

# **Projects From Hell: The Ethics of IT Project Planning**

David H. Gleason  
Principal Consultant  
[ITforProgress.com](http://ITforProgress.com)

Presented at ETHICOMP98  
Erasmus University, Rotterdam, Netherlands  
March 25, 1998

© Copyright by  
DAVID H. GLEASON  
1998

## **Abstract:**

This paper investigates the ethics related to Information Technology (IT) project planning. It argues that many IT project failures can be avoided through informed, comprehensive and ethical planning methodologies. Poor planning leads to wasted resources, conflict and consequential damages. Effective planning can prevent these problems, but the information industry is young. In the absence of standards, IT professionals must seek effective methods. The paper outlines such methods. Case study summaries include the Denver Airport baggage handling software, a substance abuser case management system, a project management system, and the NASA Space Shuttle software systems. The paper provides a list of areas in which insufficient or ineffective planning can induce ethical problems. Analysis demonstrates several reasons why IT projects fail. Philosophical background is provided. Effective methodologies for IT project planning are discussed, followed by a five-phase planning methodology which breaks IT projects down into 1) definition; 2) needs analysis; 3) design; 4) development and 5) implementation. A structured set of questions is provided to help raise the ethical issues that should be applied during project planning.

## **1. Introduction:**

Information Technology (IT) project planning failures lead to ethical problems. Money and time are wasted, arguments erupt, consequential damages ensue. Many IT project failures can be avoided through informed, comprehensive and ethical planning methodologies. Effective project planning methods anticipate and avoid ethical problems.

There are two separate planning issues here. The first is systems specification, which must be executed carefully with attention to technical, user and management requirements. The second, which is the focus of this paper, is the IT project plan. To be successful, a project must be thought through with the same care as the system under development. Planning must address such issues as project management personnel, work breakdown, estimates and deadlines. Absent such planning, ethical issues will arise. Since organizations are built on a web of interdependencies, failures often have far-reaching implications.

The paper first explores the ethical problems that arise when IT projects are poorly planned. Industry statistics and a series of short case studies are presented to provide concrete examples of the issues. A practical analysis of incomplete project planning methods demonstrates some of the reasons why projects go awry. Philosophical background is provided and effective methods are discussed. The paper presents a methodology for project planning to avoid such problems, and a set of questions to help evaluate specific situations. Interdisciplinary perspectives include philosophical ethics, information science and business case studies.

## **2. Poor Planning**

American Business spent \$250 billion in 1995 on IT projects [Carroll, 1996]. According to W. Wyatt Gibbs, staff IT writer for *Scientific American*:

- One quarter of software projects are cancelled before they are completed
- The average software development project overshoots its schedule by half
- Some three quarters of all large systems are ‘operating failures’ that either do not function as intended or are not used at all [Gibbs, 1994]

Others suggest that the statistics are even worse. Chris Carroll, writing for *Fast Company* Magazine, says, “Only 16% of [IT] projects get completed on-time and on-budget, and nearly a third get cancelled outright” [Carroll, 1996]. These disappointing results, Gibbs notes, are in most cases the result of poor planning.

Poor planning leads to wasted resources: Systems fail to meet expectations resulting in ethical disputes and legal battles. Effective planning can prevent these problems, but since the information industry is young, has unrecognized professional codes and limited professional certification, there exist few widely accepted IT project planning standards. In the absence of such standards, IT professionals must seek effective methods. Not using such methods, or using weak methods, demonstrates professional incompetence, an ethical lapse that leads to poor decisions, failure to serve stakeholders and ultimately to unusable or faulty IT applications.

### **2.1. Case Studies**

The following case study summaries include the Denver Airport baggage handling software, a substance abuser case management system, an architecture project management system, and the NASA Space Shuttle software systems.

#### **Denver Airport Baggage Handling Software**

W. Wyatt Gibbs writes, “To veteran software developers, the Denver debacle is notable only for its visibility” [Gibbs, 1994]. The Denver International Airport (DIA) was originally scheduled to open in October, 1993 (then March, 1994, then May, 1994, then December, 1994). Delayed by the failure of its automated luggage handling system, it finally opened in February, 1995. The delay cost \$1 million per *day* in interest and operational expenses. This in addition to the \$193 million original cost for the system, plus \$50 million for an unplanned, traditional conveyor system.

