Computer Aid, Inc. presents

Conversations on

Effective IT Management
Every task we undertake and every project we initiate encounters some degree of risk. Successful managers identify and evaluate risks in order to determine if the risks should be accepted or mitigated. The risk assessment process should measure the probability (likelihood) of the risk occurrence and the resulting impact or severity of the occurrence.

Calculating the probability of a risk is difficult especially when there is little historical data. With respect to IT projects, the experience of the project manager, the maturity of the management processes, the completeness of the requirements, and the cooperation of business executives are examples of risks that cannot be mathematically calculated.

How do we measure the potential impact? Most failures occur as a partial failure, i.e., a single defect, often a single line of code. How do you measure the “failure” of a line of code, when its impact on business depends on precisely which line of code has failed? It can be argued the time lag between occurrence and detection results in the greatest impact.

This release of Conversations on Effective IT Management includes interviews with subject matter experts from the academic and business world who discuss risk and the processes involved in evaluating and measuring it. Many of the issues are the result of human nature: people don’t like to think about, discuss, or expend much effort on risk mitigation. Project teams focus their attention on risks that impact schedules (“Will we meet our deadline?”) rather than the future impact of undetected errors. In the world of software maintenance, where business impact is more immediate, the focus is on containment and recovery after a failure rather than prevention.

Risk assessment is both an art and a science. The science requires a defined process to consistently evaluate, measure, and mitigate risk. The art involves applying subjective values and experience to the various components under evaluation.

The interviews in this report discuss the value of communicating uncertainties and evaluating the probability of failure in the software development life cycle.
Contents

Tim Lister

Changing Your View of Risk

A great risk manager is one who not only watches the risk at hand but also searches through consultation with others for new risks likely to appear as the project’s environment changes.

Dr. Victor Basili

A Question of Measurement

The Goal Question Metric (GQM) paradigm is the most popular way to measure a model. The idea is to build an "experience factory" that will let you predict and optimize everything you’ve got.

Dr. Robert Charette

No Time to Waste

When you’re looking at risk management, there are two things you need to do. First, you need to prioritize your risks. Second, you need to mobilize [your solution] against them.

Dr. Robert Charette

Why Software Fails

As our society comes to rely on IT systems that are ever larger, insight is needed into what may go wrong and what can be done to eliminate or mitigate the risk.

Tony Salvaggio

IT Risk Management

An insightful perspective that deals with how organizations address IT Risk Management on individual projects as well as throughout the global enterprise.
A CONVERSATION WITH:

Tim Lister

Changing Your View of Risk

A great risk manager is one who not only watches the risk at hand but also searches through consultation with others for new risks likely to appear as the project’s environment changes.

Extracted from an interview conducted by Michael Milutis

CAI: How would you actually define risk management and its key components?

TIM LISTER: Risk management is the process of unearth both uncertainty and risk in projects. It asks what the unwanted possible consequences of an event or a decision are. The essence of risk management software is to help us decide whether to deal with problems before they appear or to wait till they clearly emerge and then deal with them as problem management. I’ve been in software all my life, so I may be biased, but I think software particularly lends itself to risk management because more often than not you have to fight early problems rather than late ones.

A classic early problem occurs when a looming deadline looks tight and you may need to hire more people to make it. Getting people early and integrating them into the project can help you enormously, whereas waiting too long to hire and train additional people is often wasteful and useless. Risk management is about understanding when to make decisions. It involves a conversation among all of the stakeholders – the technical people, the sponsors, the users, the managers about the best time to make decisions. At risk time, we ask the question of whether to spend some money to lower the probability of a problem or how to lower the...
cost should the problem occur, or some combination thereof.

Risk management assumes that careful study of a plan will result in someone spotting a potential problem. Risk management says, "Let's identify and understand the risks early, determine their root causes, and decide whether it makes good business sense to spend money before or when the problem hits." There is no way to identify all the risks at the start of a large project and manage them down. A great risk manager is one who not only watches the risk at hand but also searches through consultation with others for new risks likely to appear as the project’s environment changes.

The major components of risk management are identification, evaluation, prioritization, and strategizing. Just because somebody identifies or "nominates" (a term I prefer) a risk early doesn't mean we are going to do anything about it. We might accept it and pay for it later if we judge there is no advantage in tackling it now.

Another aspect of evaluation is judging the probability of a risk becoming a problem. Are we looking at a one in a thousand or a 50/50 risk? And then there's cost evaluation. If the risk hits, what's it going to cost us in terms of manpower, delay on schedule, money? In the prioritization component, we ask what are the most important risks in terms of probability and cost. Finally, there is the strategizing component where we ask such questions as the following: When do we make the call? What are our options here? How long can we delay before we take action or should we act immediately and mitigate the risk up front?

CAI: How then would you characterize the current state risk management practices throughout corporate IT organizations?

TIM LISTER: Sadly, I would say the vast majority of organizations are not practicing real risk management. A small minority do it very well. There are also some who say they do it, but what they do is identify risk and go back to business as usual. They may have a little step early in their process that says, "Identify risk, evaluate risk," but there is no evidence they do anything with that information.

In genuine risk management, you change something on a big project based on rigorous risk assessment. You change your development strategy, you change the sequence, the definition of the project, the schedule, the staffing, and you keep a detailed record and rationale of the decision-making which led to such changes. This kind of risk management is rather rare.

