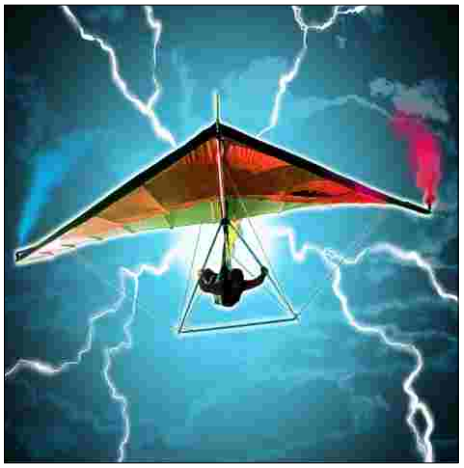
Remember to pack a parachute

Extreme Programming and Quality Assurance

Think about an extreme sport like sky surfing. What comes to mind? Challenging? Daring? Threatening to life and limb? Participants in extreme sports, who take risks and excel in ways never thought possible, know that proper safety gear is essential.

Christopher Artrip
is QA Manager of
Seapine Software

About the author
Christopher "Trip" Artrip has been involved with all phases of testing from writing test plans to performing system-wide integration and performance testing. His technical specialties include automated testing and load testing.



Just as sky surfers need back up parachutes, Extreme Programming (XP) teams need reliable Quality Assurance (QA) teams supporting them. Unfortunately, some programmers see no room for QA in the XP process; after all, a pair of programmers works on every user story, checks each other's work, and writes unit tests as they work through each story and its respective code.

While unit testing is the programmers' job, customers are responsible for higher level functional testing. In companies with internal QA departments, the QA team is the customer of the development team. Successful XP projects require the QA team and the programming team to work concurrently - therefore, the QA team must be involved in all parts of the project lifecycle from planning and requirements gathering to final release and maintenance.

According to the XP paradigm, development is performed in blocks of time with a release of functioning code at the end of each block. These development and release cycles are referred to as iterations. Before an iteration of coding can begin, the QA team must write functional test cases based on the user stories. Ron Jeffries, editor of XProgramming.com, recommends that functional tests performed on one iteration of code should be completed and ready to run no later than halfway through that iteration. This lets the QA team work on functional tests for the next iteration while the current iteration is still underway. This type of testing and development requires clear and frequent communication between the QA team and the programming team. Without this communication, QA testers cannot write tests for code that is not developed and programmers cannot write code to pass those tests.

Due to the fast pace of XP projects, it is critically important to provide test reports and performance metrics to keep projects on track. Concerning performance tracking, Jeffries suggests the following: “One of the quality measures in the process is the daily graph of performance on functional tests. The general shape of this graph, over the course of the full system release period, is that of two s-curves: the upper curve is the total number of tests written, the lower curve is the number running at 100%. A healthy project of course shows these curves coming together at 100% by the end of the schedule.”

Test performance graph example
Of course, simply generating graphs and other performance metrics does not keep projects on schedule. The daily reports must be easily accessible and visible to everyone involved in the project. Posting printed graphs on a central bulletin board is a great way to ensure the visibility of the

teams' progress. It is the high visibility of the metrics that keeps the project on schedule. Problem areas in the functionality are easy to spot, allowing the teams to quickly react to problems before the schedule is affected.

While some XP purists claim there is no room for testers in the XP process, this simply is not true. Integrating the XP team and the QA team means programmers can focus on feature implementation and testers can focus on feature usability and integration. Let's face it. An XP project without a good, solid QA team is like sky surfing without a back up parachute. Go pack yours ■

artripc@seapine.com