISE OF THE PERFORMING ORGANIZATION?)

Scope uncertainty (Do you have complete knowledge of the entire project scope/system operational requirements/analysis requirements? Will this "known" scope change and by how much? Is the scope change control process adequate for the purpose?)
Conditions of use (Are the conditions under which the technology will be used unknown or outside the explored envelope for this technology? Are the conditions imposed on the technology by system or user requirements and the cost or schedule outside the capabilities of this technology? Are the various constraints (cost, schedule and performance) imposed on the project consistent with the application and choice of the technology?)
Complexity (How complex is the project/system operation/analysis? Can it be compared to other projects, systems, or analyses within an industry or to projects/systems operations/analyses being accomplished within your organization?) This area includes **Complexity of Requirements**, (i.e. what is the level of difficulty involved in grasping the system requirements?)

***B.* TECHNOLOGY ADEQUACY RISK AREAS** (ASSUMING YOU KNOW WHAT TECHNOLOGY IS TO BE USED TO ACCOMPLISH THE PERFORMANCE, IS IT ADEQUATE TO THE TASK?)

Technology Maturity (How mature is the technology?)
Technology limits (What are the limits to the technology and are they approached in this project?)

***C.* TECHNOLOGY APPLICATION RISK AREAS** (ASSUMING YOU KNOW WHAT TECHNOLOGY IS NEEDED AND THAT IT WILL BE ADEQUATE FOR THE JOB, CAN YOU SUCCESSFULLY APPLY IT?)

Organizational experience (Does your organization have successful experience in the entire area of the project? An example of this type of risk would be a non-software shop attempting to develop a complex piece of software.)
Personnel skill sets and experience (Do your personnel have the required skill sets and experience to accomplish the project, operate the system or perform the analysis?)
Physical Resources (Equipment and facilities – do you have the required physical resources to accomplish the project, operate the system or perform the analysis within cost, schedule and performance requirements?)

Section 3.0 Specific Risk Area Comments

The wireless music box has no imaginable commercial value. Who would pay for a message sent to nobody in particular?
--David Sarnoff's associates in response to his urgings for investment in the radio in the 1920s.

For several of the Risk Areas, there were a considerable number of “Lessons Learned” provided by Project collaborators. The following subsections provide some of these discussions on “Lessons Learned” the hard way. Some verbiage changes have been made for readability purposes, but none of the meanings have been changed.

3.1 Schedule Risks

Most project or operational schedules are of very low quality. When schedules are correctly linked with logic, they reveal a lot. During both cost and schedule risk analyses, the interviews usually reveal new or revised "most likely" estimates. This is because things have changed or because the original estimates were biased, usually downward to get the bid. The comparison of these new estimates with the original ones can be very revealing.

3.2 Technology Risks

There's another side that needs elaboration here - Software Integration Risk. This, in my humble opinion, is the largest unknown unknown facing us, and it’s due to a real change in the way software is now developed, without a corresponding understanding of the implications or a change in process. Previously, most software was manufactured from scratch. This meant that the developers had full control over the requirements, the design and the build process. If all things were taken in account, improvements could be gained by improving process stability. The Capability Maturity Model is the fulfillment of this.

Unfortunately, now most software is not manufactured from scratch, but is made from a set of integrated components. We no longer make database management systems, message brokers, search engines, product data management systems, web servers, browsers, etc. Each of these components is developed by vendors, who are under pressure to continually upgrade and enhance their products. Each component has its quirky operations, which change with each version. This destroys chances for stable interfaces, as standards usually are behind the latest "sexy" innovations. We can see this in HTML standards, workflow standards, etc.

Additionally, the way each set or "ensemble" of components is used changes radically depending on our individual needs. We can use a DBMS for data warehousing, e-commerce, configuration management, etc. When combining these sets of components into an "ensemble" of components, nobody, including the component developers, have a clue how they will work.

This all leads to the fact that we are no longer in control of either our requirements (we must fit requirements through our set of potential ensembles) or our design. We are limited by the components available, the components we choose, and the integration problems that arise in the combination. None of these things are in our control. This means that even if we clearly define the requirements as stated in CMM, and then fully document the design, we have not addressed the real risk that arises: ***the integration issues associated with combining varied and***

unstable components. It also means we need a cadre of knowledgeable specialists to problem solve the issues associated with component integration.

What this means for development efforts (other than complex military builds in which the old model still works) is that the primary function of system architects should be risk mitigators: they need to engage in exploration and discovery to get a sense of the potential problems prior to deciding on a set of components. Additionally, the idea, as called out in CMM, that requirements can be gathered in buckets and then built into clean and sophisticated designs no longer applies in a market-driven commercial component-based world. A better build approach is to use the Performance-Centered Design approach (from the performance support systems development world), in which the requirements and design process are integrated. The focus is on building an interface and behavior set that is validated through iterative usability testing prior to building a line of code. While building the interface and behavior to validate exact requirements, architects can be engaging in exploration and discovery to identify and mitigate potential integration risks. This approach will lead to better-used systems while more closely meeting cost and schedule goals. While more upfront time is spent, it is now planned for.

Another implication of this approach is that it requires thought prior to action. The initial build will occur later in the process. The good part comes down the line, when there are substantially less testing and integration issues.

3.3 Organizational Risks

Organizations have various characteristics, e.g. multi-projects with resource problems or poor communications. Management can cover more specific failings such as not leading, poor change control, biased estimates (might be endemic to organizations), poor application of project management tools (e.g. scheduling, one of my favorites), etc. These are characteristics of the organization that cause endeavors to miss their targets or fail in spite of good project management.

Organizational characteristics are usually deeply entrenched in the organization's culture rather than being a simple management practice that can be changed overnight. It is seldom "inadequate senior management", per se, but more an endemic problem deeply rooted in the organization's culture. We have known incredibly talented managers who couldn't change the way projects were managed. Overt issues are easy to deal with, but the more common and insidious cases are those where the team pledges to execute the changed processes, really believe they are, but don't have the talent and keep on doing the same old thing -- and don't realize it. Failures continue, and they are left with believing that it is the fault of the "new" process.

Organizational culture has its own inertia, and is difficult to change in a hurry, particularly from below. I have not seen any instances of a crusading project manager making any significant changes in the short term. Consequently, I believe that organizational inadequacies are part of the environment in which we work. By all means try to change them – but it is generally a mistake to regard organizational culture as a temporary barrier to success.

It is the responsibility of senior management to create a culture/environment in an organization where risk management, and project management, can be effective. That includes setting and communicating risk tolerance, policies (such as contingency level expectations), and taking responsibility for their own risk issues. The point of this category on the list is to enable/encourage project managers to explore and communicate these issues with management.

Organizational characteristics are a frequent source of risk. How many of us have had to deal with senior management that thinks that any mention of the word risk is merely evidence of lack of moral fiber? This is a source of risk that is outside the control of the project.

