



## **Crystal Seed, LLC offers services for achieving high quality software with near zero failure.**

The US Department of Commerce's, National Institute of Standards and Technology's study in 2002 reported: "software bugs, or errors, are so prevalent and so detrimental that they cost the U.S. economy an estimated \$59.5 billion annually".

Type "Software Glitch" into a search engine and you will quickly see the detriments caused by software failures. Remember this is a tiny fraction of total software failures.

Using total cost of ownership analysis, a company determined that one poor quality software cost them an estimated \$2,000,000 over time where a high quality one would have cost only \$100,000.

Poor quality software creates unnecessary downtime for users and increases support cost. Additional, incalculable cost of poor quality software is the impact to the delivery of services to an organization's clients.

Organizations contribute to poor software quality because they ignore key steps in the software development process and pursue fixed target timelines or a predetermine release date.

## **How would you answer the following two questions?**

To demonstrate confidence in your team, how much of your annual salary would you be willing to lose if a new application developed by your team had an avoidable problem within a year, 5%, 20%, or 100%? 100% means you have so much confidence that you will give away your entire salary.

Any answer besides 0% is commendable confidence given the prevalence of software failure today.

### **Second question:**

If you scrambled the coding assignments of your team, assuming they are all coding in the same language, how long will it take the team to get back to pre scramble productivity, one day, one week, one month or three months?

It will take a typical team between one to three months to return to pre scramble productivity depending on the number of developers. I have seen developers start from scratch to write a new application instead of modifying an existing application because it was too much trouble to understand the existing code. In addition, I have seen developers struggle to maintain their own code a year later because it was poorly written.

**What is the solution to these issues? One of our services is professional code review. Notice I said professional code review, not peer code review.**

- ✓ Professional code review alone will boost a team manager's confidence in his team.
- ✓ Professional code review will allow team members to fill in for each other within minutes.
- ✓ Professional code review will allow new team members to be productive and up to speed within a few days, not weeks or months.
- ✓ Professional code review will allow maintenance/support teams to resolve issues in minutes, thereby reducing technical support cost to negligible amounts.
- ✓ If code review is integrated into the development process early, it will reduce total development time due to fewer bugs and fewer round trips between quality assurance testing and remediation.
- ✓ Professional code review is ongoing, uses standard reports and does not require developers to stop for unnecessary meetings.

**Professionally reviewed code has these characteristics:**

- ✓ Significantly exceeds minimum standards.
- ✓ Contains long-term view, flexibility and robustness for maintainability.
- ✓ Has clarity in every way from backend to user interface.
- ✓ Error messages and warnings are clear and useful.
- ✓ Has near zero failure.

**The alternative to professional code review is peer code review:**

Peer code review is better than no review at all. Unfortunately, peer code review is too often omitted or done poorly because of the following reasons:

1. Code review results in long-term gains only while developers typically focus on the short-term goal of delivering software on time.
2. To meet deadlines, the first items eliminated are intangibles that contribute to quality such as code review but are not readily observable by development managers.
3. Until now, there were no real guidelines for reviewing code. Some teams make up theirs.
4. Peer reviewers are time pressed to complete their development assignments. As a result, code reviews are performed hastily and poorly.
5. Some teams review code every 3-4 months when too much code has been written and the effort to remediate becomes overwhelming.
6. Co-workers may be sensitive to each other and unwilling to point out serious flaws in each other's code.
7. Developers review peer's code based on personal standard and style, which may itself be flawed.

## **Why perform code review in addition to quality assurance testing?**

By the time most software reaches quality assurance testing, thousands or even millions of lines of code would have been written. If the code contains repetitions, quality assurance testing can only catch the areas tested, leaving wrong or poorly written code un-remedied.

Most quality assurance tests focus primarily on specific business requirements and very rarely test the main factors that plague most software.

Passing quality assurance test does not guarantee that software is good, hence, the high prevalence of bug-full software.

Quality assurance testing is only equivalent to reading the table of content of a book to make sure that all needed topics have been covered.

## **Crystal Seed's Code Review Process:**

Every line of code needs to be professionally reviewed, even those written by very experienced quality conscientious developers. Imagine for a minute the quality of books if authors did not submit their books to editors and proofreaders before going to press. Authors who have published over 50 books still use reviewers. The same goes for research scientists.

All Crystal Seed's consultants are trained to become professional code reviewers, not developers using their individual styles. As a result, all code is reviewed to topnotch professional standards.

We treat the developers with great respect, understanding that their mandate is to write code to business rules, while ours is to look for what can go wrong and how to avoid it or minimize the impact.

We start with the architecture of the application and review end to end.

We review database designs for clarity (objects and scripts), and flexibility. For example, compare these error messages from a database and see which would result in a quicker resolution.

1. ORA-02291: integrity constraint (SYS\_C002359) violated.
2. ORA-02291: integrity constraint (CreditCard\_In\_CustomerOrders\_FK) violated.

We review user interface for clarity, input validation and user experience optimization. Users could perceive even an optimized application as slow due to poor user interface design.

We review middle ware and business logic for clarity, maintainability, existence and appropriateness of messages. The following common software error messages are useless:

1. File not found.
2. Fatal error.
3. The system has encountered an error.

We review UNIX shell scripts and other scripting environments such as JavaScript for clarity and maintainability.


### **Crystal Seed's Code Review Justification:**

By now, we believe you are convinced about the merits of code review and in particular, Crystal Seed's approach. However, you may not be the ultimate decision maker and need to justify engaging Crystal Seed to management.

The challenge with computing the financial benefits of code review is not knowing what, when and how software will fail and how much the failure will cost. Even the maintenance cost is not computed during development. To estimate future software cost, you need to quantify the maintenance cost of existing software plus the support and business disruption costs and assume worst for new software. New software and modifications to software will not be any better than your existing software without the essential step of code review. It really is cost effective to spend a little money up front to save a lot of time and money later.

A classic failure that could have been prevented by professional end-to-end code review is NASA's Mars Climate Orbiter that crashed on to Mars in December 1998 because the thruster software was programmed to use metric units while the ground software was giving instructions in imperial units. In addition to not achieving the goal of the mission, money spent on building and deploying was \$327 million. Code review may have cost an additional \$200,000 and unknowingly saved NASA embarrassment plus the \$327 million.

You do not have to carry the burden of convincing management on your own. Our founder is the author of the book titled "**Common Software Development Mistakes made by Managers and Developers**", with a sub-title "**Ways to eliminate poor quality software, business disruption and high technical support cost**". We will be happy to visit with your management. Contact us to discuss potentials for reducing your long-term software cost and business disruption.



Crystal Seed, LLC  
281 923-0484 | go2quality.com  
Email inquire@go2quality.com