An example of chaos theory (or Murphy’s law), there were too many unknowns in the project plan, and when it came time for implementation, the system failed to work. Blame was ultimately laid on power instability, software bugs and mechanical defects. By the time the airport opened, the delay costs were higher than the original projection for the entire baggage system. The take-home message is: examine complexity as a factor in making commitments to schedules and budgets.

## **Case Management System**

I once worked on a project for a federally funded counseling service. We were writing software that would track clients and report statistics back to the government. About halfway through the programming project, the counseling service lost its funding. However, we still had to complete the software in order to comply with the original grant, so we finished it. It operated for three months. Then, unlike a used car, it became completely worthless.

Less than a year after we started work our software was obsolete. Should the project ever be restarted the software would need to be substantially rewritten, because of changes in the IT industry. Moore's Law advises that computing power doubles every 18 - 24 months. In general, computers become obsolete in only three years — time enough for computing power to increase by a factor of four. Over five years computers become about ten times faster. Project planning must evolve to adapt to a changing industry.

## **Architecture Project Management System**

In the fall of 1996, a programmer I know took control of a project that had been under discussion for seven years. The system was to manage projects for a large architecture firm. By the winter of 1998, she had completed it within budget. It was very close to the client's specifications and time-line. She followed a structured approach, starting with planning, then designing and prototyping, and finally programming. She worked very closely with the client to show what she was doing every step of the way. She constantly sought input, judgment and client satisfaction. She was clear about when things would be done and kept those commitments. The endless feedback, the constant updates, and the interaction made her a superb resource to the client.

This project demonstrates the skills of the model programmer. The qualities to look for, aside from the normal human resources issues of organizational fit, are technical competence, sociability and the ability to manage time.

## **NASA Space Shuttle**

The people who write the software for the space shuttle have one of the best accuracy records in the industry. According to *Fast Company* magazine, "The last three versions of the program—each 420,000 lines long—had just one error each.... Commercial programs of equivalent complexity would have 5,000 errors" [Fishman, 1997]. Of course, if the onboard shuttle group makes mistakes, people die in a very public forum. It is therefore incumbent on them to get it right—the first time.

The team is about 260 people, divided between those who write the software and those who try to break it. There is intense competition between the two groups. They document everything: The change specifications to upgrade the software to use the Global Positioning Satellites were 2,500 pages long (*Global Positioning Satellites* form a web that can precisely locate an object on the planet or in orbit). The specs for the software overall are 40,000 pages long. Every line of the code is documented, not only with its function, but also with its history.

The group is one of only four in the world to receive the Software Engineering Institute's (SEI) "Level 5" (top) rating (SEI is part of the U.S. Federal Government). "According to SEI's

studies, nearly 70% of software organizations are stuck in the first two levels: Slightly better than chaos” [Fishman, 1997].

## **2.2. Why IT Projects Fail**

The most common problem with IT projects is underestimates, which not only incur unexpected costs, but can compromise organizational operation. This leads to questions of forthrightness: whether the planners were competent and ethical in their initial approach. Less frequent, but still pervasive, poorly structured planning processes produce systems with design flaws. Such flaws can compromise sensitive data, produce inaccurate results, undermine quality control and cause accidents. In the worst cases, life or death decisions may be based on erroneous information.

Analysis demonstrates several reasons why IT projects fail, including:

- Failure to consider planning at all
- Faulty planning methodologies, or misapplied methodologies
- Lack of thoroughness
- Inattention to stakeholders and their needs
- Poor estimating
- Unfamiliar or changing technology
- Lack of required technical skills
- Poor communication and teamwork
- Ineffective time management
- Inattention to the ethical implications of projects or the resulting systems

Failure in any one of these areas will lead to problems during a project. Failure in several areas will cause a project to fail. Each of these reasons for failure can be turned into questions to supply strategies for effective projects, e.g., “are we being thorough?”

