Managing Risk, an IT Manager's Responsibility

This brief essay was prompted by a realization that today's IT managers seem to pay more attention to their career path than ensuring that IT projects are valuable to the enterprise, work as intended when implemented, and can serve as a stepping stone for future projects. What passes for management is disappointing. I hope your experiences are better than mine.

Scope

- Comparing benefits with risks.
- Mitigating risks.
- Spending the enterprise's money wisely—maximize return on investment (ROI), realizing that return has more dimensions than money. Investment is of time and money, where time has both a monetary component and a strategic component (e.g., first to market).

Requires

- Complete enumeration of benefits and risks by business managers and IT engineers.
- Solid understanding of business objectives and constraints.
- Engineering staff able and willing to recognize benefits and risks and communicate them to others.

Process

The process of software development is more iterative than linear, hence the "cycle" in "software development life cycle."

- Organized into phases for requirements, design, development, verification (testing), and installation. These phases are mostly sequential, but can overlap to some degree. They are based on the following four policies: (1) you do not install software that you have not proved works as intended, (2) you cannot prove software works until it is developed, (3) you cannot develop software without a design, and (4) you need to know the software's functionality and infrastructure needs before you can design it.
- Has procedures, actitivites, tools, milestones, and deliverables.
- Is a mixture of left brain and right brain activities—analysis and visioning (ideation).
- Relies on communication to formalize current thinking, share progress (activities, findings, conclusions), and re-assess next steps. Communication is both oral and written.
- Costs money and takes time.
- The process used for a particular sofware project should be chosen deliberately to reflect the mitigation of relevant risks.
- A software development process is the means by which a manager can (1) introduce efficiency measures into the work and (2) provide a basis for tracking and evaluating progress.

Realistic Risks

- Business needs have changed substantially by the time the software is ready for installation.
- Actual benefits of completed software are a poor value when compared to its development costs. "I would never have done this had I known it would cost this much" is a typical complaint.
- Program code is held to be the end, not the means. The "rush to code" can obscure an incomplete analysis and design.
- Underlying IT infrastructure is insufficiently understood, which can lead to its misuse.

- Business and IT objectives for the software may be in conflict with each other.
- Stakeholders are dishonest.
- IT engineers are insufficiently skilled.

Mitigation Techniques and Their Dangers

- Foregoing formal documentation. This may save time up front, but inevitably it allows programmers to make business decisions, decisions that rarely if ever get recognized and communicated. Hence the odds for software failing verification increase.
- Considering a prototype to be the final product. Prototyping is a valid technique for verifying both functional requirements and technical details. It is typically done in a hurry. Consequently it is completely devoid of consistent error handling and a standardized architecture. It is neither reusable nor documented.
- Relying on commercial testing applications. These have their place, but they are unlikely by themselves to be capable of detemining if the underlying business objectives have been met.
- "Blind" adoption of a formal process. The standardized process that comes to mind is IBM's Rational; it is based on the five-phase model outlined above with additional interim phases intended to make the overlap and iteration more obvious. The danger with Rational is that too much time and effort is spent following the process. Agile, which can be seen as a response to the dangers of Rational, is more of a a methodology than an actual process; my assessment of it is that it relies on prototyping and lack of documentation in order to shorten the development time. The dangers of Agile are that (1) as for Rational, the process gets more attention than the problem the software is intended to solve, (2) the software is not reusable, and (3) the software cannot be extended without significant redesign. In both examples, "blind" adoption of a process usually means that deliberate risk mitigation is not done.
- Dividing an application into pieces that can be developed sequentially. This is a time-honored method. The biggest challenge is to design an architecture that can accommodate all the pieces while not making the first piece over long to develop. The danger is that the architecture for the first piece does not accommodate the later pieces, at which point the choice is to change the initial architecture to accommodate all the pieces or let the later pieces use a different architecture. In either case, the total elapsed time for the whole application may not be any less than if it was developed as a whole.
- Establish a documentation team with responsibilities for writing, maintaining, and publishing all project documentation. This neatly solves the dilemma of programmers who will not and/or cannot write. The team, which may be only one person, can work closely with the project manager to coordinate with deliverables and milestones and ways to measure them.