# Common Errors in Large Software Development Projects

David A. Gaitros
*Florida State University*

*During the past 40 years, numerous techniques for improving software reliability and efficiency have been developed. These techniques, when used properly, can contribute to the success of a major software development effort. However, despite these new techniques, software projects continue to fail. This article attempts to explain a few of the reasons why, despite advances in technology, a project can fall flat on its face, and what management can do to prevent these problems.*

Developing software is a relatively new area of enterprise that bears little resemblance to other engineering disciplines. Although the term software engineering is widely used throughout the business, the act of creating a new piece of software can hardly be compared to the design and construction of a new building or bridge. Computer scientists are still struggling after 30 years [1] to define *software engineering* and to find the right combination of techniques, procedures, and tools that assure success in development of large complex systems.

The closest comparison I can make of software engineering to another creative process is writing a book. Give two authors the exact same subject matter; they will almost certainly generate different works. Although the subject matter may be the same and perhaps the outcome of the book similar, other aspects such as number of pages, references, organization, writing styles, and even the number of chapters would be all different.

As a creative process, writing a book shares many things in common with writing software. [2] A great deal of research as well as planning is required. Knowing the outcome of the book or the desired effect is essential for success. The book, like software, will evolve over time going through many stages of development and modification. Software, like a book, once created need not be created again, just replicated for whoever wishes to use it.

Although development of software is a complex and mysterious process to most, there are some basic management techniques that you can apply that may greatly reduce the risk of failure. In this article, I provide some general guidelines that I have gathered over the years in dealing with large and complex software development projects that have remained constant over several decades. While many books and articles focus on techniques that you should use, they often ignore the fact that mistakes can still be made during a project that will cause it to fail if certain aspects are ignored. This article will focus on some of the more common errors I have observed over years of participation in large software development projects.

## Not Knowing What You Want or Need

It is not uncommon for companies and organizations to believe that the creation of a piece of software in itself will reduce costs and increase productivity. The more accurately the needs of a proposed system are defined the greater the chances of success. One of the most critical pieces of information is the proposed cost savings or benefits of the new system. Projects that involve hundreds of thousands to millions of dollars and several years in the making should not be undertaken unless there are clear and measurable objectives with provable benefits to the organization.

*“To avoid the perception of bias, the project manager should not originate from any of the specific groups but should be brought in from a neutral agency.”*

Information management (IM) or information technology (IT) can be very expensive to develop, purchase, and maintain. Management must not only consider the cost of the software but also the hardware, infrastructure, software maintenance, facility maintenance, additional technical support, recovery procedures, alternative processing, etc.

Introducing this technology into an organization is usually done for one or more of the following reasons:

1. The resulting technology will reduce manpower/labor requirements and reduce cost by automating what was once a manual process and prone to errors. In other words, the cost of developing the system in conjunction with yearly maintenance cost is offset by the reduction of the labor force.
2. The resulting technology will increase the productivity of the current manpower/labor force. The cost of developing the system in conjunction with the yearly maintenance fees is recouped through increased profits. Since federal, state, and local government agencies do not track profits, they would be looking for an increase in their ability to service the public by offsetting the cost of system development and maintenance with the cost savings in hiring additional labor to accomplish the same tasks.
3. The software would offer a capability that previously was not available. Early in the development of computers, scientists realized their potential to perform extensive calculations. It was not until computers became reliable and fast enough that certain mathematical problems could be solved.
4. The new IT and IM technology would increase the decision-making capabilities of upper level management. There are instances where more accurate, detailed, and properly manipulated data gives senior management better resources to make critical business or strategic decisions. The cost of development and maintenance of the system is offset by increased profit/productivity through improved decisions or through avoiding costly mistakes.

Managers must understand the intended purpose of the new system before investing time, money, and other resources in the development of a completely new product. Such endeavors are extremely expensive and there are many risks involved with developing a new piece of software. The spiral software model takes into account that as a project progresses, risks are assessed at each step to ensure the product being developed is not only of good quality but will be the right project [3].