It becomes clear that in such cases we have a business culture environment that is the source of risk. All the other factors, such as poor communications, poor scheduling practices, etc., spring from this.

We do not lack techniques, but we (U.S. of A) do lack management discipline. By discipline, I mean doing the thing that is difficult and time consuming rather than that which is easy and popular.

Having been in the software industry for over thirty years, I recognize that the lack of process and discipline are major contributors to failed projects. However, they are only some of the contributors. Mis-management at the project level is high on the list.

Many times the technical risks are serious but addressed because people tend to look to technical issues first. They are schooled in their identification and solution. The political risks often involve the customer or some outside actor, the prime contractor or some key funding issue. There are two problems here:

1. These are messy and not amenable to easy solution. People do not tend to dwell on problems that are not solvable in deference to the technical that are more solvable, more concrete, and maybe more enjoyable to think on, and
2. To discuss these political issues might be embarrassing, cause client anger and ultimately cause defeat of the program. On one project the estimate was old and OBE, but they could not change the estimate because the political structure was stuck on that number and "did not want to hear" a number larger than that.

Customers cause some of these problems, absolutely. If they refuse to admit risks exist they will cause the contractor to deny the risks that do exist. If the customer causes the risk the problem is worse. On one large job, the customer was going to under fund the design phase. This was causing a huge risk in my estimation, but the contractor could not speak of it. On another, the prime contractor was jerking around the subcontractor (my client) but was seen as doing this forever and having the right to do it. Made a lot of risk but they had to live with it. In another case a funding limitation was causing a schedule risk (remember the triple constraint?) but nobody wanted to deal with it. Items had to be stretched out and hiring was frozen, but these were viewed as conditions of the project, not something to be addressed.

I believe also that organizational risk occurs when the organization cannot make decisions, shifts personnel around, changes priorities, does not commit top priority to the project, interrupts funding, cannot keep an organization stable, etc. These are not technical risks, but can cripple a project. In a recent project with an IT component, the IT professionals I interviewed felt they had their technical act together, though it was challenging. What THEY wanted to make me understand was the difficulty in dealing with the prime contractor and the changing priorities, hiring freezes, etc. Not technical.

I have also seen large project management risks. Project management? That's supposed to be the solution, not the problem? Well, if it is performed poorly or abused that is a problem. I have seen organizations lie on their schedules or have two schedules. I have seen many, if not most, project network schedules that do not calculate the right dates or critical paths, and nobody seems to care. I have seen schedulers and estimators make their projections match top level targets by fudging the numbers. These project management abuses are risks to the project.

So I agree, and in fact believe that these non-technical risks are more universal and insidious than the technical ones. They are just more difficult to control, estimate, quantify, and resolve. And their resolution can cause trouble for the project. Much better, many people believe, to deny they exist and get into trouble than to step up to that particular plate.

3.4 High Impact – Low Probability Risk Identification

Hurricane Andrew (hit Florida in the early 1990's) did $25 billion in damage. $17 billion was covered by insurance, $2 billion paid by assistance (such as FEMA). The remaining $6 billion was just lost. Dade County lost approximately 300,000 permanent jobs due to this storm. The population of Dade County was permanently reduced by approximately 40,000. Tropical storm Allison, which hit Texas in June, was the 3rd costliest storm in US history. The total losses are $5 billion and still climbing. This storm had 37 inches of rainfall over 5 days in one part of Houston. The most rainfall in North America in a 24-hour period was 43 inches from a storm in the Houston area in 1973. So this storm was not even close to that one, but simply had more stuff to drown.

It would be interesting to know how many organizations have considered high impact/low probability "risks" like this in their risk identification and assessment procedures. There are two questions to be asked:

1. Does your organization evaluate such high impact/low probability risks or do they basically ignore them?
2. WHO within your organization "owns" this type of risk?

My experiences over the last three or so years (in a particular environment) are making me think that we cannot identify "High Impact, Low Probability Risks" with any certainty. Just when we think we have a good handle on something, some other little area of the project comes flying out of the dark and wipes out the previously controlled "critical path." What seems to be missing is real engineering discipline. We call every VB or C++ guru a "systems engineer," yet fail to recognize that they are technicians, not engineers. Then, when things fail, we claim that, after all, this is more art than science. If we ignore effective model building and analysis from those models, we will forever be stuck in this poor paradigm for systems development.

3.5 Using Historical Data to Identify Risks

Using past performance is not really working all that well. Systems development, hardware or software, tools and methods are running ahead of what managers know about and can deal with. Witness the current furor over "agile programming" (is that an oxymoronic phrase?). How many developers are actually claiming that they are doing this, when reality is that they are still doing what they have always done, in terms of analysis, design, and development?

3.6 Requirements and Scope Creep Risk

One of the other key contributors is trying to shoot at a moving target using manual methods which are error prone. The software engineering methods and tools are insufficient at best (where is that Software CAD/CAM system anyway). The design, even with design tools such as they are, does not overcome the need for smart people spending lots of time thinking about all the alternatives when creating software applications. Writing the code is the easy part and any programmer can do it. Writing the code that people want is difficult and requires an engineer.

Additionally, requirements and implementation designs are always changing. In a manual development process, changes to requirements cause lots of rework. Add that the complexity of the applications which causes lots of design analysis along with customer's changing expectations as they finally "see" the application, and you have an unknown cost and schedule impact.

The best we can hope for today is to manage the expectations of our customers through continual participation and communication. The bottom line is that any software development effort is high risk! Unfortunately, we can still have shoddy hardware and construction projects, with occasional disastrous results. However, these areas do have some things going for them that constrain the practitioners: standard parts, materials and processes; inspections; codes; licensing and certification. Interestingly, the degree to which these constraints are applied seems to relate to the risk of producing a faulty system. For example, the selection of materials, the design process, manufacturing, user support, etc. are tremendously different for el cheapo plastic toys than they would be for an implanted heart pacemaker.

One of the challenges would be to find how one defines the level of risk that will invoke tighter constraints. Another would be to examine what I will call the granularity of the constraints. Let me explore an apparent tangent to make the point: the idea of software reuse toys with the idea of "standard parts" for software. However, what is the level at which a software construct should be considered a "part?" Is it at the module (which my familiarity with the practice suggests)? But should it rather be at the logical function level? Or perhaps even a single instruction—line of code--should be a part. The analog back to physical parts would be standard materials, equivalent to an instruction, that are used in hardware (semiconductors, wires, boards) equivalent to a logical function, that make up a power supply that is equivalent to a module. Each level has standards, inspections, etc.