## **3. Philosophical Background**

Four ethical frameworks that have roots in antiquity can provide a basis for project planning. While these philosophies are simple, their application leads to great complexity and depth. The first is the Golden Rule: “Do unto others as you would have them do unto you:” Endeavor to empathize with users, employees, others who are impacted by IT decisions. The second is Aristotle’s “Golden Mean:” Find balance among competing interests; make informed, rational decisions. The third comes from the Eastern traditions of Hinduism and Buddhism and suggests, like Aristotle, that avarice (greed for wealth, power or knowledge) enables short-term gain but, in the long run, is destructive to both individuals and organizations. The last framework is that of complexity: Effective decision-making procedures require ongoing attention to dynamic and complex technical and ethical issues.

The Golden Rule means systematically evaluating a project from the perspectives of stakeholders: shareholders, users, management, customers and vendors.

Aristotle's Golden Mean requires informed, rational judgment. Each part of a project must be reviewed carefully, with attention to details. Areas of insufficient planning must be identified and remedied. Cost-effectiveness should be evaluated. The reliability of the developers must be investigated. Decisions about whether time-lines are reasonable should be based on experience, not guesswork.

Eastern Philosophy teaches that avarice clouds the mind. Decisions based on avarice lead to failure because they are not based on knowledge. Effective project planning requires fine distinctions between the real and the imagined. If one imagines riches or fame instead of a successful project, one should expect cost overruns, missed deadlines and angry stakeholders.

IT projects are complex. Small design changes can have major ramifications. This complexity must be controlled. Chaos theory holds that small changes in initial conditions have major implications on outcomes. IT professionals must bring experience, training and a watchful eye to the seemingly insignificant details of their projects.

A kind of organic ethics stems from understanding the issues at hand and making informed decisions. The good is achieved by finding a balance between extremes. It is not possible to automate everything, and it is not sensible to automate nothing. Finding the middle ground requires skill, experience and attention.

#### **4. Effective Methods**

There are many methods in use for effective software development. However, programmers on smaller projects often use no method whatsoever. IT professionalism requires knowing not only the technology, but different methods for applying it in order to bring projects in on-time and within budget. Because there are no widely accepted standards in place, estimates for a project may range from \$5,000 to \$50,000, depending on the methodology. The prudent manager may doubt the validity of both such estimates.

An effective methodology for IT project planning has several components: stakeholder analysis, documentation of desired outcomes, establishing an effective project team, work breakdown, estimating, contingency planning, planning for change management and complexity analysis.

Business leaders must come to understand this process. Even those not directly responsible for implementation are nonetheless affected by it. Case studies demonstrate that poor technology implementations can undermine an entire organization. Done right, effective development improves efficiency, and saves money, time and even lives.

Efficient developers subsume IT technology piece by piece over time. Subsumption speeds projects by re-using code. As Wired Magazine's Kevin Kelly puts it, "When something works, don't mess with it; build on top of it" [ Kelly, 1994]. This has long been known in assembly-line manufacturing: Once a process is functioning well, leave it alone. Allow large processes to subsume smaller, reliable ones.

Building on experience is subtle in technology. A good programmer will build documented libraries of functions and share them with colleagues. An effective programming team will build

toolboxes that save time and money through reuse. For example, if all the members of a programming firm use the same log-in routine, time is saved in each application. Furthermore, as the routine is re-used and edited over time it becomes better suited to its purpose.

The use of technology subsumption requires forethought. Planners need to take account not only how subsumption can facilitate projects, but the effect that such systems will have when they are implemented. Managers must take into account the impact of the system on organizational process, management and staff.

As Simon Rogerson from De Montfort University and Terrell Ward Bynum from Southern Connecticut State University put it, “It is a common problem with information systems development projects that decisions concerned with, for example, feasibility, functionality and implementation do not take into account the requirements of all those affected by the system once it becomes operational.... The view is primarily ‘techno-economic’ rather than ‘techno-socio-economic.’” In order to account for the ethics of a situation, “concerns over, for example, de-skilling of jobs, redundancy, the break-up of social groupings can be aired at the earliest opportunity and the project goals adjusted if necessary” [Rogerson & Bynum, 1997]. If an organization accounts for these factors properly the chances of a successful implementation are greatly increased.