## Fractured Development Teams

Often, a company or organization will involve several other organizations in the actual requirements analysis and product development. This is done to involve all those who have a stake in the outcome and to prevent alienating potential users. Several textbooks and journals I have read have never addressed this particular problem although it is more common than one might realize.

Modern software engineering techniques all focus on the assumption that the development will be accomplished by a single entity under the control of a strong management structure. The division of work and responsibilities among several organizations is often done for *political reasons*. Although one particular group may have the title of program manager, they may not have any direct authority over the other development teams. This presents several roadblocks to the successful development of a software product.

Without clear lines of communications and authority, the requirements and eventually the product development are accomplished in an inefficient fashion using different standards, different approaches, and sometimes even different technical standards. Even if all teams adhere to written standards, common interfaces, duplication of code, different interpretation of those standards, and other factors usually lead to integration problems late in the program development life cycle. The results are almost always unsatisfactory and at best it is inefficient.

One particular example typifies this problem. Several government agencies were tasked to develop a standard software package to be used by all branches of the federal government and military [4]. Although one branch was titled as the project lead, each branch that would use the end product was permitted to identify their own requirements, establish their own development teams, propose technical standards, identify existing modules to be used as an interim solution, and maintain their own cadre of contractors for support. After several years and several millions of dollars, very few lines of code had been successfully delivered.

Each different government agency perceived its requirements to be different from the other agencies, resulting in the very basic requirements of the system to be different. Different requirements dictate different specifications, which lead to different designs and even different implementation strategies. Unless great care is taken to ensure there are sufficient lines of communication among the different agencies, the products will almost always be incompatible.

To increase the chance of success, the development team should be geographically located in the same town, in the same building, and if at all possible, on the same floor [5]. Also, there must be clear and undisputed lines of authority even if some of the team members originate from other companies or groups. This clears up any ambiguity for both the development group and the customers who now have a single focal point for feedback. The customers identified with each of the groups must have equal say in the outcome of the new product, and great care should be taken to include representatives from each group in all critical phases of the project. To avoid the perception of bias, the project manager should not originate from any of the specific groups but should be brought in from a neutral agency.

*"Through years of experience and mistakes, I have discovered there are no such things as generic managers capable of managing any type of project."*

## Lack of Experienced and Capable Management

Organization and proper planning are absolutely essential elements of any project. You would not think of hiring an experienced chef to run an automobile assembly line or likewise a software development project. Through years of experience and mistakes, I have discovered there are no such things as generic managers capable of managing any type of project. Other large software development firms have learned the same lesson.

Although the customer calls the shots as far as functionality, cost, and schedule, the person in charge of the actual software production should be an experienced data automation manager with technical knowledge in the area being developed. Hire someone with experience in dealing with the kind of system you are trying to develop. The following are some reasons for hiring such a person:

1. IT managers must be competent enough to identify and hire a qualified work force.
2. Unqualified managers too often must rely on less experienced technical staff or, worse yet, vendors on project decisions.
3. Managers must be able to assess accurately the capabilities of other shops and sub-contractors assigned to the project. Shop managers or contractors who hire inexperienced or the wrong type of personnel will find out late in the project that they must hire expensive consultants to make up the differences. An experienced manager will have a good idea what capabilities are required to complete an assignment and be aware of the risks of peer organizations that hire sub-standard personnel.
4. The manager is usually the face-to-face customer contact. He or she must converse intelligently on the impacts of alternatives and be able to address technical questions without drawing from other staff members. Customers must have confidence in management.
5. Decisions on cost, schedule, and performance cannot be delegated and must be made based upon a combination of education, experience, metrics, and the interpretation of those metrics.

## Lack of Proper Work Environment

Imagine a hospital with a staff of highly trained and skilled surgeons that lacked proper operating rooms, instruments, and other medical staff for support. Operating on a patient would be hazardous at best. Likewise, software development organizations will sometimes hire very skilled software engineers and purchase expensive computer-aided software engineering (CASE) tools but ignore establishing the right development environment needed to ensure a successful project.

