

Four Steps to Improve Software Development Projects

By Robbie Mac Iver

To the uninitiated, and some who are, software development is a deep mystery. It is a unique domain with its own language that often intimidates those who are not members of the club. Some even consider it a dark art that is best left undisturbed. Although few are completely happy with the results they get from their software development efforts, little action is taken to remove the mystique and make the changes necessary to get the full value from their efforts. Here are four ways to get started.

It's about the Business



Software development efforts are not about technology. They are about business. They make the business bigger, better or more efficient. If they don't we should seriously reconsider why we are developing software. Because these efforts are about business, organizations need to be very selective in just what they choose to develop. In today's world of ever-growing technology, the need for software does not equate to the need to develop it. There are two rules I would apply to the selection of development projects:

- Does development require the knowledge of the intellectual property of the organization? If so, the organization must develop this software, no one else can without obtaining your IP.
- Does development require knowledge of significant domain knowledge of the organization? Beyond just industry knowledge, does the development effort require significant information about how the organization operates in the industry that differentiates it from all others? If so, the organization should develop this software. The domain knowledge may not be on the level of intellectual property, but it is likely still company confidential and should be protected.

If you can't answer yes to either of these questions, then anyone could develop the software in question, and there is a good likelihood that some already has. Software development resources are too precious, and development is too expensive and too risky to consider undertaking these kinds of efforts.

1

It's all about the business, do a rigorous job of project selection. Be very certain that the effort provides a value to the business that only your organization can bring to fruition.

Success from Failure

Shortly after the September 11, 2001 attacks on the World Trade Center, the Federal Emergency Management Agency (FEMA) and the American Society of Civil Engineers (ASCE) in conjunction with others, conducted a study to determine the exact nature of the structural failure of the twin towers¹. This study also became the basis of a NOVA documentary “Why the Towers Fell” broadcast on PBS in April 2002². No one doubted the structural integrity of the buildings, or the competence of the architect Minoru Yamasaki or the builder. In fact most engineers were surprised by the collapse of the towers. The study was undertaken to learn what could be done to lessen such catastrophic results in the future. Similar studies have been done for centuries in many different engineering disciplines for the purpose of improving the trade. That is why we enjoy the relative safety and comfort we do today in modern skyscrapers, air planes, automobiles and a limitless list of elements that make up the infrastructure of our daily lives.



By comparison software engineering is a relatively young discipline, and industry study after industry study shows a remarkably poor success rate for software development efforts. Based on the criteria of delivering all the requested functionality on time and within budget, the 2004 CHAOS research³ indicates that only 34% of software development projects succeed. More remarkably, in the face of such research few

organizations routinely exam their unsuccessful projects to determine the cause, and as a result there is little or no improvement of the trade. Failures are not often fun, and they are not pretty; but they are always educational if we let them be. With 2 out of every 3 projects failing industry wide, we have some significant education ahead of us.

2

Learn from past experiences. We must work to defeat the organizational culture that says “failure is not an option”. It is an option, and currently it is the reality most of the time. No one wants to fail, but when we do we need to learn from it, and improve the trade.

Experiment in Reasonable Ways

The definition of insanity is doing the same thing over and over and expecting different results.

- Benjamin Franklin

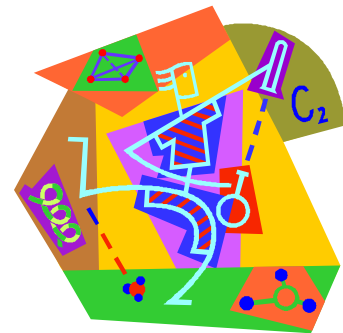
Are your software development efforts like this? When a project does not meet its expectations do you declare “we’ll do better next time”, but then do nothing different?

Hard questions to answer honestly, but the truth is simple.

If we want different results from our software development efforts, we have to do something different.

There is no silver bullet of software development, no magic formula, no wondrous series of steps that will guarantee success every time. There are very few guarantees in life, and none of them have anything to do with software development. When we do have a success, we should not expect that repeating the process step-by-step will succeed on the next project. As your financial advisor likely tells you, past results is not indicative of the future performance. With each project we are developing different software, under different circumstances, for a different audience, for a different purpose. Think back to the two rules of project selection; we develop for intellectual property, or significant domain knowledge. That means every effort will be at least a little different. How we approach each effort should have some differences as well.