I would never attempt to minimize the impact of "scope creep" and ill-defined requirements. However, I don't believe that the majority of software problems with "scope creep" are really problems with changing scope at all! At best, the problems are more likely the results of improperly understood, or improperly communicated requirements. These problems, in turn, result from the tremendous complexity of most non-trivial software projects. In the worst case, when it becomes apparent to the software development team that they under-estimated the difficulty or effort required, they conveniently identify "out-of-scope" effort that, by rights, should be in scope (or would have been in scope if resources were available to accomplish those efforts.)

Until we identify a way to limit the complexity of software projects (systems) or to deal effectively with that complexity, then non-trivial software projects are going to be very risky. The OOA/OOD methodologies and "component-based" software are attempts, but they have a long way to go to approach the predictability of the physical world (making the analogy to electronic or mechanical design using well-described discrete components.)

There are, obviously, other issues that make software development risky and unpredictable, but its inherent complexity is the king of them all. This is a question that should be asked during the risk identification process - what is the process maturity level of the customer versus the process maturity level they require by contract? If they are not the same, then that increases the likelihood of an unrealistic bid being accepted, excessive requirements volatility and other bad things happening. I know that we usually assume risks are independent to simplify the assessment process, but in this case, the potential for a significant increase in several of the risks might make that simplifying assumption too incorrect. Another problem is communication - just how do you tell a potential customer (during your bid or in your proposal) that their "lack of process maturity" increases your risk?

In response to "Scope Creep", I think there are two things that need to be mentioned:

1) I agree that requirements volatility, particularly for software products, is a way of life. Technology changes alone within a 6-month timeframe can cause this to happen along with a changing customer expectation and the competition.

2) Giving an estimate to complete a project up front when only high-level requirements/ features are known is very risky. A way around this is to first give an estimate with boundaries (say + or - a percentage ... and I would use at least 20% at completion of the definition phase). Second, refine the estimate as the development continues. If you have given a decent window around your estimate originally then you will most likely still be in the bounds of that window when you re-estimate later. It would still be very helpful to have good requirements management through the process so that you could identify the upsides easily.

Let me add another perspective to this 'scope creep' thread. I think that it is rare that there would never be a requirements change after any large systems development project has started. I also realize that on some projects it is almost a way of life. But I'll contend that it is not the requirements changes that cause problems, but rather inadequate and sometimes incompetent or entirely missing 'assessment of requirements changes'. In the civil architecting analogy, any change to a building/structure has to be reviewed by the architect (and probably many others) for a cost/schedule impact and the customer (the one paying the bills) is informed of these impacts and has to agree to cost increases or schedule delays. In our software systems business, we somehow seem unable or unwilling to do this very straightforward act of 'assess change, identify impacts, accepts impacts or don't, update cost/schedule as appropriate'. If requirement changes are coming hot and heavy, then that is a major indicator that something (perhaps everything?) is out of control. If the acquirer and supplier cannot put a halt to such requirements violence, then you're likely to encounter a bunch of problems.

I have watched a number of large software projects where the "real" requirements were not truly known before folks started bending metal, cutting silicon, and writing code because there were advertised schedule requirements that "had to be met."

In answer to your question about complexity, I think we have to consider not just complexity, but complexity of requirements. The reason we have to do this is that, in the software world at any rate, we have not developed an adequate technology for visualizing complex requirements, so it is difficult for anyone to have a clear, coherent and comprehensive concept of the requirements. The root of the problem is that software requirements are usually stated in writing, but the written word is not a very effective communication medium (how much time do you spend viewing the world through television compared to the time you spend reading newspapers?)

Going back to the construction project analogy, no developer ordering a new building would be content with written specifications alone (...the building shall have 40 floors...). Instead, we have a well-developed methodology - one might almost call it a technology - of architectural drawings, illustrations and scale models. Furthermore we have an entire profession - architect - whose function it is to convert the developer's business case into a recognizable and implementable form.

The steps involved in creating a new office building might be summarized as follows:

• Developer	- business case for new building
• Architect	- overall concept to implement the business case
• Engineer	- detailed structures to implement the overall concept
• Contractor	- converts detailed plans to physical reality

Along the way there is always a direct visual perception of the end item. Any changes can usually also be perceived visually. If the developer wants a wall moved, it is fairly easy to show the total effort that this would require (changing the underlying structure, re-routing pipes and wiring, etc.).

Compare this now with software development, noting along the way that overall project budgets can be much the same ($10M - $100M, for a large project). Whereas an office building has a readily conceivable and well-defined function, the ultimate functionality of a software project is less easy to grasp. The architecture may be just as complicated as that of a large building, and the inner details of processes and data flows, equivalent to that of a building's piping, wiring and other services, may be more complex. And yet, by and large, we rely essentially on written specifications for everything.

Back to risk reduction. It would seem to me that the best method of risk reduction would be the development of a reasonably standardized method of visual perception of both software requirements and software implementations of those requirements (OK, there are some on the market, but I've not seen one which works well). I suspect that written specs in some form will still be used for the final stage of development, but prior to that we need a really convincing visual method of showing how requirements flow from the business case and the implementation flows from the requirements. Recent e-mails have talked about clients wanting ‘just a little change', without any perception of what that little change implied in terms of overall system changes. One of the requirements of the type of visual system I envisage is the ability to show the effects of changes, in much the same way that, say, the effect of moving a wall in a building can be shown on a scale model of the building.

3.7 Project and Technical Management Risks

The reason why so much software is over budget and late is that, by and large, the software industry pays only lip service to project management. Having been involved in a few software development programs, I would say that the common feature in them is a lack of concrete awareness of a project management structure, unlike, say, a building project. Perhaps this is because building projects have been around for at least 5000 years, whereas software projects have been around for barely 50?

The main program risk, in my experience, is easy to describe but infinitely hard to fix. It almost always centers on an ineffective change management process. Ineffective, in this case, may be either a poor process badly implemented or a good process where management will not enforce the discipline. The result of either is always a bad product which people then subject to as much testing as possible within time and funds constraints to force functionality early, or to test in quality that should have been designed-in to start with. This has gone on so long now that it has become the cultural norm. A really well run program is very, very rare. When you are talking about a Government procurement, the problem is worse because, in order to sell the program to Congress, all the risks and most of the problems have been viewed by the acquisition folks and the contractors through rose colored glasses. Why? Have you heard yourself or your management say, "If we really tell them what it will cost, the program won't be approved. Let's get it started and then work the problem in the out years." Did you answer yes? Well, shucks. How can we expect good decisions when real information has been deliberately or unconsciously suppressed?

On the flip side, is your acquisition organization smart enough to run a reasonable "should cost" for a procurement? If so, how many bids that were clearly too low were simply rejected -- when the bidder couldn't fully justify (and simply claiming a software productivity rate an order of magnitude better than the industry norm isn't proper justification) his cost basis. Were any of these folks allowed to win the contract based on their low bid? If you answered yes to any of the above, then you just identified a major part of the risk. It is kind of easy to fix, too. But, unfortunately, when discussing such things, no one seems to want to talk reality.