Having modern automated software products in conjunction with proper organization, a skilled support staff, and the right work environment can greatly increase overall productivity and improve software quality at the same time. Organizing your staff is just as important as the tools they use. Here are a few hints on setting up the proper environment:

1. Be sure to separate your development staff from computer operations. You do not want your programmers and analysts spending their time troubleshooting networks, installing servers, maintaining a database environment, creating Web pages, etc. They should spend their time using the services of the system to satisfy customer requirements. It is well worth the money to hire a few dedicated help-desk/operations staff to alleviate the burden from the rest of the workers. Part time or on-call services usually benefit smaller projects by not requiring full time employees.
2. Do not purchase top-of-the-line equipment for the development phase of your project. If software is designed on state-of-the-art equipment, chances are it will only work properly in that environment. Developers/coders/analysts should try to develop systems that run efficiently and will operate across a broad spectrum of capabilities using the worst-case scenario for standard testing and development. Provide computer programmers with the minimum configuration you would expect most of the users to possess. Be sure to test it on all target platforms, memory ranges, and operating systems.
3. Set up a separate configuration and control group that keeps close tabs on quality control and version maintenance. Individual programmers usually do a poor job of policing themselves. It is equally important that the configuration and control group not be managed by anyone in the development chain but report directly to the IT manager. This will avoid any attempts by development staff to fudge on schedules and skip important reviews and test cycles.
4. Although your hardware should be minimal in nature, do not scrimp on sophisticated tools or software engineering environments. Automatic code generators, development environments, and extensive library setups with automated configuration and control tools increase productivity tremendously. Their only drawback is the extensive amount of training required for their use.

## Inexperienced or Mediocre Technical Staff

The last thing a manager needs on a project that requires creativity and ingenuity is mediocre people. A few highly experienced and educated developers can outproduce any number of average programmers. A painful lesson that I keep learning repeatedly is to hire top-notch people and empower them to do the job. Here are some hints on hiring good people:

1. Be very specific on technical experience and education when advertising for empty positions. If you need experience in specific languages, operating systems, machines, and networks with specific versions then specify them on your advertisement and do not accept less. A common mistake some companies make is to advertise for an employee with a wide range of experience never really expecting to find such a person but rather to find someone as close as possible.
2. Confirm all applicable classes/schools attended along with grades, degrees awarded, companies worked for, as well as contacting all references. Do not hire someone who has lied on his or her resumé.
3. Test the applicant. If the applicant claims to be an experienced C++ programmer, give him or her a test. Have him or her write a small complex program in the environment they claim to have experience in. You will be surprised how few are capable of passing these kinds of tests.
4. Technology changes rapidly. Look for experienced developers who continue to update their skills. A good, prospective employee will keep up with technology and be willing to prove it.

*“Set up a separate configuration and control group that keeps close tabs on quality control and version maintenance. Individual programmers usually do a poor job of policing themselves.”*

## Getting a Slow Start

The easiest place to make up for lost time is during the beginning of the project. Too often, slow and methodical project starts result in a panic race near the end to finish and deliver the software. All too often, this results in a shoddy product. This is where experienced management will be the most visible and aggressive. The first half of the project should not be devoid of meaningful deliverables. Too often, high-level models, minutes of meetings, funding expenditures rates, and personnel reports take up most of the first half of a project.

An experienced manager should be looking for results of requirements analysis, architecture plans, development strategies and schedules, technical impacts, preliminary database design, detailed models, timing diagrams for real time systems, etc. These time-consuming and important efforts cannot be put off until the latter half of the project.