Continued on Page 19
Dr. Victor R. Basili is Professor of Computer Science at the University of Maryland. He holds a Ph.D. in Computer Science from the University of Texas and honorary degrees from the Universities of Sannio (Italy) and Kaiserslautern (Germany). He was Executive Director of the Fraunhofer Center Maryland and a founder and principal of the Software Engineering Laboratory (SEL) at NASA/GSFC.

He is a recipient of numerous awards including a NASA Group Achievement Award, a NASA/GSFC Productivity Improvement and Quality Enhancement Award, the 1997 Award for Outstanding Achievement in Mathematics and Computer Science by the Washington Academy of Sciences, the 2000 Outstanding Research Award from ACM SIGSOFT, and the 2003 Harlan Mills Award from the IEEE Computer Society.

Dr. Basili has authored over 200 papers, served as Editor in Chief of several journals (IEEE TSE, Journal of Empirical Software Engineering) and program chair and general chair of several conferences (ICSE). He is an IEEE and ACM Fellow.

A CONVERSATION WITH:
Dr. Victor Basili

A Question of Measurement

The Goal Question Metric (GQM) paradigm is the most popular way to measure a model. The idea is to build an “experience factory” that will let you predict and optimize everything you’ve got.

Extracted from an interview conducted by Michael Milutis

CAI: What is the value of process and measurement? Why is it so important?

VICTOR BASILI: Whatever the product, whether it’s software or something else, process is important, because it is the only way to optimize what you are doing. And measurement is important because it’s the only way you can observe and get feedback about what is really happening.

CAI: To what extent is a successful measurement initiative contingent upon standard processes? How do these two things hang together?

VICTOR BASILI: I have a slightly contrarian perspective on this. I think process is a variable that needs to be tailored to the specific problem at hand. Although you may have lots of commonalities in your processes, they each have to be a little bit different for various projects.
While we were at NASA Goddard, we developed an approach to this called quality improvement. The first step in using this model is to collect data that doesn't necessarily tell you where you are or what you're doing. We called it “characterized” data but I heard you guys call it “visualized” and I like that term. I like it because that's what you are doing – you are visualizing what's going on in your environment.

The second step is to set your goals for particular projects and also for the entire organization. Where does this organization want to be and how will this project contribute to the overall knowledge that is within the organization?

In the third step you choose your process, one that allows you to achieve your goals relative to your environment, relative to what the business is about, and relative to what you've got in the present moment.

The fourth step is to do it.

The fifth step is to try to do as much feedback and analysis as you can in real time in order to help manage the project and to help increase learning from that project. Your real goal is to learn from every project you perform at a particular organization.

The sixth step is about the analysis you do after the project has been completed. You've done as much analysis as you can in real time, and that's usually very complicated. But now you must do more analysis. With post-project analysis we try to recognize what really happened, what was a success, what wasn't a success, etc.

The seventh step is to package all of this knowledge so that it becomes part of your processes, part of your organization, part of the way you think about how you solve all of your problems. And then the next project comes through and you keep going. You build up and save this knowledge in what we call an experience base.

The idea at the root of all of this is to build an “experience factory” within an organization. Such an experience base can tell you at any given point how your projects are being developed. But while all of this is happening, you are also learning that every project is an experiment. You're testing and observing what is happening, what should have happened, and what will happen. And you're making changes to the way you understand your organization.

At the end of all of this you will end up with a lean and optimized set of processes for various classes of problems. Your future projects will be different, but that's OK because you'll have classes of processes that will work for different kinds of projects. You will be able to conduct predictions and optimize everything you've got, including code.

"
A CONVERSATION WITH:
Dr. Robert Charette

No Time to Waste

When you’re looking at risk management, there are two things you need to do. First, you need to prioritize your risks. Second, you need to mobilize [your solution] against them.

Extracted from an interview conducted by Michael Milutis

CAI: How would you define software risk management? Can you break it down into key components for us?

ROBERT CHARETTE: I look at software risk management as I look at risk management as a whole. It’s all about making high quality decisions through high quality risks. Risk management is just an element of decision making and software risk management is just a sub-component of systems engineering risk management which, in turn, is a sub-component of business or enterprise risk management. I don’t believe that you can say there’s something called software risk management without also saying that you’re involved in systems risk management or in business risk management. The three are really intimately interconnected. This gets back to the linkage between strategy and technology.

To understand this linkage, you first of all need to worry about principles. What is it, for instance, that you are trying to accomplish in terms of managing risk? What are the things that you really value as an organization when it comes to risk? For instance, are you a risk-taking company or a risk-averse company? Which one will determine your risk tolerance? What are the things that you value as an organization in terms of managing risk? For example, is open and honest communication of risk a core organizational principle?

The second thing you need to worry about is the risk management process itself. What do you need to have in place that is visible, repeatable, and measurable in terms of a risk management process?

The third thing is behavior. Behavior is critical because when we make choices, we intend to act. When it comes to risky situations, we need to think about what we want people in our organization to do as well as what we don’t want them to do.
When you're looking at doing risk management, you need to create a principle-based, process-focused, behavior-driven system — a system in which any two of its pillars support the third. For example, the principles and processes you create must condition the risktaking behaviors you desire within your organization. Similarly, the behaviors and processes should reinforce the risk principles you value.

You have to understand what it is that you want people to do when they are faced with risk, when they are faced with something that may make it look like they're not going to succeed. You have to convince people to look at things differently than they normally do. But within a framework; otherwise you risk making the kinds of mistakes that ensue from more ad hoc approaches.

What has always been intriguing to me about risk management is that, superficially, it appears extremely easy. You simply look at what might occur, you clarify how these things might hurt you, and then you develop some approaches to keep the bad outcomes from coming into being.