Under ideal conditions, the customer would be apprised of the cost and schedule impact of every change so they could make a rational decision about which functions to incorporate, based on priority of need. However, that rarely happens.

I could go on for hours regarding risks forced on a program by a combination of incomplete requirements coupled with a "faster, cheaper, focus -- I know that there should also be a better in there, but that would make it an oxymoron.

Until all parties involved recognize that there are just four elements involved -- content, resources, schedule, and quality -- in every development program, regardless of size, and that when one of those elements changes, at least one of the others must also change, there will always be a major risk built into the development. The fuzzier the initial requirement set, the worse it will be and the more discipline is needed in change management to avoid huge cost and schedule hits. By the way, if you added content, the result is not a cost overrun and to refer to it as such is misleading, at best. You really have created a different contract which simply costs more. It is only an overrun when they can't deliver content initially promised within the budgeted cost. There are enough of those without making the problem worse by adding in content changes. But, I'll admit that there are those who do not agree. They think they should be able to add content or stretch the schedule with no impact on cost.

There has been entirely too much, "Well, build it anyhow and we'll straighten it out later" going on. And, despite many of the good words heard lately in many circles, I have yet to see any real change in either the customer or developer communities when the situation shifts from a theoretical discussion to reality.

How many of your customers have said, "I really don't want to hear that," or words to that effect? How many of you developers have cut corners you knew would result in problems to avoid being "one of those people who doesn't support the program"? For you acquisition folks, how many of you have forced companies to "underbid" the real cost in the name of competition and then screamed bloody murder over the cost overrun when real work starts?

3.8 Customer Management Risks

Let me offer up a point that was made several times here and which I agree with, namely that if the acquisition group for a large complex system (software-intensive or otherwise) is a low maturity outfit but is demanding high maturity (Software Engineering Institute (SEI) Capability maturity Model (CMM) Level 3 or higher), then there are likely to be problems in this situation. I’d offer that if a bidder is selected as the winner of a competition to build a large system, then it is possible (maybe likely) that their bid may not be as realistic (e.g. achievable within cost/schedule) as other bids or even at all. If there is not a savvy acquisition group assessing such bids, then it is certainly possible that the selected winner may already have established a failing project plan/offer without having expended 1 dime of contract funding. All of which is to suggest that it is not only the fault of the PM and the development organization (sometimes it is and sometimes it isn't) but also the fault of the acquisition organization that is paying for these large complex systems and is supposed to be overseeing these developers.

I'd offer that if your customer does not think that their own maturity/capacity/competence is an issue/risk, then telling them that it is is probably a losing position. Certainly suggesting something like this to a customer should be done carefully and very early in your business relationship with them. Including such a statement in a bid is probably a bonehead maneuver. But if we see that such acquisition communities as DoD, the FAA and NASA have recognized this situation as a problem and have begun to increase/improve their own maturity/capability/ competencies, then this would be strong rationale to suggest that your own customer might want to consider the same. All of this would certainly depend on the customer and the situation.

3.9 Complexity Risk

In principle, I agree that complexity exists; in software projects and elsewhere. I also agree that complexity adds risk to an endeavor (required skills, ability to communicate the content, ability to develop a shared appreciation of the goal, etc.). And that is true for all fields of human endeavor. The hard part is cutting through the complexity to find the core concept that makes the rest of the complexity disappear; ask the scientists who are presented with complex theories - most of them make an instant determination that there has to be a "better" way.

I also agree wholeheartedly that complexity is relative. For a developed country, building yet another power substation is a run of the mill event and the skills are widely available, even though very specialized; for a less-developed country, the venture is fraught with risk of all kinds, normally mitigated by bringing in qualified people from somewhere else to additionally provide "technology transfer" as a project deliverable.

I would suggest that we have two special aspects of software development that lead to the type of discussion we have seen so far:

- Innovation: or has this been done before? With several possible answers and mitigation strategies including the one above; I submit that most true "innovation" is incremental and evolutionary; it's the truly "revolutionary" that carries the highest risks and rewards.
- Technology maturity: I submit that the software industry is very immature and that the level of innovation (continually new tools, languages, approaches, etc.) is an indicator of that fact.

I suggest that, in relative terms, both of these aspects of the software industry and software application development lead to the high risk levels that are borne out by the observed results (late, over budget, etc.). I also suggest that when we DO solve the problem of codifying requirements as well as the construction industry (for example) we will begin to observe the degree of predictability that industry (generally) demonstrates.

I also submit for your consideration the idea that the "hidden" nature of software (as discussed by Mr. Graves earlier in this thread) means that many people make a choice (conscious or otherwise) to serve their egos by deciding that their software project is "unique" and "innovative", therefore blocking or failing to seek out any evidence to the contrary and, thus, increasing the risk on their project unnecessarily.

A simple working definition of complexity: ***An entity is defined as complex when it is not possible for the people working with that entity to have a clear and complete concept of it in their minds.***

This definition encompasses not only the intrinsic characteristics of whatever it is we are considering as complex, but also the people trying to deal with it and the conceptual tools available to us. The workings of a petrochemical plant seem very complex to me (all those pipes going everywhere) but is probably conceptually elegant and simple to the engineer in charge. Similarly, Maxwell's equations in their original form were horribly complex, but with the right conceptual tool - matrix algebra - they become elegantly simple. The point of this is that a lot of the risk associated with complexity arises either because we may, at some given point, be low down on the learning curve, and/or because we don't have access to the right people at the right time. This then reduces the risk of complexity to a management-type risk.

Complexity as a risk has many connotations, for instance:

- Complexity can be a risk within one element such as technological complexity, however when this is combined with another element such as human factors, complexity - the risk - is not linear but becomes exponential. Another connotation is **emerging complexity -**

which arises from the scaling up of a system or the interconnection of processes or subsystems. This is often not foreseen at the requirements definition stage, and is usually only realized when it is too late!

- The 'gap' between the system requirements and the users competence becomes an area of risk, which you must decide to accept or mitigate.
- Complexity also must take into account the following: a system is defined as complex when the intended user/operator/ component is unable to intuitively operate the system.
- An example of a non-complex system is a door. We are shown once how to open and close a door, we can then intuitively operate all doors following this.

