Achieving Better Results for Your Software Projects

By Ian Ippolito

For any of us who have seen a software project crash and burn, or simply never make it to the finish line, it's probably no surprise that only 32 percent of software projects actually succeed, based on figures from **The Standish Group**. The others are either late, over budget or are missing critical features, resulting in billions of dollars of loss for the businesses that needed them.

The good news is that best practices exist that can quickly and substantially improve your success rate. These are clustered in the general areas of hiring and managing, so let's discuss what will help you handle projects with confidence.

Hiring

Often the first consideration with hiring is the *location* of the workers. Choices include:

- Onshore (in the United States): hourly rates range from $25-250 per hour
- Near shore (Ireland, Philippines): $15-75 per hour
- Offshore (India, Romania, Pakistan): $5-25 per hour
- Further offshore: Cheapest, but there are tradeoffs and additional issues

Other location-related considerations include:

- Copyright issues: As most third world countries lack enforceable copyright laws, it's important to identify a country with effective intellectual property and copyright laws. An alternative is to split your project into pieces so no one supplier has the full project.
- Language and culture: If you require a significant amount of assistance, you should consider working with native English speakers. In addition, cultural aspects of language may mask the true meaning of what has been said. For example, "yes" may just mean "I heard you" rather than signifying agreement.
- Cost: Emerging economies such as Vietnam and Bulgaria can offer workers at rates of up to 80 percent less – something that might be considered exploitative by our standards but can be reasonable in relation to their cost of living.
- NDAs: Although using an NDA can protect intellectual property, these aren't enforceable in all countries and may well reduce the number of bidders. It's always best to check with your attorney.

Selecting a programmer is the single most crucial hiring decision you'll make in your project. Here are a few pointers:

- Deadline flexibility: If you are very flexible, you can settle for coders who are less experienced, but meet your other criteria. They are less expensive and if they fail, the "refund on failure" clause you should include will protect you. However, the less flexible you are, the more you can expect to pay and you should require coders to guarantee on time delivery with a forfeitable deposit.

- Motivation and professionalism: Look for motivated coders who ask intelligent questions and show a genuine interest in your project. Criteria to measure professionalism include whether they have antivirus coverage, backups of source code and development computer, use of source control software and, if offshore, consistent availability of power. You should verify that they own their development environment, have necessary certifications and understand the tool or language they will be working with, as well as understanding object oriented design, database design and security concerns in design.

Managing

There are a number of facets related to managing a project, beginning with *design methodologies.*

- Waterfall method: Although many people use this method, it is the riskiest and we do not recommend it. It works by having the stages – requirements, design, implementation and verification – flow into each other like a waterfall, but "freezing" at each step before proceeding to the next. Continuing the waterfall analogy, it's easy to see that going in reverse would be extremely difficult. Problems include software being out of date with real-world requirements and software is not tested until the end (verification) stage.
- Spiral development: As you picture a spiral, imagine that you identify the bare minimum core for iteration #1. After the designer or programmer prototypes it and you approve it, the programmer programs it, the quality control person tests and verifies and then a fully tested and verified version is released. After each iteration, you know precisely whether the project is on schedule and how well the programmer is doing. You can then adjust accordingly, reviewing and correcting for the next iteration, picking the next most important features and repeating. The project then continues to grow and you need not wait until the end to release, because you worked on core items first.
- Tips on maximizing spiral development: Be ruthless in jettisoning unnecessary features, pay with an hourly rather than fixed-price model and keep iterations short.

The next management area is that of *creating delivery dates that actually get achieved*. Good performance requires good management, which requires good tracking. We want to share some tips based on what we've learned over the years.

- Set deadlines (and expectations) before you start.
- Estimating: Make the programmers estimate. As they typically cannot estimate accurately in longer time spans than three hours, have them break longer tasks into two-three hour increments; and add time for "risks and unknowns." Even at that, the estimated time will not be sufficient, so multiply by 2 for an experienced programmer and multiply by 5 for an inexperienced programmer (but don't tell the programmer!). When they miss a deadline, have them re-estimate. Above all, *do not* estimate the time yourself or impose a short deadline to motivate the programmer.
- Tracking: First, ask the programmer to put times and estimates into a Gantt chart. Establish the ground rules so that the programmer must report delays immediately, update the Gantt chart weekly or if there is a problem, no task is marked "done" until it is 100 percent complete and proof is required. Never assume that a one-time

problem is one-time – after several of these, require the programmer to make adjustments in the Gantt chart.

- Missed deadlines: Decide whether you will stick with programmers who miss deadlines. It may be costly to fire them, so make sure they are of truly bad quality. If you continue with them, get a commitment to a new deadline, measure progress more carefully, increase the level of communication and, if doing a fixed price contract, require mini milestones and deadlines. If your deadline can't be extended, consider "crashing the scope": there are three elements of what we call the software trade-off triangle of schedule, cost and product. If you hold any two constant, you must allow the developer room on the third. You can always delay non critical features for a later release. However, be careful of adding people, as the additional communication needs and ramp up time usually slow down the project, rather than speeding it up. Nor is overtime an acceptable alternative, as the mental strain of additional hours reduces efficiency and accuracy. Finally, increasing project scope as an incentive is typically schedule suicide.
- Inspections: According to various resources, code inspections can reduce the schedule by up to 30 percent and is 20 times more effective than testing. You should hire a separate coder for this and perform security inspections as well. These will avoid many accidental mistakes and eliminate back doors inserted by coders.
- Quality control: Although it may be impossible to create bug-free software, quality control more than pays for itself. However, testing requires time and work. Your coders should perform the "unit testing" while you do "user acceptance testing." To do so, act like a normal user and exercise all functionality. Then try to break the program by doing unexpected things. Record everything you find in a flaw list.

Ian Ippolito is founder and CEO of vWorker (www.vworker.com), a global online job marketplace connecting entrepreneurs and professionals with skilled workers in fields such as website design, secretarial services and software development. He can be reached at IanIppolito@exhedra.com.