However, while the process itself can appear to be quite superficial, it quickly becomes extremely subtle and complex. That's because risks are perceived and unreal. They are only possibilities. They are not actual things. By the time risks become actual things, they are already problems and at that point they cease to be risks.

Furthermore, when you try to manage or reduce these probabilities, you quickly come up against a perplexing problem: if one allows resources to be spent on the reduction of risk, will the probability of project success be increased or, in fact, reduced? In other words, if you are spending finite resources to reduce mere probabilities, things that might not happen, couldn't (and shouldn't) you also be applying those same resources to things that you know really are happening?

As you can see, the process can quickly become very messy. It is often quite counter intuitive, and that is something that is eternally fascinating to me, because were dealing here with potentialities, with trade-offs, with futures, and to really understand what is going on with these intricacies, your thought processes have to be broad, deep, and quick. It's like the old saying "cheaper, faster, better - pick any two." In managing risk, you often can identify and prioritize the risks, but you may not be able to mobilize to actually deal with them.

And if you manage to mobilize the right resources to deal with one set of risks, you will simultaneously be making a conscious choice not to mobilize against some other set of risks, which effectively means that you are accepting those risks. In this respect, knowing what your opportunity costs are is going to be key.
CAI: Would you be able to quantify the percentage of IT organizations that are using risk management practices properly and getting positive benefits from them?

ROBERT CHARETTE: One of my side jobs is that I am the Director of Enterprise Risk Management and Governance for the Cutter Consortium. We did a study back in 2002 that took a look at organizational risk management practice. To answer your question, we found that 51% of the organizations we surveyed claimed to be using some sort of formal approach to assess or manage risk. In other words, 51% had some sort of repeatable process that they were following. Nevertheless, of the organizations we surveyed, only 39% were applying software risk management practices. In fact, risk management was still a fairly new practice within the organizations we surveyed. On average, we found that companies had been using risk management for only four or five years. Consequently, program and project risk management has yet to be integrated into a corporate approach to managing risk.

What was interesting, though, is that although there was a minority of people using software-specific risk management practices, 90% of the people in our survey agreed that managing IT risk was either important or very important for achieving project success. In fact, 75% believed that software risk management made their projects more successful than projects that didn’t employ risk management practices.

However, if I were to refer simply to my own personal experiences, I would say that the number of people who are using software risk management practices effectively is probably in the 20-30% range, and I’m probably being optimistic here. One of the problems that I regularly encounter even in organizations like the Department of Defense, where risk management practices have been mandated now for almost 30 years, is that although a large number of organizations are using risk management, its practice is really just pro-forma. In other words, they’re applying a tick-in-the-box risk management process, and it’s not affecting organizational decision making in any way.

CAI: Could you highlight for us what, in your opinion, might represent the top three software development risk factors?

ROBERT CHARETTE: How about the top 100?

The primary factor I see is the lack of realism. Our industry likes to over-promise and under-budget. We seem addicted to unrealistic objectives and unrealistic goals, even in the face of very complex projects. We pretend that we know more than we do, and then feign surprise when things don’t go as planned.

The second major factor lies in the fact that, as an industry, we tend to be very sloppy in terms of our development practices. If you take a look at the Software Engineering Institutes CMM or CMMI results, you will see that the vast majority of organizations are still employing undisciplined or chaotic development practices. Poor project management practice is pervasive throughout the development world today. And poor project management will take a project down faster than any other type of risk factor except the lack of realism.

The third major factor revolves around politics. Projects do not sit in an objective, purely rational vacuum. They are part of a greater whole, one involving the political realities of an organization. Most people don’t manage organizational politics well, nor do they recognize their
importance to project success.

Each of these three factors, and there are certainly more, must be examined, understood, and managed in a very aggressive, realistic, holistic, and honest way. And we should not ever underestimate the importance of honesty. We must always ensure that our objectives are both realistic and honest. The paradox with honesty, of course, is that you might have a hard time getting your project supported if you are totally honest about the risks that exist. The temptation to over-promise is rooted in this paradox. To be unrealistic, however, is to court disaster. That is far worse.

What this means is that until we get a development environment both at the business end and at the technical end, a fact-based environment in which we can be honest, then all of our risk factors will just be exacerbated.

**CAI:** Once identified, organizations could spend years investigating their own risk items. In light of this, what is the most practical approach for proceeding, once the risk identification phase has been completed?

**ROBERT CHARETTE:** There are two things you need to do. First, you need to prioritize your risks. Second, you need to mobilize against them.

Regarding prioritization, there are two simple questions that you can ask yourself here: 1) what is going to hurt me the most; and 2) what is going to hurt me soonest? You must deal with these risks right away; specifically, the ones that are going to keep you from accomplishing the next milestone or the next objective in your schedule.

Regarding mobilization, and this is an area that people tend to forget about, you must remember that a risk hasn’t gone away just because you’ve allocated resources to try to avert it. You’re not done until your mitigation strategy has actually accomplished its goals. So once again, in the short term, attack those things that are going to cause the most amount of damage to you soonest. Second, be aware of the great danger posed by the medium risks, too. The medium level risks are the ones that really can hurt you because they are the ones that you tend to accept. And if you have enough of them, they can overwhelm you. The worst thing in the world, in my opinion, is having lots and lots of medium level risks on your project. I’d much rather manage a project that has lots of reds and greens; don’t give me one with lots of yellows.

Finally, keep in mind, despite their very low corresponding probability, that there are still some extremely high consequence risks that may be able to take you out. Keep a close eye on them.

