When software projects exhibit warning signs like chronic delays, missed milestones, and cost overruns, senior managers often turn to third-party experts for unbiased, comprehensive project audits. Impartial auditors deliver results that are credible to all stakeholders (including investors), an advantage when it comes to building consensus on a corrective strategy for a troubled project. This white paper describes our checklist-based, collaborative audit process that effectively diagnoses failing software projects, and sets them back on the road to recovery.
Introduction

When asked “Where do you think your software project will be in six months?”, few of us are likely to answer “Behind schedule, over budget, and underperforming.”

But according to the Standish Group’s Chaos Summary 2009 report, 44% of software projects are “late, over budget and/or with less than the required features and functions”, and 24% of software projects fail, meaning they are “cancelled prior to completion or delivered and never used”.¹

While it’s easy to look back at a failed software project and trace its downward spiral, an active project is more difficult to assess. Fortunately, a project doesn’t fail overnight. Serious problems are usually preceded (perhaps months in advance) by warning signs such as:

• Project delays
• Missed milestones
• Cost overruns
• Turnover
• Spotty documentation
• No progress reports
• Tendency to deny that problems exist
• Fear
• Lots of bugs
• Regression (i.e. fragile code)

Figure 1: Software Project Outcomes

Even if the Standish numbers are a little pessimistic—remember, their results are based on specific definitions of terms and parameters—the study at least suggests that software projects continue to pose a significant challenge for organizations of all shapes and sizes. The question is, why?
As technology advances, software projects tend to become more complex, which makes them a moving target for development teams. Greater complexity means that planning must be more sophisticated, and execution more disciplined. With so much going on, the success of a software project often depends on the stakeholders’ ability to collaborate with one another. For this reason, team cohesion can be a determining factor in a project’s outcome.

Software projects usually fail for “people” reasons rather than technological reasons. Consequently, when warning signs appear, the root cause—or more often, root causes—may remain elusive, frustrating a manager’s best efforts to define them. As a project deteriorates, a manager’s ability to fix it can be impaired by:

- **Lack of experience with similar failures**
  Given that software projects are continually growing in complexity, it’s unlikely that even an experienced manager has “seen it all”.

- **Exhausted political capital**
  Stakeholders no longer have confidence in any of the manager’s diagnoses or solutions, regardless of how convincing they are, how competent the manager is, or where the problem actually lies.

- **Absence of a troubleshooting game plan**
  Problems will remain undiagnosed and unaddressed without an unbiased, systematic analysis of the entire project.

Indeed, an errant software project cannot be successfully rehabilitated until the development team understands just how far down the rabbit hole they’ve gone with it. In other words, they need a *software reality check*.

This white paper describes the key benefits and characteristics of a comprehensive diagnostic approach that provides a reality check for software projects in crisis.

### 10 Possible Sources of Project Failure

Software projects don’t just suddenly fail. Warning signs appear and grow in severity as the negative effects of underlying problems propagate.

Here are 10 potential contributors to the ill health of a project:

1. Unrealistic schedule
2. Insufficient budget
3. Poorly defined requirements
4. Poor technology choices
5. Unfocused business strategy
6. Poorly defined project scope
7. Lack of formalized processes
8. Poor adherence to standards
9. Lack of expertise and experience
10. Poor team chemistry (e.g. low morale, office politics)
Fortunately, “failing” does not have to mean “failed”. According to a 2005 survey by the Cutter Consortium, “45% of software development organizations almost always manage to get seriously troubled projects back on track…30% say that they manage to do so sometimes…” Getting a software project back on track requires substantial effort, and to succeed, that effort must begin with an unflinching assessment of the project.

**Start Fresh with an Unbiased Third-party Audit**

An effective software project audit is administered by unbiased, experienced third-party specialists. To be credible, their review must be:

- Objective
- Structured
- Quantifiable
- Verifiable
- Solution-oriented
- Conclusive

The audit must comprehensively evaluate the software’s architecture, code, and development team—including the development process—in a timely manner without significantly disrupting the client’s operations. The audit process has two phases, namely:

- **Information Gathering**
  In a brief onsite visit (no more than a day or two), auditors gather information about the project in a courteous, constructive manner.

- **Project Analysis**
  Offsite, auditors study and evaluate the detailed project information that they’ve gathered.

**Key Benefits**

For senior managers who need to make informed decisions about failing projects, an unbiased audit by third-party specialists has several benefits, including:

- **Credibility**
  Make it easy for stakeholders to achieve consensus on a project by presenting an impartial audit report from recognized experts.

- **Visibility on “showstoppers”**
  Make go/no go decisions with confidence, backed by a credible, comprehensive analysis of the project.