One point that I'd like some comment on is the difference between **'complex'** and **'difficult'**. I consider **'complex'** to be an attribute that has to do *with "ability to understand something that has many parts"* where as I consider **'difficult'** to an attribute that has to do with *"my knowledge/expertise on a subject matter."* These are related but not identical. I'd offer that it is difficult for me to understand particle physics because I do not have expertise in this technical area. I don't consider particle physics to be complex, primarily because I'm too stupid to know if it is complex or not. Further, I do not consider a systems architecture to be either difficult or complex when described/depicted at a high level. But when further details (i.e., more stuff is added to the picture) are provided, then a systems architecture may become either complex or difficult or both. If I understand the intent of the architecture when depicted at a high level but then I don't get it when more stuff is added, then I'd call the detailed depiction of the architecture to be **'complex'**. If, on the other hand, the additional details allow me to retain my understanding of the overall architecture, then I'd consider this more detailed architecture depiction to not be complex. And if this more detailed architecture depiction introduced some items that I was not particularly familiar with, then I'd claim that this detailed architecture was more 'difficult to understand' but not more complex.

I think that we have 2 related but different topics under discussion. First, I think that complexity is a characteristic of any appropriate noun that can only be measured in a relative sense. That is, A is more complex than B. I am well aware of such measures as Cyclomatic Complexity but I think that any absolute number can only be useful when compared against some other 'complex thing'. Thus, I think that complexity is a risk factor that applies to many types of nouns (systems, software, requirements, organizations, etc.) and that to measure it requires some other noun with some complexity characteristics to compare against.

I believe change management is a key practice that is well known and usable but is not used. This is partially because we are so success oriented in our society (including the workplace), that we do not want to mention anything that hints at failure. This is particularly true in DOD and other government contracts. It even happens in commercial software companies as I can attest. Nothing like being fired for telling the truth. So, we never want to upset the customer with bad news, especially early in a contract when something can be done about the risk before it becomes a project killer.

Lastly, complexity to me does come in many different flavors. There is complex requirements, complex code, and complex design (and as mentioned can be measured) in software. Having complexity on different parts of the process I believe is good. At least we will be forced to consider such risks at the correct point in the process - and not forget it. Maybe even an executive manager would remember the process and ask if a project has complex whatever (what a concept).

There is no question that capturing the requirements of a system's performance is vital and becomes extremely difficult as that system's performance becomes more complex. To use written language exclusively to capture the requirements only compounds the problem. As for tools and methods, the most promising tools I've seen to solve these types of requirements problems involve "use case" descriptions, executable specification, or rapid prototyping in conjunction with spiral development.

Complexity must be viewed in the context of the 'system'. Complexity is influenced by the numbers of parts, different kinds of parts, and different kinds of interaction between parts of the system, as well as the behavior of the parts themselves. It is the case with most software that the tremendous number of parts (functions, operators, routines, structures, etc.) and the richness of the desired system behavior lead to a high degree of complexity.

Attempts to apply techniques developed for the engineering of physical systems to software engineering often fall short because of the abstract nature of software domain. Many software constructs have no physical representation, not even if you stretch the analogy. In some cases, attempts to make a physical representation introduce unnecessary constraints due to human assumptions about physical objects which are inappropriate in the abstract world of software.

Complexity is more like a modifier than like a noun by itself. Increasing complexity increases the risk associated with other factors, such as with requirements definition. It is certain that risk increases much faster than the associated complexity, too! This said, the assessment of complexity must be made within the scope of the factor to which it applies. For example, overall system complexity is not relevant to a technology risk that can be (and presumably is) isolated within a subsystem.

From a risk management standpoint, I've seen no notable use of a quantitative measure for complexity. Complexity can be captured as high, medium, or low, in a way similar to the quantification of risk in terms like high/medium/low. This at least forces the team to consider complexity, but it is still very subjective. Perhaps such an approach isn't bad at all. Once some assessment of the relevant risk is made, appropriate modifications can be applied to the underlying risk factor.

I just finished an analysis of the risk of a software development program. While they had a lot of complexity and were developing a lot of sophisticated new software, the participants talked about organizational and management risks more than technical. They felt that areas such as staffing, not knowing what their prime contractor needed or how it would be tested, and other, more crucial items, kept getting in the way. These were more common in the interviews about project risk than the complexity of the software itself.

I think that the definition of complexity needs to go beyond the traditional idea of system complexity to include organizational, personal, and process complexities and interfaces as well as data, software, and hardware complexities and interfaces. Yes, I know, I snuck the word "interfaces" in there. It's not synonymous with complexity necessarily, but I believe it is as significant as maturity (the two are not orthogonal, in fact, but have a significant interaction term in their impact). I see a lot of risk in all these elements -- especially the "human" and organizational ones.

On the other hand, the management and interpersonal "risks" related to controlling the outside vendor and the rest of the organization, its' processes and the requisite information gathering and organizing process are all activities / skills / challenges that these people are, generally, least well qualified to handle. They are, in fact, the main reason for needing to ensure that the senior PM on the project has these skills and has access to significant pools of resources with more of them and complementary sets needed to fill any gaps.

Of course complexity is not usually a risk – it either exists in a given situation or it doesn't. In that sense then, where it exists, complexity might be a ***source*** of risk, but it wouldn't be a risk according to the definition of "an uncertainty that affects objectives".

I would suggest that the relevant risk-related question should be "Given that we have a degree of complexity, what uncertainties (risks) does that introduce?" It's useful for us to think about understanding complexity of course, but I suggest that we must be careful to keep risks and their causes (and effects) separate.

Thanks for bringing us back to the essence of a risk. If I understand your comment, it is that a degree of complexity exists and should be planned for. The risk is that the actual complexity of the project (technology, organizational, external complexity) differs from that incorporated in the plan, or that the plan's dealing with the complexity does not match the complexity assumed.

The first is more obvious. An interface is assumed to have a certain degree of complexity, but when you get there you find that you have underestimated or overestimated (threat or opportunity) the degree of complexity and the activities take longer (shorter) and cost more (less) than planned.

The second is that the organization correctly evaluates the degree of complexity but underestimates (overestimates) the resources and time necessary to address it.

Both of these risks can occur in a project, and I doubt that it is ever clear which was more present. However, it is good to admit each of these risks when doing a risk analysis and response.

A few thoughts on complexity: I would offer that an entity becomes increasingly complex as the number of entities it interacts with (interfaces with) increases. Within this, I would also offer that the entity itself has been reduced by good systems engineering practices to the simplest entity consistent with the parameters that define entity value. In addition, a system becomes increasingly complex as the number of entities it contains increases.

What does all that mean? Well, if we have decomposed a system down to its basic functions, we can look at how many other functions each function interfaces with. If, for example, one function interfaced with one other function, and another function interfaced with twenty other functions, I would offer that the latter function must be more complex. In addition, if we had reduced a system to its basic functions, and it had a million functions, I suggest we would agree that it is more complex than a system with ten functions. A few conclusions flow from this definitional point.