**CAI:** How would you characterize the importance of processes in all of this? From a process perspective, what in your opinion are the critical success factors for effective software risk management?

**ROBERT CHARETTE:** First of all, you need to have some measures. You need to have information that is fact-based or evidence-based. Whether or not you call them software measures, or performance measures, it doesn’t really matter to me. What I’m interested in is having something that I can objectively measure against, and then predict against. I also need a process that’s going to help me evaluate not only my objectives, but also my assumptions and constraints. We tend not to look at our assumptions. One of the things that I frequently tell organizations is that if they can’t perform a full-blown risk assessment they should at least conduct an assumptions analysis, because its the assumptions that underpin your project. You need to constantly test those assumptions against reality.

**CAI:** You mentioned the importance of having measures, of being able to objectively measure against things in order to develop a starting point. What do you think of the relative value of external versus internal benchmarking data?

**ROBERT CHARETTE:** You have to get information from the inside of your organization. Set up your measurement programs, start getting data, and at that point compare what you have with the external world. I should also mention that I rarely see organizations that have effective risk management programs in place without also having very effective measurement programs as well. It’s kind of a chicken and egg problem, though. Do you start with a measurement program first and a risk management program second, or vice versa? I’m not sure, but if your leadership is willing to simply state “Where are we?” that’s a good start.
Dr. Robert Charette

Why Software Fails

*As our society comes to rely on IT systems that are ever larger, insight is needed into what may go wrong and what can be done to eliminate or mitigate the risk.*

_Have you heard the one about the disappearing warehouse? One day, it vanished—not from physical view, but from the watchful eyes of a well-known retailer’s automated distribution system. A software glitch had somehow erased the warehouse’s existence, so that goods destined for the warehouse were rerouted elsewhere, while goods at the warehouse languished. Because the company was in financial trouble and had been shuttering other warehouses to save money, the employees at the “missing” warehouse kept quiet. For three years, nothing arrived or left. Employees were still getting their paychecks, however, because a different computer system handled the payroll. When the software glitch finally came to light, the merchandise in the warehouse was sold off, and upper management told employees to say nothing about the episode._

This story has been floating around the information technology industry for 20-some years. It’s probably apocryphal, but for those of us in the business, it’s entirely plausible. Why? Because episodes like this happen all the time. Last October, for instance, the giant British food retailer J Sainsbury PLC had to write off its U.S. $526 million investment in an automated supply-chain management system. It seems that merchandise was stuck in the company’s depots and warehouses and was not getting through to many of its stores. Sainsbury was forced to hire about 3,000 additional clerks to stock its shelves manually.

This is only one of the latest in a long, dismal history of IT projects gone awry. Most IT experts agree that such failures occur far more often than they should. What’s more, the failures are universally unprejudiced: they happen in every country; to large companies and small; in commercial, nonprofit, and governmental organizations; and without regard to status or reputation. The business and
societal costs of these failures — in terms of wasted taxpayer and shareholder dollars as well as investments that can’t be made — are now well into the billions of dollars a year.

The problem only gets worse as IT grows ubiquitous. This year, organizations and governments will spend an estimated $1 trillion on IT hardware, software, and services worldwide. Of the IT projects that are initiated, from 5 - 15% will be abandoned before or shortly after delivery as hopelessly inadequate. Many others will arrive late and over budget or require massive reworking. Few IT projects, in other words, truly succeed.

The biggest tragedy is that software failure is for the most part predictable and avoidable. Unfortunately, most organizations don’t see preventing failure as an urgent matter, even though that view risks harming the organization and maybe even destroying it. Understanding why this attitude persists is not just an academic exercise; it has tremendous implications for business and society.

SOFTWARE IS EVERYWHERE. It’s what lets us get cash from an ATM, make a phone call, and drive our cars. A typical cell phone now contains 2 million lines of software code; by 2010 it will likely have 10 times as many. General Motors Corp. estimates that by then its cars will each have 100 million lines of code.

The average company spends about 4 - 5% of revenue on information technology, with those that are highly IT dependent — such as financial and telecommunications companies — spending more than 10% on it. In other words, IT is now one of the largest corporate expenses outside employee costs. Much of that money goes into hardware and software upgrades, software license fees, and so forth, but a big chunk is for new software projects meant to create a better future for the organization and its customers.

Governments, too, are big consumers of software. In 2003, the United Kingdom had more than 100 major government IT projects underway that totaled $20.3 billion. In 2004, the U.S. government cataloged 1,200 civilian IT projects costing more than $60 billion, plus another $16 billion for military software.

WHEN A PROJECT FAILS, it jeopardizes an organization’s prospects. If the failure is large enough, it can steal the company’s entire future. In one stellar meltdown, a poorly implemented resource planning system led FoxMeyer Drug Co., a $5 billion wholesale drug distribution company in Carrollton, TX, to plummet into bankruptcy in 1996.

IT failures can also stunt economic growth and quality of life. Back in 1981, the U.S. Federal Aviation Administration began looking into upgrading its antiquated air-traffic-control system, but the effort to build a replacement soon became riddled with problems. By 1994, when the agency finally gave up on the project, the predicted cost had tripled, more than $2.6 billion had been spent, and the expected delivery date had slipped by several years. Every airplane passenger who is delayed because of
gridlocked skyways still feels this cancellation; the cumulative economic impact of all those delays on just the U.S. airlines (never mind the passengers) approaches $50 billion.

Worldwide, it’s hard to say how many software projects fail or how much money is wasted as a result. If you define failure as the total abandonment of a project before or shortly after it is delivered, and if you accept a conservative failure rate of 5%, then billions of dollars are wasted each year on bad software.