#### **4.1. Stakeholder Analysis**

Stakeholder analysis must represent not only all affected parties, but also the ethical impact the resulting system will have on them. Stakeholders include shareholders, management, users, vendors, and customers. A grid should be developed to show the effects that the project will have on each of them in the following categories:

- Cost (both time and money)
- Efficiency losses or gains
- Benefits and opportunities
- Training requirements
- Documentation requirements
- Policy changes
- Organizational restructuring
- Business process redesign

#### **4.2. Documentation**

Documentation of desired outcomes both imposes a discipline on the planning process, and provides clear criteria to determine project completion. A project should have specific performance measures. Etec Systems CEO Stephen Cooper says that to “operate on the leading edge of technology with very demanding customers... you have to execute on schedule” [Matson, 1997]. Milestones to which the project adheres are needed to raise red flags when deadlines are missed. Most of these milestones should be time-based. As Thornton May, Vice President of research and education at Cambridge Technology Partners puts it, “Time is the only unambiguous performance measure.... Business isn’t about return on investment, it’s about return on minutes” [Carroll, 1996]. Time management requires documentation.

### **4.3. Project Team**

A project team must include the right people and also establish their working relationships. The Project Team members should include at least:

- Project Manager: Heads the client team; oversees the project; serves as liaison to the developer
- Management Representative: An executive or manager who understands the role of the software within the client organization
- Technical Representative: A systems manager or other person who understands the client's system configuration
- User Representative: An end user

The Developer's team needs to include people with hard skills, such as: Systems analysis, systems development and documentation, and the soft skills of facilitation and training.

### **4.4. Work Breakdown**

Work breakdown must be detailed and thorough. Projects should usually be divided into sequential phases, which break down into smaller and smaller tasks until realistic time estimates can be made. Estimating must be thorough and agreed to by all stakeholders.

Work breakdown should include the project schedule. The client should expect implementation to be scheduled to accommodate the organization's needs, with the business impacts of conversion, training and documentation carefully considered. Staff should know what to expect, and have opportunities to influence outcomes, timing and usage. Implementation should be timed to coincide well with other organizational initiatives.

### **4.5. Contingency Planning**

Contingency planning is needed to address areas of potential project failure. Specific, written guidelines should be established for managing unforeseen circumstances. Areas of greatest potential failure should be identified and tracked carefully. Once the project is underway, red flags in any area should be heeded and addressed as quickly as possible. Project managers should analyze the potential effects of contingencies on the overall project and the client's organization.

### **4.6. Change Management**

There must be a process to manage project changes. It is not possible to foresee all eventualities. Therefore, the project must include several built-in opportunities to reassess the project and revise the plans. In addition to these built-in opportunities, the process should allow either the client or the developer to initiate changes to the scope of the project. A change of scope form should be used which includes at least:

- Client name; project name
- Requester's name

- Date
- Requested date for completion
- The name and date of the last agreement or contract signed
- A description of the requested change
- Signature lines

The form should go to the developer for an estimate of the cost and time required. Any impact on the project schedule should be noted. If the client approves the cost, they should sign the form, and the developer should also sign. Each party should keep a copy of the completed change agreement.

Failure to manage changes often leads to ethical problems. If a developer makes a change without prior approval, charging for it is questionable. If the client requests changes orally, the developer may not be able to meet expectations. Without careful documentation it is not possible to reach closure on a project, because the records are incomplete.

#### **4.7. Complexity Analysis**

Complexity is driven by the number of unknowns in a project, the number of IT function points, and the experience and knowledge of both the professionals and the customer. Project planning must also incorporate analysis of uncertainty. Professional consultants should be engaged to reduce unknowns to minimal levels.

The general formula  $D = C (G + J)$  may prove helpful, where D is the task duration given by C (complexity factor) times G (general experience factor) plus J (job knowledge factor). The experience and knowledge of resources allocated to the project have significant impact on task duration, given by the factors G and J. These factors are lower for experienced and knowledgeable resources [Rakos, 1990]. The complexity factor can be enhanced by estimating both the difficulty of the task and the uncertainty about how it is to be accomplished. Once the duration has been calculated for the entire project, risk analysis can determine feasibility.

