Avoid disposable development with the 5—ities:

Successful software development requires a focus on the non-functional features

By Rodney Gill, Director of Project Delivery, Affinity Systems

Outsourcing software development creates efficiencies, allowing a focus on core competencies and fostering innovation. Too often, though, missteps around the choice of outsourcing vendor—sometimes, but not always geographic—will result in broken or unrealized projects.

Rebounding from IT project failure takes a tremendous toll; the wrong failure—a mission-critical application or system—at the wrong time can spell the end for a business.

Typically outsourced software projects don't fail because established criteria and specifications are not being met by the developer, but rather because unspoken-but-crucial features are overlooked due to inexperience or to cut costs. Investing in and specifying these crucial non-functional features or "-ities"—usability, maintainability, scalability, security and reliability—differentiates the software triumphs from flops.

Examples abound in which the absence of one or more –ities have had devastating results.

Short-term projects and one-off applications may not need to incorporate all five – ities, but long-term, sustainable projects must incorporate many. Doing so demands proactivity and answering numerous, in-depth questions of the developers.

USABILITY:

An application is only a success if intended users—whether employees, customers or prospects—actually use it. Usability is not simply an attractive appearance, although aesthetics play a part. It requires an understanding of users' roles and activities in the context of project goals.

Consider also accessibility for people with disabilities, increasingly important and mandated in some jurisdictions over the next decade. Add to that accessibility by the plethora of different devices and clients available on the market.

Usability-challenged applications often have great functional features that users cannot find (or be bothered to find), rendering them useless.

Questions to ask: Is the developer involving users throughout analysis, design and testing phases? Does the developer have access to people with the educational background—such as psychology, design, computer science, information science—to perform all UX (user experience) activities?

MAINTAINABILITY:

An application meant to last must be maintainable. This requires proactive consideration of, and designing for, future functionality. With the pace of innovation and the nature of the Internet, forthcoming features cannot always be anticipated, so the software should be designed to adapt.

The alternative is for major innovations to demand a forklift upgrade to the whole system or starting over from scratch. Knowledge transfer is of utmost importance to mitigate risks around changing external vendors.

Maintainability-challenged projects can overlook the simplest of future changes, because the business and developers didn't see them as important at the time. For example, cycles wasted when it takes a 48-hour overhaul to change copyright notices on a website.

Questions to ask: *Does the development vendor have a plan to continuously transfer knowledge through the life of the project? Can the vendor demonstrate that their development practices simplify maintenance, enhancement and troubleshooting? Will they commit to following your coding and development standards and have their software audited throughout the project?*

SCALABILITY:

Designing for scalability always comes with a cost, but as the adage goes: a stitch in time saves nine. Investments made to ensure scalability prevent much higher future costs. The cloud is not the panacea for scalability that some believe, since software not designed to scale still cause costs to skyrocket alongside performance demands.

To support scalability an application must be designed to scale and key performance indicators must be measureable throughout the development and deployment. Performance tests, load tests and stress tests should determine not just when a solution breaks, but how gracefully it fails.

Without software designed from the ground up to scale in the best case scenario expensive hardware must be purchased to solve impending scalability issues. In worst-case scenarios systems, systems break under the pressure—just when businesses need them most.

Questions to ask: *Does the developer's proposed solution accommodate the highest possible growth pattern? Can they demonstrate the scaling features of the solution during development? Do they evaluate performance targets throughout the development lifecycle? Is the solution instrumented so performance can be easily measured?*

SECURITY:

Security must be considered at every phase of development, not as an afterthought just before deployment. It must be spelled out during the requirement phase, and security use cases—and abuse cases—must be developed.

At the design phase, the framework and architecture should be built with security in mind. What actual code is to be trusted, what needs further scrutiny? The solution's security must be lab-tested and auditable. Security-hardened software should introduce trust boundaries between other technology components, and must also take into account human error and the prospect of internal breaches.

One needs to look no further than the front pages of news sites and newspapers to see the devastating impact security-challenged software solutions have on major retailers, governments and service companies.

Questions to ask: *Have security details been specified in the requirement phase and carefully considered in the design phase? Is there one clearly defined person with oversight that is responsible and accountable for security?*

RELIABILITY:

Not just how well the application runs but, if something does go wrong, reliability represents the grace with which it degrades and how well does it recovers? Like scalability there is cost associated with the redundancy required for reliability but, again, this up-front expense prevents future loss associated with downtime and lost business.

Uptime expectations must be incorporated into reliability decisions—the application may or may not need to run 24/7—and its environment. Software makes assumptions about the environment on which it runs; if one element fails, does the entire solution go down? Counteract this by developing solutions with independent software components.

Not all failures result from technology, the worker tripping over a cord must be considered, as well as how the system reacts and recovers from such human error.

Questions to ask: *Are availability and fault tolerance measures specified during the requirements phase? Is failover testing and stress testing performed continuously during the project lifecycle in order to refine graceful degradation and recovery?*

Like any project, balance must be found between the –ities and associated costs, but many, such as security, do not cost more when incorporated up front. Others need to be looked at from the perspectives of mitigating risk and creating operational efficiencies.

Just think of the staggering cost of the days, or sometimes weeks, of downtime caused by solutions that lack reliability or scalability, the expense of starting again from scratch every time a new innovation reaches the market, or the waste of a fine application that customers or employees don't take to. Covering off the five "-ities" makes sure those don't happen.

ABOUT AFFINITY SYSTEMS

For 25 years, Affinity Systems has been building world-class custom software solutions. Affinity combines broad industry experience with deep technical knowledge to deliver unrivaled and insightful solutions that exceed expectations. Affinity has an extensive record of tackling complex software challenges from design and development through implementation and support. As a Microsoft and Kx Systems partner, Affinity's thoughtful and systematic approach to custom software solutions, business intelligence, big data, and software quality assurance has established it as an industry leader. For more information, visit www.affsys.com