**WHY DO PROJECTS FAIL SO OFTEN?**

Among the most common factors:

- Unrealistic or unarticulated project goals
- Inaccurate estimates of needed resources
- Badly defined system requirements
- Poor reporting of the project’s status
- Unmanaged risks
- Poor communication among customers, developers, and users
- Use of immature technology
- Inability to handle the project’s complexity
- Stakeholder politics
- Commercial pressures
- Sloppy development practices
- Poor project management

Of course, IT projects rarely fail for just one or two reasons. The FBI’s VCF project suffered from many of the problems listed above. Most failures, in fact, can be traced to a combination of technical, project management, and business decisions. Each dimension interacts with the others in complicated ways that exacerbate project risks and problems and increase the likelihood of failure.

Consider a simple software chore: a purchasing system that automates the ordering, billing, and shipping of parts, so that a salesperson can input a customer’s order, have it automatically checked against pricing and contract requirements, and arrange to have the parts and invoice sent to the customer from the warehouse.

The requirements for the system specify four basic steps. First, there’s the sales process, which creates a bill of sale. That bill is then sent through a legal process, which reviews the contractual terms and conditions of the potential sale and approves them. Third in line is the provision process, which sends out the parts contracted for, followed by the finance process, which sends out an invoice.

Let’s say that as the first process, for sales, is being written, the programmers treat every order as if it were placed in the company’s main location, even though the company has branches in several states and countries. That mistake, in turn, affects how tax is calculated, what kind of contract is issued, and so on.

The sooner the omission is detected and corrected, the better. It’s kind of like knitting a sweater. If you spot a missed stitch right after you make it, you can simply unravel a bit of yarn and move on. But if you don’t catch the mistake until the end, you may need to unravel the whole sweater just to redo that one stitch.

If the software coders don’t catch their omission until final system testing — or worse, until after the system has been rolled out — the costs incurred to correct the error will likely be many times greater than if they’d caught the mistake while they were still working on the initial sales process.

And unlike a missed stitch in a sweater, this problem is much harder to pinpoint; the programmers will see only that errors are appearing, and these might have several causes. Even after the original error is corrected, they’ll need to change other calculations and documentation and then retest every step.

In fact, studies have shown that software specialists spend about 40 - 50% of their time on avoidable rework rather than on what they call value-added work, which is basically work that’s done right the first time. Once a piece of software makes it into the field, the cost of fixing an error can be 100 times as high as it would have been during the development stage.

If errors abound, then rework can start to swamp a project, like a dinghy in a storm. What’s worse, attempts to fix an error often introduce new ones. It’s like you’re bailing out that dinghy, but you’re also creating leaks. If too many errors are produced, the cost and time needed to complete the system become so great that going on doesn’t make sense.

In the simplest terms, an IT project usually fails when the rework exceeds the value-added work that’s been budgeted for.

All of which leads us to the obvious question: why do so many errors occur?

**SOFTWARE PROJECT FAILURES** have a lot in common with airplane crashes. Just as pilots...
never intend to crash, software developers don't aim to fail. When a commercial plane crashes, investigators look at many factors, such as the weather, maintenance records, the pilot’s disposition and training, and cultural factors within the airline. Similarly, we need to look at the business environment, technical management, project management, and organizational culture to get to the roots of software failures.

Chief among the business factors are competition and the need to cut costs. Increasingly, senior managers expect IT departments to do more with less and do it faster than before; they view software projects not as investments but as pure costs that must be controlled.

Political exigencies can also wreak havoc on an IT project’s schedule, cost, and quality. When Denver International Airport attempted to roll out its automated baggage-handling system, state and local political leaders held the project to one unrealistic schedule after another. The failure to deliver the system on time delayed the 1995 opening of the airport (then the largest in the United States), which compounded the financial impact manyfold.

Even after the system was completed, it never worked reliably: it chewed up baggage, and the carts used to shuttle luggage around frequently derailed. Eventually, United Airlines, the airport’s main tenant, sued the system contractor, and the episode became a testament to the dangers of political expediency.

A lack of upper-management support can also damn an IT undertaking. This runs the gamut from failing to allocate enough money and manpower to not clearly establishing the IT project’s relationship to the organization’s business.

Frequently, IT project managers eager to get funded resort to a form of liar’s poker, overpromising what their project will do, how much it will cost, and when it will be completed. Many, if not most, software projects start off with budgets that are too small. When that happens, the developers have to make up for the shortfall somehow, typically by trying to increase productivity, reducing the scope of the effort, or taking risky shortcuts in the review and testing phases. These all increase the likelihood of error and, ultimately, failure.

AFTER CRASH INVESTIGATORS CONSIDER the weather as a factor in a plane crash, they look at the airplane itself. Was there something in the airplane’s design that caused the crash? Was it carrying too much weight?

In IT project failures, similar questions invariably come up regarding the project’s technical components: the hardware and software used to develop the system and the development practices themselves. Organizations are often seduced by the siren song of the technological imperative — the uncontrollable urge to use the latest technology in hopes of gaining a competitive edge. With technology changing fast and promising fantastic new capabilities, it is easy to succumb. But using immature or untested technology is a sure route to failure.

The IT debacle that brought down Fox-Meyer Drug a year earlier also stemmed from adopting a state-of-the-art resource-planning system and then pushing it beyond what it could feasibly do.