- First, the entity, or function, can be anything. If the entity/function is a software function, we gauge its complexity by the number of its interfaces with other software functions. If the entity/function is a software development organization, the organization is more complex if it interfaces with a lot of other organizations. The same comments apply to system complexity, regardless of the nature of the system.
- Second, complexity is relative, rather than absolute, in this definition framework. It's like the definition of fat: one can only assess it relative to other things of a similar nature. So, we have to agree for a given type of entity or system when we will call it complex and when we won't. So, what I am suggesting is an operational rather than an absolute definition. This relative scale approach has been lived with by the semiconductor industry for a long time. So long as everyone agrees that we called a million gate IC complex yesterday, but today a ten million gate IC is complex, we are OK. We are especially OK if we know what we should do when we think something is complex, versus what we do when we think something is not.
- Third, this does lead us back to risk. We can make the connection that we will associate increasing risk with increasing complexity. This does mean that things that were risky yesterday are not risky today, given we allow our view of complexity to change with time.

Section 4.0 Universal Risk Project Test Cases

4.1 Universal Risk Project Test Cases

We have evaluated several Test Cases. These are being investigated to determine if all of the "universal risk areas" identified are present and whether the list of universal risk areas can be used appropriately for risk identification. The test cases provided for this report (Appendix 1 and 2) have had each "universal risk" evaluated and placed in a proper perspective for the specific type of project under consideration. An interesting use of the universal risk area list in test case #2 was the three-tier risk identification process - global risks, project risks and individual contract risks. So far, our Universal Risk Area List has been used in several test cases, and has proven appropriate for each type of test case.

Section 5.0 Conclusions

There do seem to be a set of “universal risk areas" that apply to every project, operation, analysis, or system regardless of organizational type, country of origin, or numbers and types of people involved. Note that, having stated the “Universal Risks” apply, for each specific project, the importance (consequence or impact) of each of these “universal risks” changes, some becoming more important, some less. Each must be evaluated relative to the specific conditions of each project. In some cases, the impact of the “universal risk” may be insignificant (an example is political stability in a developed country versus an undeveloped one), but the potentially of the risk is still present. The compilation of this “Universal Risk Area List” has been shown to be useful in providing an organization with a starting point in risk identification for any project being worked, operation being carried out, analysis being accomplished or system being used.

Determining the importance (consequence or impact) of a specific risk for a specific project will be covered in a subsequent Program report.

Appendix 1
Universal Risk Project Test Case #1

1. Project Description
This Project is defined as a design, development, prototyping and initial manufacturing project. It is to design, build and test a total system that is to interface with a larger platform. The system under consideration is one of numerous systems to go on the larger platform. This system contains both hardware and software; will use both Commercial-Off-The-Shelf (COTS) and developed hardware and software; and needs to be built to a performance and cost base. Schedule is not as important as performance and cost since the larger platform will require more time to complete than this system.

The following are the risks noted for this development project.

1. **Management Risk Area**
    - a. Corporate Management
        - i. History
        - ii. Experience
        - iii. Culture
        - iv. Organization
        - v. Methodology/Process
            1. Business support processes
            2. Scoping processes
            3. Estimation processes
            4. Communications processes
        - vi. Financial

    - b. Customer Management– Since this is an example of an external customer, these risks have been moved to the External Group.

    - c. Project Management
        - i. History
        - ii. Experience
        - iii. Culture
        - iv. Organization
        - v. Cost Development
        - vi. Schedule Development
        - vii. Process maturity – process execution, mature system mgt processes
            1. Business support processes
            2. Scoping processes
            3. Estimation processes
            4. Communications processes
    - d. Technical Management

i. History
ii. Experience
iii. Culture
iv. Organization
v. Cost Development
vi. Cost Achievement
vii. Schedule Development
viii. Schedule Achievement

e. Support Environment
    i. Network
    ii. Work Environment
    iii. Training

2. External Risk Area

a. Natural
    i. Physical Environment
    ii. Local services
        1. Electricity
        2. Water
        3. Natural Gas
        4. Public Safety
    iii. Population
    iv. Facilities/Sites
    v. Transportation Complexity
b. Cultural - Country Environment
    i. Political
    ii. Legal/regulatory
    iii. Interest Groups
c. Economic
    i. Labor Market
    ii. Financial Market
    iii. Labor conditions
d. Customer
    i. History
    ii. Experience
    iii. Culture
    iv. Customer/User Interaction
    v. Contractual
    vi. Requirements Definition and Stability
    vii. Methodology/Process
        1. Business support processes
        2. Scoping processes
        3. Estimation processes
        4. Communications processes

3. **Technology Risk Area**
    a. Technology Requirements
        i. Scope uncertainty
        ii. Conditions of use
            1. COTS Components Applicability
            2. COTS Components Quality
            3. COTS Components Logistics
            4. Developed Components Quality
            5. Developed Components Logistics
        iii. Complexity
    b. COTS Hardware Technology Adequacy
        i. Technology Maturity
            1. Number of Manufacturers
            2. Availbility of Component
            3. Cost of Component
            4. Logistics
        ii. Technology limits
    c. COTS Hardware Technology Application
        i. Organizational experience
            1. Experience required to effectively use component
        ii. Personnel skill sets and experience
            1. Experience required to effectively use component
        iii. Physical Resources
            1. Test Resources
    d. COTS Software Technology Adequacy
        i. Technology Maturity
            1. Integration
            2. Functions Used versus Functions Available
            3. Availbility of Module
            4. Cost of Module
        ii. Technology limits
    e. COTS Software Technology Application
        i. Organizational experience
            1. Experience required to effectively use module
        ii. Personnel skill sets and experience
            1. Experience required to effectively use module
        iii. Physical Resources
            1. Test Resources

    f. Developed Hardware Technology Adequacy
        i. Technology Maturity
            1. Technology Maturity
            2. Design/Engineering Maturity

        3. Hardware/Software Interface Definition and Control
        4. Hardware Product Integration Maturity
        5. Hardware/Software Integration Maturity
        6. Fabrication Process
        7. Fabrication Material
    ii. Technology limits
        1. Safety Critical Requirements
        2. Material Limits
g. Developed Hardware Technology Application
    i. Organizational experience
    ii. Personnel skill sets and experience
        1. Development Personnel
        2. Logistics
        3. Integration Personnel
    iii. Physical Resources
        1. Fabrication Resources
        2. Testing Resources
        3. Integration Environment and Resources
h. Developed Software Technology Adequacy
    i. Technology Maturity
        1. Methodology/Process Maturity
        2. Design/Engineering Maturity
        3. Fabrication Process
        4. Fabrication Material
        5. Quality
        6. Data Requirements
        7. Hardware/Software Interface Definition and Control
        8. Hardware/Software Integration Maturity
    ii. Technology limits
        1. Software Limits
        2. Safety Critical Software Requirements
i. Developed Software Technology Application
    i. Organizational experience
    ii. Personnel skill sets and experience
        1. Development Personnel
        2. Logistics
        3. Integration Personnel
    iii. Physical Resources
        1. Devlopmental Support Resources
        2. Testing Resources
        3. Platform Requirements
        4. Testing Support Resources

Appendix 2
<u>Universal Risk Project Test Case #2</u>

1. Project Description

This Project is defined as a design and construction project. It is to build a generic airport, (i.e., green field site, new terminal building, car park, two runways, baggage handling, road network, metro connection, etc.). This construction project contains both the infrastructure required along with the systems necessary for operational readiness; will use Government-Off-The-Shelf (GOTS), COTS and developed hardware and software; and needs to be built to a cost base. Schedule and cost are equally important.