### **5. A Five-Phase Methodology**

The following five-phase planning methodology breaks IT projects down into 1) definition; 2) needs analysis; 3) design; 4) development and 5) implementation. This is but one methodology among the many available.

#### **5.1. Phase 1: Definition**

The *project definition* (or project scope) includes:

- The overall objectives for the software.
- Selection of the members of the Project Team.
- A estimated cost for the whole project.
- The process by which needs will be analyzed and software will be designed, written and tested.

- A description of what the completed software will do.
- The plan for developing and implementing the software.

## **5.2. Phase 2: Needs Analysis**

*Needs Analysis* begins with an assessment of business processes. This phase presents an opportunity to improve and clarify business processes.

- What activities are to be automated and why?
- What related systems currently exist?
- Can the operation be streamlined by eliminating or consolidating steps?
- What outputs are required?
- What processes must be automated?
- What are the implementation requirements?

## **5.3. Phase 3: Design**

The starting point for *Design* is the Needs Analysis document, which is used to develop a system design that specifies exactly what the programmers must accomplish. This comprehensive statement spells out exactly what the software will do, how it will work, and what it will look like.

## **5.4. Phase 4: Development and Testing**

*Development and Testing* encompasses writing the software and making sure it works properly. The initial round of testing is the responsibility of the developers; the second round of testing (Beta testing) is done by the client, who determines whether software meets expectations. Documentation is developed as the software is completed.

## **5.5. Phase 5: Installation and Training**

During *Installation and Training* the final software is installed and all existing data is converted to the new system. This phase also includes training and the delivery of contracted documentation.

At the end of Phase 5, the client should sign acceptance of the final project.

## **6. Situation Analysis**

The following set of questions will help to help raise the ethical issues that should be applied during project planning.

1. What impact will the system have on stakeholders? Are project resources allocated appropriately?
2. Is there an effective and responsible project team? A competent project manager?
3. Is there buy-in from key people in the client organization?

4. What functions should the systems provide, and—equally important—how will those functions support the organization?
5. Do all proposed function serve organizational objectives?
6. Does the design specify the database structure, flows, triggers and processes? The design will serve as the criteria by which the systems will be considered completed, and by which to judge acceptability of the final product.
7. Is there a reasonable work breakdown, schedule and a budget with defined deliverables?
8. How will the project be managed? Both the client and the systems developers must actively manage the project.
9. Are contingency plans in place?
10. Has complexity been analyzed?
11. How will changes be addressed quickly and decisively?
12. Are the expectations of all parties clear?

## **7. Conclusion**

Successful IT project management requires effective methods professionally applied. In the absence of such processes, projects are bound to failure. This paper has argued that IT project failures are usually the result of poor planning in several, specific areas. Success requires at least: Stakeholder analysis, documentation of desired outcomes, establishment of an effective project team, work breakdown, estimating, scheduling, contingency planning, planning for change management and complexity analysis. Four ethical principles (the Golden Rule; the Golden Mean; avoiding avarice; complexity) can be applied to increase chances for success. A five-phase methodology was presented, along with a list of questions to assist in situational analysis.

IT Project failures result in the ethical problems of wasted resources, conflict and consequential damages. Successful projects require effective methods applied with skill, experience and attention.

## **Acknowledgments**

Thanks to Sonia and Christopher.

## **References**

- Carroll, C. (1996), Speed Kills (the competition),” *Fast Company*, August:September, 85, 88.
- Fishman, C. (1997), They Write the Right Stuff, *Fast Company*, December:January, 96-97.
- Gibbs, W. W. (1994), Software’s Chronic Crisis, *Scientific American* September, 87.
- Kelly, K. (1994), Out of Control, Addison-Wesley, 39.
- Matson, E. (1997), The Discipline of High-Tech Leaders, *Fast Company*, April:May, 36.
- Rakos, J. (1990), *Software Project Management*, Prentice Hall, 132-136.
- Rogerson, S. and Bynum, T.W. (1997), *Information Ethics: The Second Generation*, [www.cms.dmu.ac.uk/CCSR/ccsr/pubs/papers/ukaisabs.html](http://www.cms.dmu.ac.uk/CCSR/ccsr/pubs/papers/ukaisabs.html); INTERNET.