A project’s sheer size is a fountainhead of failure. Studies indicate that large-scale projects fail three to five times more often than small ones. The larger the project, the more complexity there is in both its static elements (the discrete pieces of software, hardware, and so on) and its dynamic elements (the couplings and interactions among hardware, software, and users; connections to other systems; and so on). Greater complexity increases the possibility of errors, because no one really understands all the interacting parts of the whole or has the ability to test them.

Sobering but true: it’s impossible to thoroughly test an IT system of any real size. Roger S. Pressman pointed out in his book Software Engineering, one of the classic texts in the field, that “exhaustive testing presents certain logistical problems... Even a small 100-line program with some nested paths and a single loop executing less than twenty times may require 10 to the power of 14 possible paths to be executed.” To test all of those 100 trillion paths, he noted, assuming each could be evaluated in a millisecond, would take 3,170 years.

All IT systems are intrinsically fragile. In a large brick building, you’d have to remove hundreds of strategically placed bricks to make a wall collapse. But in a 100,000-line software program, it takes only one or two bad lines to produce
major problems. In 1991, a portion of AT&T’s telephone network went out, leaving 12 million subscribers without service, all because of a single mistyped character in one line of code.

**THE PILOT’S ACTIONS JUST BEFORE** a plane crashes are always of great interest to investigators. That’s because the pilot is the ultimate decision-maker, responsible for the safe operation of the craft. Similarly, project managers play a crucial role in software projects and can be a major source of errors that lead to failure.

Back in 1986, the London Stock Exchange decided to automate its system for settling stock transactions. Seven years later, after spending $600 million, it scrapped the Taurus system’s development, not only because the design was excessively complex and cumbersome but also because the management of the project was, to use the word of one of its own senior managers, “delusional.” As investigations revealed, no one seemed to want to know the true status of the project, even as more and more problems appeared, deadlines were missed, and costs soared.

Bad decisions by project managers are probably the single greatest cause of software failures today. Poor technical management, by contrast, can lead to technical errors, but those can generally be isolated and fixed. However, a bad project management decision — such as hiring too few programmers or picking the wrong type of contract — can wreak havoc.

Project management decisions are often tricky precisely because they involve tradeoffs based on fuzzy or incomplete knowledge. Estimating how much an IT project will cost and how long it will take is as much art as science. The larger or more novel the project, the less accurate the estimates. It’s a running joke in the industry that IT project estimates are at best within 25% of their true value 75% of the time.

There are other ways that poor project management can hasten a software project’s demise. A study by the Project Management Institute, in Newton Square, PA, showed that risk management is the least practiced of all project management disciplines across all industry sectors, and nowhere is it more infrequently applied than in the IT industry. Without effective risk management, software developers have little insight into what may go wrong, why it may go wrong, and what can be done to eliminate or mitigate the risks. Nor is there a way to determine what risks are acceptable, in turn making project decisions regarding tradeoffs almost impossible.

Poor project management takes many other forms, including bad communication which creates an inhospitable atmosphere that increases turnover; not investing in staff training; and not reviewing the project’s progress at regular intervals. Any of these can help derail a software project.

**THE LAST AREA THAT INVESTIGATORS LOOK** into after a plane crash is the organizational environment. Does the airline have a strong safety culture, or does it emphasize meeting the flight schedule above all? In IT projects, an organization that values openness, honesty, communication, and collaboration is more apt to find and resolve mistakes early enough that rework doesn’t become overwhelming.

A recent report by the National Audit Office in the UK found numerous cases of government IT projects’ being recommended not to go forward yet continuing anyway. The UK even has a government department charged with preventing IT failures, but as the report noted, more than half of the agencies the department oversees routinely ignore its advice. I call this type of behavior irrational project escalation — the inability to stop a project.
even after it's obvious that the likelihood of success is rapidly approaching zero. Sadly, such behavior is in no way unique.

**IN THE FINAL ANALYSIS**, big software failures tend to resemble the worst conceivable airplane crash, where the pilot was inexperienced but exceedingly rash, flew into an ice storm in an untested aircraft, and worked for an airline that gave lip service to safety while cutting back on training and maintenance. If you read the investigator's report afterward, you'd be shaking your head and asking, “Wasn't such a crash inevitable?”

So, too, the reasons that software projects fail are well known and have been amply documented in countless articles, reports, and books. And yet, failures, near-failures, and plain old bad software continue to plague us, while practices known to avert mistakes are shunned. It would appear that getting quality software on time and within budget is not an urgent priority in most organizations.

Even organizations that get burned by bad software experiences seem unable or unwilling to learn from their mistakes. In a 2000 report, the U.S. Defense Science Board, an advisory body to the Department of Defense, noted that various studies commissioned by the DOD had made 134 recommendations for improving its software development, but only 21 of those recommendations had been acted on. The other 113 were still valid, the board noted, but were being ignored, even as the DOD complained about the poor state of defense software development!

Some organizations do care about software quality, as the experience of the software development firm Praxis High Integrity Systems, in Bath, England, proves. Praxis demands that its customers be committed to the project, not only financially, but as active participants in the IT system’s creation. The company also spends a tremendous amount of time understanding and defining the customer’s requirements, and it challenges customers to explain what they want and why. Before a single line of code is written, both the customer and Praxis agree on what is desired, what is feasible, and what risks are involved, given the available resources.

After that, Praxis applies a rigorous development approach that limits the number of errors. One of the great advantages of this model is that it filters out the many would-be clients unwilling to accept the responsibility of articulating their IT requirements and spending the time and money to implement them properly.