Note that, for this specific test case, the universal risk area list was used for a three-tier risk identification process - **global risk areas** (those risks with the potential to affect the project as a whole and may require a change in overall strategy or reassessment of the global project cost/schedule estimates), **project risk areas** (those risks with the potential to affect each primary project and have a potential to affect the achievement of a project objectives or an interfacing project and may require a change in scope at the project level or the individual contract level and how it interfaces with other projects), and **individual contract risk areas** (those risks with the potential to affect only those areas within an individual contract and are specific to a particular contract and impact directly upon the cost, schedule and efficiency of a specific contract).

Shown below are the airport projects work breakdown structure (WBS), detailing the major projects and a sample of the contracts contained within each project.

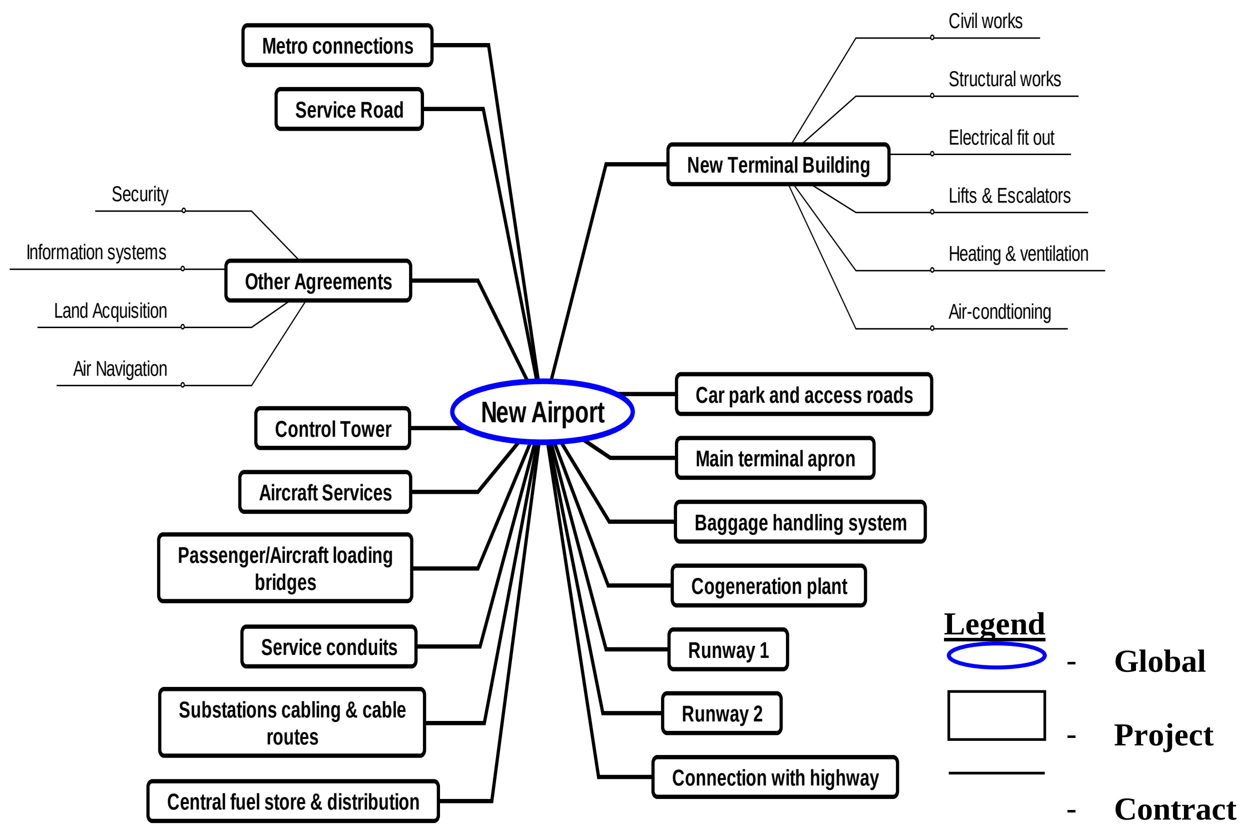

Figure 1.

The following are the risks noted for this development project. They have been placed under appropriate groupings.

Global Risks **–** Those risks with the potential to affect the project as a whole. Risks identified and assessed here may require a change in overall strategy or reassessment of the global project cost/schedule estimates.

1. Management Risk Areas

a. Corporate

1. History
2. Experience
3. Culture
4. Risk Management
5. Organization
6. Methodology/Process
7. Financial
8. Resources
9. Security

b. Customer Risk Areas

1. Experience
2. Culture
3. Customer/operator interaction
4. Contractual
5. Requirements definition and stability
6. Resources
7. Security
8. Policies

c. Project and Project Management Risk Areas

1. Experience
2. Culture
3. Organization
4. Scope development & control process
5. Process maturity
6. Business support processes
    i. Scoping processes
    ii. Communications processes
    iii. Hand-over processes
    iv. Change order processes
7. Schedule development
8. Master Planning development
9. Monitoring regime
10. Cost development
11. Monitoring regime
12. Interface management
13. Risk mitigation practices

d. Technical Management Risk Areas

1. Design regulatory requirements
2. History
3. Experience
4. Culture
5. Organization
6. Cost development
7. Cost achievement
8. Maintenance regime
9. Resources
10. Schedule development
11. Airspace planning

e. Support Environment Risk Areas

1. Training

2. Work Environment
3. Safety Culture
4. Network
5. Manufacturing

2. External Risk Areas

a. Environmental Risk Areas

1. Noise
2. Air Quality
3. Water
4. Soil
5. Vegetation
6. Fauna
7. Archaeology
8. Land
9. Restoration
10. Public Objections/enquiry
11. Wildlife

b. Cultural - Country Environment Risk Areas

1. Regulatory requirements
2. Political trends
3. Maturity
4. Languages
5. Terrorism
6. Government stance
7. Stability

c. Economic Risk Areas

1. History
2. Currencies - exchange rates
3. Business cycle – market buoyancy
4. Macro-economic trends/indicators
5. Micro-economic trends/indicators
6. Stability
7. Fiscal Policies
8. Inflation
9. Unemployment rate
10. Credit ratings
11. Accountancy systems/regulations
12. GDP/GNP affects and objectives
13. Forecasts –
    - i. Passengers
    - ii. Airplanes