- **Cost savings**
  Use the detailed audit report to precisely target future project spending in areas where it will be most effective.
• **Opportunity to demonstrate leadership**
  Take the initiative, get to the heart of the problem, and get on with fixing it. No-nonsense problem solvers earn respect, and with good reason: they get results.

**Case Study: MyScreen Mobile**

In anticipation of an important meeting with overseas investors, MyScreen Mobile asked us for a software reality check of its new mobile advertising platform. Our detailed report had to be completed prior to the meeting, which was in two weeks.

The project audit uncovered serious defects in MyScreen’s new software. Most notably, security was badly flawed and the new platform would not scale to the projected traffic volume. To illustrate our findings and prove their veracity, we built a simple test that demonstrated the identified shortcomings.

At MyScreen’s request, we developed a full architectural strategy (a roadmap) for creating a platform that would meet their requirements, which were:

- Massive scalability
- Excellent performance
- High security
- Extensibility
- Carrier-grade system

We completed the roadmap before the crucial meeting.

At the meeting, the investors expressed concern about flaws in the software, and were clearly prepared to pull their support from the project. However, our audit report and platform roadmap reassured them that MyScreen understood the problems and was actively working to rectify them.

MyScreen Mobile retained its investors, who were impressed by the new strategic plan and the proactive, problem-solving approach that it represented.

**Software Reality Check**

Our software reality check offers a bird’s eye-view of the project labyrinth. Using a carefully designed checklist, we create a panoramic view of the project that includes all key aspects:

- Company (as a whole)
- Development team
- Project documentation
- Software

Loosely based on the 4+1 View Model of Software Architecture, our checklist is designed to elicit a detailed description of a software project, and then
place that description in a broader context by comparing and contrasting with known or logical alternatives. In essence, the checklist asks:

- What was done?
- What was used to do it?
- How was it done?
- How could it be done better?
- Were best practices followed?
- Are there any showstoppers?

Figure 2: 4+1 View Model of Software Architecture

The checklist consists of three sections, which separately review the software architecture, code, and development team.

Architecture Review

This section examines the big picture for signs of weakness using quality metrics such as security, scalability, testability, performance, extensibility, flexibility, robustness, and maintainability.

- **Understand the business**
  Understand the essentials of the mission statement, strategy, philosophy, core values, and core competencies. These business basics provide important context for analyzing the technical aspects of the software project.
- **Examine business systems and processes**  
  Describe current systems and processes, and define the major “hows” of the business. (e.g., how the business creates value)

- **Assess the technology**  
  Measure the effectiveness of the chosen technology. Determine if the technology is appropriate to the application and justify that determination with concrete, credible evidence.

- **Survey the infrastructure**  
  Take stock of existing infrastructure and assess whether or not it is adequate.

- **Analyze data formats and models**  
  Compare data formats and models with requirements and evaluate their effectiveness.

**Code Review**

This section looks at the nuts and bolts of the software to determine quality, professionalism, adherence to standards, and use of best practices.

- **Assess code quality**  
  Determine if the code is up to professional standards with respect to design, efficiency, best practices, and style.

- **Assess effectiveness of code structure**  
  Determine if the code structure is optimal for the intended application and if it would benefit from structural changes.

- **Scrutinize adherence to standards**  
  Describe how standards are or are not adhered to, and explain how this impacts the project.

- **Evaluate open source versus proprietary technologies**  
  Determine if open source technology is being employed to best advantage.

- **Evaluate frameworks, packages, other 3rd party software**  
  Assess the effectiveness of the software tools being used.

**Team Review**

This section focuses on how well the project was implemented.

1. **Analyze the software development process**  
   Assess how the software was designed and coded. Identify areas for improvement.

2. **Evaluate adherence to software engineering best practices**  
   Comment on how best practices were an asset or how their absence was a liability, and what impact this had on the project. Identify areas for improvement.

3. **Assess the quality of project artifacts**  
   Determine the quality, completeness, professionalism, appropriateness, and effectiveness of the project’s documentation.

---

*The bitterness of poor quality remains long after the sweetness of meeting the schedule has been forgotten.*  
- Anonymous

*Creating a Software Engineering Culture* by Karl Wiegers

*There are thousands of ways to mess up or damage software projects, and only a few ways to do them well.*  
- Capers Jones

*Applied Software Measurement*
References


**Architech Solutions** is a Toronto-based technology consulting and software development firm. We design and build powerful, user-centred, strategic software solutions that transform businesses. We’re agile, disciplined and passionate about delivering for our clients.

To book a free on-site Discovery Workshop led by our team of consultants, e-mail [info@architech.ca](mailto:info@architech.ca) or visit us at [www.architech.ca](http://www.architech.ca)