**SOME LEVEL OF SOFTWARE FAILURE** will always be with us. Indeed, we need true failures — as opposed to avoidable blunders — to keep making technical and economic progress. But too many of the failures that occur today are avoidable. And as our society comes to rely on IT systems that are ever larger, more integrated, and more expensive, the cost of failure may become disastrously high.

Like electricity, water, transportation, and other critical parts of our infrastructure, IT is fast becoming intrinsic to our daily existence. In a few decades, a large-scale IT failure will become more than just an expensive inconvenience: it will put our way of life at risk. In the absence of the kind of industry-wide changes that will mitigate software failures, how much of our future are we willing to gamble on these enormously costly and complex systems?

We already know how to do software well. It may finally be time to act on what we know.
A CONVERSATION WITH:
Tony Salvaggio, CEO of CAI

IT Risk Management

Extracted from an interview conducted by Michael Milutis

CAI: What is your perspective on Risk Management in the IT sector? How do you address this on individual projects and also, throughout the enterprise?

TONY SALVAGGIO: Risk Management is something that has real meaning for us. On any given day, CAI is managing approximately 2,500 IT professionals around the world who are charged with executing very specific software development and maintenance projects. In all of these initiatives, CAI is contractually bound, either partially or totally, to execute to specific targets, objectives, SLAs or deadlines. As a result, our organizational survival is dependent upon making sure that none of our people do anything that will get us sued or lose customers. I get frightened just thinking about it.

We have survived over the course of two decades by focusing very carefully on the implementation of Risk Management best practices. This is no small challenge given the size of our organization, the broad array of technologies and projects that we are working with at any given time, and the various cultural and regulatory environments within which we function, both domestically and internationally. It requires continual organizational learning along with the rigorous documentation and institutionalization of that learning. This is also tied to another principle, and a strong corollary; namely, how do we get our teams to continually perform close to their highest levels, on a daily basis, given our total organizational knowledge and experience over two decades?

CAI: Could you share with us some of the specific strategies that you have employed for addressing these challenges?

TONY SALVAGGIO: Please recognize that whatever strategies I share with you come from over two decades of continuous learning, experience, and evolution. As a result, how we function today is dramatically enhanced from how we functioned five years ago. And this will be true once more in another five years.

Anthony (Tony) Salvaggio is CEO and President of Computer Aid, Inc. (CAI), an international IT outsourcing firm that is currently managing active engagements with over one hundred Fortune 1000 companies and government agencies around the world. CAI employs over 2,500 associates across the United States, Europe, and Asia. Mr. Salvaggio founded CAI in 1981 and since then CAI has been leveraging the lessons of manufacturing in their development and maintenance of software. Prior to founding CAI, Mr. Salvaggio spent 22 years at IBM. In 2003, Mr. Salvaggio was a recipient of Ernst & Young’s “Entrepreneur of the Year” award. In 2004, he founded the IT Metrics and Productivity Institute.
Nevertheless, to answer your question, we have stayed focused over the years on several overarching strategies for mitigating IT risk. These strategies are primarily a response to the project failure statistics that have plagued the IT industry since its inception. Such failures represent a major risk for all IT and software organizations, but they also represent an enormous opportunity and by developing strategies that address this directly, we’ve been able to transform such risks into an area of competitive advantage.

First of all, and perhaps most importantly, it is almost impossible to recover from systemically poor estimation and bad scope management. Consequently, it is of the utmost importance to be continually building up the best possible estimation and scope management practices within the organization, given the total amount of organizational knowledge, past and present, that is available.

Consistent with this, estimation should never be the product of an individual person, something that is known in the industry as the “expert judgment” method. Moreover, estimates can be politicized, so any estimate should be validated and scrutinized by a second or third party, independent of the origin of the initial estimate.

You must also make sure that you are always tracking the estimate versus the actual, at a work component or module or project level, and you must have a method for feeding these actuals back into your estimation process, so that you can continually learn from experience and history and so that you are able to adapt on a daily basis. This data should also be very visible to all members of the project team.
and to management. And whatever the project size, a project plan must be updated with actuals and reported to all key stakeholders on at least a weekly basis.

We are also very big believers in work review processes and methodologies. Wherever possible we use a project office approach. Depending on the size and inherent risk of each project, this approach has evolved into various management and reporting systems. We believe that with projects that are only managed and reviewed by the project team and the project team's direct management, serious trouble may not be recognized until it is too late.

What ties all of this together? Time reporting. It’s not glamorous, but your estimates and project management techniques can only be as good as the data that goes in, and all of this ties back to how people are measuring themselves, what tasks they are working on, and how long they are taking to complete these tasks. Consequently, time tracking must be task based and it must be detailed, thorough, and immediate. By immediate, I mean “in real time.” Time and task data entered at the end of the work week is essentially meaningless and will not provide the same level of insight as real time task data.

Finally, we have learned that we will make mistakes. We will occasionally estimate a project badly or badly scope a project, especially when dealing with new technologies or unique projects. We treat these experiences and these lessons learned as “valuable
corporate assets” that must be stored in our corporate memory and lever-aged throughout the enterprise. This is clearly where we have done our best work over the years. Wherever possible, we have put what we have learned about estimating, tracking, reporting, planning, and management into real time software systems that institutionalize our collective learning and that can be leveraged throughout the enterprise. We use these systems continuously to manage our projects and our work and we insist that our aggregate organizational knowledge must be totally available to all projects and all individuals. Most recently, we completed the development of an “Automated Project Office,” which is essentially a software system that brings aggregate organizational knowledge and review processes to bear on every one of our activities and projects in a real time, internet based manner. This effort has been a five year, multi-million dollar project from conception through development and is now implemented within our business.