iii. cargo

d. Customer - Stakeholder Risk Areas

1. Unions
2. Concessionaires
3. Trends
4. Forecasts
5. Demographics

e. External Projects Risk Areas

1. Airline alliances
2. Cargo alliances
3. Hotels
4. Business Parks

f. Regulatory Bodies Risk Areas

1. FAA requirements
2. Fire service requirements
3. Customs
4. Quarantine
5. Planning

g. Subcontracts Risk Areas

1. History
2. Experience
3. Culture
4. Organization
5. Cost development
6. Cost achievement
7. Maintenance regime
8. Resources
9. Schedule development

h. Transportation Risk Areas

1. Current layout
2. Required layout
3. Site logistics

i. Facilities Risk Areas

1. Hospitals
2. Hotels
3. Catering

j. Services and Utilities Risk Areas

4. Water
5. Power
6. Sewerage
7. Landfill
8. Raw materials

9. Fuel

3. Technology Risk Areas

a. System integration
b. Network architecture requirements
c. Technology Requirements
d. COTS Hardware technology adequacy
e. COTS Software Technology application
f. Developed hardware technology adequacy
g. Developed hardware technology application
h. Developed software technology adequacy
i. Developed software technology application
j. Testing support resources
k. GOTS hardware technology adequacy
l. GOTS hardware technology application
m. GOTS software technology adequacy
n. GOTS Software technology application
o. Operational readiness
p. Commissioning support resources
q. Decommissioning of old technology
r. Lifecycle costs
s. Maintenance & upgrade schedule (i.e. UNIX variants)
t. Safety requirements of technology
u. Scalability of COTS & GOTS systems
v. Security & confidentiality of data
w. Intellectual property rights of software
x. Version supportability of software
y. Version supportability of hardware
z. Maturity of technology
aa. Technological maturity of users

Project Risks – Those risks with the potential to affect each primary project (typical risks at this level have a potential to affect the achievement of a projects objectives or an interfacing project and may require a change in scope at the project level or the individual contract level), and how it interfaces with other projects. Results from this assessment may require changes to the individual contracts, schedules and costs or allowances against specific risks.

1. Management

a. Corporate

1. History
2. Experience
3. Culture
4. Organization

5. Methodology/Process
6. Financial
7. Resources
8. Security

b. Customer

1. Experience
2. Culture
3. Customer/operator interaction
4. Contractual
5. Requirements definition and stability
6. Resources
7. Security

c. Project

1. Experience
2. Culture
3. Organization
4. Process maturity
5. Business support processes
6. Scoping processes
7. Communications processes
8. Handover processes
9. Change order processes
10. Schedule development
11. Monitoring Regime
12. Cost Development
13. Interface Management

d. Technical

1. Design regulatory requirements
2. History
3. Experience
4. Culture
5. Organisation
6. Cost development
7. Cost achievement
8. Maintenance regime

e. Support Environment

1. Training

f. Safety

2. Contractor culture
3. Regulatory requirements
4. Training

5. Auditing

2. External

a. Environmental

1. Noise
2. Air Quality
3. Water
4. Soil
5. Vegetation
6. Fauna
7. Archaeology
8. Land
9. Restoration

b. Cultural-Country Environment

1. Regulatory requirements
2. Political trends
3. Languages

c. Economic

1. Macro-economic trends/indicators
2. Micro-economic trends/indicators
3. Stability
4. Payment periods
5. Interest rates
6. Inflation

d. Customer

1. Unions
2. Working practices

e. External Projects

F. Regulatory Bodies

1. FAA requirements
2. Fire service requirements

3. Technology

a. System integration

b. Technology Requirements

c. COTS Hardware technology adequacy

d. COTS Software Technology application

e. Developed hardware technology adequacy

f. Developed hardware technology application

g. Developed software technology adequacy

h. Developed software technology application

i. Testing support resources

j. GOTS hardware technology adequacy

k. GOTS hardware technology application

l. GOTS software technology adequacy

m. GOTS Software technology application

n. Operational readiness

o. Commissioning support resources

p. Viability of suppliers/manufacturers

q. Lifecycle costs

r. Maintenance & upgrade schedule (i.e. UNIX variants)

s. Scalability of COTS & GOTS systems

t. Intellectual property rights of software

u. Version supportability of software

v. Version supportability of hardware

w. Maturity of technology

x. Technological maturity of users

<u>Individual Contract Risks –</u> Those risks with the potential to affect only those areas within an individual contract. These risks are specific to a particular contract and impact directly upon the cost, schedule and efficiency of a specific contract.

<u>1. Management</u>

a. Corporate

1. History
2. Experience
3. Culture
4. Organization
5. Methodology/Process
6. Financial
7. Resources
8. Security

b. Customer

1. Experience
2. Culture
3. Customer/operator interaction
4. Contractual
5. Requirements definition and stability
6. Resources
7. Security

c. Project

1. Experience
2. Culture
3. Organization
4. Process maturity
5. Business support processes
    - i. Scoping processes
    - ii. Communications processes
    - iii. Handover processes
    - iv. Change order processes
    - v. Schedule development

6. Monitoring regime
7. Cost development
8. Interface management

9. Procurement

d. Technical

1. Design regulatory requirements
2. History
3. Experience
4. Culture
5. Organisation
6. Cost development
7. Cost achievement
8. Maintenance regime
9. Fabrication

e. Support Environment

1. Training

f. Safety

2. Regulatory requirements
3. History
4. Culture

2. External

a. Environmental

1. Noise
2. Air Quality
3. Water
4. Soil
5. Vegetation
6. Fauna
7. Archaeology
8. Land
9. Restoration

b. Cultural-Country Environment

1. Regulatory requirements
2. Political trends

c. Economic

1. Micro-economic trends/indicators
2. Stability

d. Customer

1. Unions

e. External Projects

f. Regulatory bodies

2. FAA requirements
3. Fire service requirements

3. Technology

a. System integration

b. Technology Requirements

c. COTS Hardware technology adequacy

d. COTS Software Technology application

e. Developed hardware technology adequacy

f. Developed hardware technology application

g. Developed software technology adequacy

h. Developed software technology application

i. Testing support resources

j. GOTS hardware technology adequacy

k. GOTS hardware technology application

l. GOTS software technology adequacy

m. GOTS Software technology application

n. Operational readiness

o. Commissioning support resources

p. Lifecycle costs

q. Maintenance & upgrade schedule (i.e. UNIX variants)

r. Safety requirements of technology

s. Scalability of COTS & GOTS systems

t. Security & confidentiality of data

u. Intellectual property rights of software

v. Version supportability of software

w. Version supportability of hardware

x. Maturity of technology

y. Technological maturity of users