## Communication Requirements

Some of the most non-productive time spent on a project can be meetings. Take a tip from professional business process engineers on conducting meetings. First and foremost, each meeting must have meaning other then just a routine schedule [6]. Always have a published agenda with the specific meeting purpose stated. If upper-level management or the contract requires routine meetings, they should follow these simple guidelines:

- Have a fixed agenda and purpose.
- Always appoint someone in charge that can minimize extraneous talk.
- Track and publish decisions made during the meeting and deliver to appropriate parties.
- Adjourn the meeting when business has been conducted.
- Do not go beyond the scope of the meeting’s original intent. Rather, you should conduct a smaller meeting with only the interested parties to avoid taking other individuals’ time.

When coordinating project technical requirements, minimize the number of people in the meeting to essential personnel only. Make sure the people attending can satisfy all the requirements that arise during the meeting. Nothing is more frustrating than attending a meeting where key technical personnel were replaced by the next level of management who were unable to discuss the project details thus requiring another meeting. If key personnel cannot attend, postpone the meeting.

There is a huge debate going on in the industry concerning using e-mail and the Internet in the office [7]. Proponents of e-mail claim it improves communication within an organization by allowing people to transfer information anytime during the day or night ensuring that the person gets the information. Opponents claim it wastes time and productivity through abuse of the technology. Companies have restricted some office personnel to receive internal e-mails only while reserving external e-mail privileges for individuals that have bona fide requirements to do so. My recommendation is to allow e-mail throughout the organization since I believe the benefits outweigh the bad. The Internet can be a very useful tool, but it can also be a costly waste of time in both manpower and network bandwidth. Since this is primarily a research tool, I would limit access to those individuals who truly need it.

## Relying Only on User Interviews for Requirements Definition

A good computer systems analyst will extensively explore other avenues of information on gathering requirements. A common error in most failed projects is using only a selected number of individuals from the user community to define accurately functional requirements. It is rare to find individuals that possess all or most of the knowledge required to develop fully a complete information technology software system. On the contrary, corporate knowledge is usually spread among several individuals and groups. Many times, crucial business processes are contained in notebooks, briefcases, a PC spreadsheet, or in a person's desk. Here are some recommendations to help resolve this:

1. Obtain copies of any legacy software system that is being used. This will give some insight into documented business practices.
2. Obtain and thoroughly read all operations manuals, pamphlets, brochures, regulations, or laws that pertain to the organization.
3. Hire your own full-time *functional experts* to assist in filling any gaps in users' requirements. Recently retired individuals are particularly useful. Companies will seldom allocate an employee full time to your development team.
4. Perform as many of the above functions as possible before your first official meeting with the customers. You will find the advanced preparation will make your first and subsequent meetings very productive.

## Not Involving the User at All Stages

The most elaborate and efficient system in the world is doomed to failure if the targeted users are not enthusiastic about its arrival. A good deal of public relations work should be done at all stages of the project to ensure that users eagerly anticipate the new system. The best way to accomplish this feat is to involve as many of the users as possible in the software development stages. It is absolutely necessary to make a little bit of time and effort during each stage of the project to keep the user community informed and obtain their feedback.

*"A common mistake some companies make is to advertise for an employee with a wide range of experience never really expecting to find such a person but rather to find someone as close as possible."*

## Cutting Short Testing

The computer industry calls its mistakes *bugs*. Rarely does a software package make it to market without bugs, which are a result of three basic acts: (1) an omitted or improperly stated requirement, (2) errors in computer logic, and (3) a performance or timing error usually associated with hardware or networks [6]. A good software development organization will have strict testing requirements to discover as many defects as possible. Each test should examine utility, correctness, robustness, and performance. Here are the usual definitions of the different testing cycles [1]:

- **Programmer Test:** Individual programmers test their specific modules against the specifications they were given. Interfaces to external modules or systems are usually simulated.
- **Software Inspections:** No one likes to have their work criticized in an open forum, but scientists have always been accustomed to having their work reviewed. Software inspections come under the category of *fault-avoidance* techniques. Code reviews, software inspections, and walkthroughs have proven to be very effective in detecting errors [6]. Experiments have shown that up to 85 percent of software faults have been detected by such techniques [8].
- **CASE Tools:** Although we mentioned CASE tools before, they can also be used to enforce programming standards. Capability Maturity Model® Level 3, 4, and 5 organizations often will introduce strict coding standards into their projects to prevent common programming mistakes.
- **Configuration and Control Test:** The configuration and control group tests whether the delivered modules meet specific programming standards set down by the company and whether the software specifications match the documentation.
- **Alpha Test:** This test is usually done by the company either at their site or at a user's location. The following features are tested: (1) deployment features, (2) interfaces to other systems, (3) upload/download procedures, (4) requirements compared against design specifications, (5) basic user functionality, and (6) performance.
- **Beta Test:** This is the first truly operational test where the new system replaces the legacy or manual system in its entirety at limited locations or sites. This is not only an extension of the Alpha Test but also a measure of whether the initial feasibility studies on the usefulness of the new system were accurate. These test results are used in determining if the new system is acceptable to the user.
- **Extended Beta Test:** After corrections or modifications on the Beta Test, the new software is released to a larger community of users before organizational or worldwide deployment. Tests should endure past several milestones such as end-of-day, end-of-week, end-of-pay cycle, calendar date rollovers, etc. to ensure adequate numbers of scenarios are tested.

## Conclusion

Other engineering disciplines have the benefit of many decades if not centuries of refinement on their processes. The building of software is still in its infancy and may take several years to fully mature; we can help it along by practicing a few basic management techniques that have proven successful.

There have been many attempts by individuals to identify which step of the development process is the most critical. Some say the early stages of requirements definition, others say testing, while some conclude it is the actual development itself. I believe that all stages are critical to the success of any project. Failure can and does occur at any stage while success can be claimed only after the completion of all phases and successful product delivery to the customer. Software development is extraordinarily tedious and time consuming; minute details have been known to bring project personnel to their knees. A successful project will have top-notch management, a healthy work environment, expert technical staff dedicated to specific tasks, clear lines of communication and authority, involvement of the user community, high standards, modern development tools, and clear goals.◆

## References

1. Sommerville, Ian. Software Engineering. 6th ed. Addison-Wesley, 2002: Chap. 1.
2. Hamming, R. "Mathematics on a Distant Planet." Invited Talk, 1996.
3. Boehm, B. "A Spiral Model of Software Development and Enhancement." Software Engineering Project Management, 1987: 128-142.
4. Office of the Inspector General. "Audit Defense Environmental Security Corporate Information Management Program." Project No. D2000AS-0207.000. 7 Dec. 2000.
5. Jensen, R.W. "Lessons Learned From Another Failed Software Contract." CrossTalk Sept. 2003: 25-27.
6. Bruegge, B., and A.H. Dutoit. Object-Oriented Software Engineering: Conquering Complex and Changing Systems. Prentice Hall, 28 Oct. 1999.
7. Coetzer, Dudly. "Cut Unnecessary Communication Costs." Accountancy SA Oct. 2003 <www.accountancysa.org.za/archives/2002oct/features/Limiting.htm>.
8. Fagan, M.E.. "Design and Code Inspections to Reduce Errors in Program Development." IBMB System Journal 15.3 (1976): 182-211.

## About the Author

**David A. Gaitros** is the associate chair of the Computer Science Department at Florida State University, Tallahassee, Fla. He spent 22 years in the U.S. Air Force as a software developer and manager of large software projects as well as the associate chair for the Naval Postgraduate School, Monterey, Calif. He has worked on the Airborne Warning and Control System, the Air Force Data Systems Design Center, and various other development projects for Air Force Material Command (former Systems Command) and the Air Force Civil Engineer.

**Florida State University**
**Department of Computer Science**
**261 James J. Love BLDG**
**Mail Code 4530**
**Tallahassee FL, 32306-4530**
**Phone: (850) 644-4055**
**Fax: (850) 644-0058**
**E-mail: gaitrosd@cs.fsu.edu**