In short, I would say that we have developed a strategy for managing the risk of IT project failure that is rooted in the automation and institutionalization of organizational learning. This is really what it all comes down to. Darwin famously said that it is not the strongest of the species who survive, but the ones most responsive to change. I would counter that it is not necessarily the ones most responsive to change, but the ones most able to learn and to keep learning. Certainly learning is a form of change, but in our industry, it is the most critical and it will remain so as long as the technologies and projects we face in the 21st century continue to increase in scope and complexity and as long as the process of globalization continually forces us to find new and innovative ways to do more with less.

CAI is the creator of Automated Insight®, Tracer®, and the IT Metrics and Productivity Institute.

Automated Insight provides a convenient, proactive, and quantitative approach for enabling project governance and managing risks throughout project lifecycles.

Tracer is a process configuration management tool for IT organizations. Tracer defines and enforces processes that can vary based on the type of service event; it logs and tracks specific events, allows detailed tasks to be defined, and provides detailed time tracking. Additional capabilities include automated estimating, scope change management, and SLA management.

The IT Metrics and Productivity Institute is an international consortium of industry experts, fully funded and supported by CAI, and dedicated to the creation and dissemination of best practice standards in the IT and software industry. The Institute’s research focus areas include software development methodologies, software maintenance methodologies, software project management, software risk management, software estimation, and the science and practical application of software measurement.
TIM LISTER: The first one is technical risk. Most commonly this involves using a new product or tool for the first time. You know you're not an expert at it and you're trying to figure out whether the tool does what it says it does or whether we'll have to find something else because we don't know how to use it. We also have to estimate the cost of the learning curve. This issue of technical novelty is very common in our business. I like to joke with my clients about the fact that once you get really good at something, you never use it again. We finally master a tool and boom—the world changes and there are new toolsets, new ideas, and new messages that confront us.

The second big risk category is a set of organizational risks. Schedule risk is one of the most common and serious of these. With almost all projects, a deadline is set too early. From a risk point of view, the schedule and budget may be based on mere wishful thinking. I've got a project right now that's in Massachusetts, Ireland, and India—all at once! You want to have one architecture, one design, one product and you discover that there's a huge risk that these teams won't stay locked together. You have a lot of extra work to keep them coordinated.

The problem is always in the interfaces. Each team thinks they are doing fine but they may be out of whack with the others. Consequently, you have to do a lot of work to prevent large scale problems at the end. Then each team thinks it has its piece done and we discover that they just do not hook together. I guess I'm getting old but I feel it's nice occasionally to find a project where everyone's in the same building. Then you can talk to each other every day and brainstorm when problems arise. That is so rare now adds enormous complexity and increased chances for communication risk.

TIM LISTER: I think I'm going to surprise you. Practical tracking and planning is, of course, valuable. But the big issue is really cultural rather than quantitative or metric. There's not just Brownian motion going on out there. The fundamental issue is how well the people in your organization deal with straight talk. Too many organizations just have a hard time learning about what might go wrong before it does go wrong. In too many companies, it's dangerous to be frank and say, for example, "I think we are very unlikely to finish this new project in 10 months." People will say, "You haven't even tried, come on, give it a whirl; you're just trying to get out of work; you're whining." There's a very strong, "Rah, rah, we can do it" attitude, especially in top management. If you can't say what you believe without incrimination, you have a big problem. I remember talking to one organization about risks and having them tell me, "You know if you bring up a problem here, you own it." This happens on major performance issues where the boss typically says, "You're absolutely right. I want you to handle that." When that happens, no one is going to open his mouth. Instead he will think, "I'll act shocked and surprised when the problems hit because I'm not going to be the person responsible for a performance miracle with an underpowered system." So I think it's largely cultural. I am not a sociologist, but I think Americans have the hardest problem in the software industry with risk management. I think it is used more often and more effectively in Europe. Years ago I was in Finland and they were so good that I had nothing to say to them. They are much more frank about problems. It's the same in The Netherlands at Philips: the way they talk about their problems and make their decisions is very straight-forward, dispassionate, realistic. I wish we could bring that to the early stage of our projects as well. On the other hand, what makes America great is the way we do things because we don't know we can't do them.
As Lister notes, risk management is about understanding when to make decisions. Risk analysis usually deals with nebulous concepts such as probability of occurrence and the impact of failure. Failure to deliver the software “on time” creates a business risk, but so does delivering the software with defects. Given the statistical improbability of creating error-free software, engineers have to assess the cost and effort required to identify errors compared to the cost and impact that may result from undetected errors, delays, or missing functionality. The resulting cost-benefit analysis must be used to accept the risks or expend additional effort to mitigate the risks.

The art of good risk management involves naming or “nominating” a risk, assessing its probability, and defining/quantifying impact – and only then deciding what (if anything) to do about it. And it is infinitely better to name and choose to finesse a risk than to avoid it by ignoring it. As Charette says, “Bad decisions by project managers are probably the single greatest case of software failures today.”

In other words, no amount of testing, sign-offs, and traceability matrices can mitigate the risk of poor judgment. One of the most effective ways to improve a software environment is by establishing risk management within the development and support process – and modifying it, when necessary, based on the professional judgment of involved parties.

Computer Aid, Inc. (CAI) is a key player in risk management and process improvement. We provide tools and services that help technical managers gain visibility into their projects and provide clear insight into potential risk.

Contact CAI at (800) 691-5208, contact@compaid.com, or visit us online at www.compaid.com.