While the broad framework of the software development life cycle should remain fairly constant, we need to have the flexibility and the willingness to experiment in reasonable ways that have the potential to help us be better. We keep the things that work, and look for new opportunities for those things that don’t. When we do this we need to recognize that some experiments will work out really well, some will work out really poorly, and most will fall unremarkably in the middle. Each is a learning experience; each can lead to better results; each helps us improve the trade.



3

Make it acceptable to experiment in reasoned ways that offer the potential to learn and apply better techniques and methods that increase the value of the overall effort.

Software as Invention

In his book Agile & Iterative Development⁴, Craig Larman defines two broad categories of projects:

Predictive

Projects for which all the required work can be defined upfront with a high degree of confidence that the work will remain stable throughout the project life cycle.

Inventive

Projects for which the required work cannot be defined upfront but must evolve over the time and require a life cycle that accommodates that evolution.

Constructing a building is a predictive project. Upfront there is a definition of what space is needed, and how it will be used. Design elements are added, along with detailed plans for the structure and all the various infrastructure components of the building. A detailed plan can be produced to complete the work, and the construction team can execute the plan with reasonable comfort that significant changes will not happen. Once the framing of the structure is complete at 25 stories, there will be no discussion about adding 25 more stories because the market conditions have improved for rental space. Those types of changes do not happen in a predictive project, they are too risky and too expensive.



Software projects selected using the two selection rules of intellectual property or domain knowledge are inventive. Too little is known to define all of the work upfront. What requirements we do know are likely to change. When we get 25 use cases completed, there will be a discussion about 25 more, and then 25 more after that. This is the nature of software development.

To succeed we must adopt an inventive approach to software development. First we start with a broad vision of the end result, realizing that we don't know the exact outcome and we don't know exactly how to get there. Second we begin with the first two or three logical steps that will allow us to learn something significant about the project. Third, based on what we have learned, we decide if we should continue and if so we continue with the next two or three logical steps. Agile, or iterative and incremental development processes are based on this concept. The exact nature of the project evolves over a period of time as we learn more and more about the end result.

This inventive approach needs to pervade not only the project team, but also the project sponsorship and the management team of the organization. Most corporate organizations are steeped in a predictive culture that is inappropriate when thinking about software development. When you hire a fence builder to build a fence around your house he can

give you a very precise estimate of both cost and schedule. He can do this because he has built the fence before. Based on his past experience, he knows how to determine what lumber and materials he will need and how long it will take him to build it. Using that information he can easily calculate cost and schedule per linear foot, and comfortably provide a precise quote. This is known as parametric estimating. Many sponsors and funding authorities have the perception that this type of precise estimate can be provided upfront for software development efforts. It cannot, and we need to work at changing that perception. *Software is invention*; we have not built it before, if we had we likely should not be building it again. *Software is invention*; we don't know what the precise outcome will be upfront. *Software is invention*; the path to the outcome is not clear.

To be successful we need to adopt inventive processes not only for the development effort itself, but for the funding and oversight of the effort as well. Initial judgments are made on an order-of-magnitude basis. As the effort evolves and we learn more about the outcome, we can refine the precision of our estimates using the feedback gathered from each step. Periodic checkpoints are conducted with all stakeholders to assess the value being provided to the business. If the value is there, the effort continues; if not, it stops and resources can be re-directed to another effort that does provide value.

4

Change the metaphor for software development. Software development is not like building a building, it is like exploring for oil, or mining for precious metals; develop a big picture, take the first few logical steps, assess the value of continuing based on what has been learned.

On to Success



Software development efforts are not about technology, they are about business. Their goal is to make the business bigger, better, more efficient; to add value to the business. As such, development efforts need to be focused less on “on-time, on-budget, all the requested functionality”, and more focused on delivering business value on a predictable basis. A successful business is in a constant state of change as it re-assesses its market and re-positions itself to its own advantage. It is not reasonable to think that on-going software development efforts in these environments won't change as well.

Software development success cannot be guaranteed, but results can be significantly improved by:

- Careful selection of the efforts to undertake.
- Making failure an acceptable learning experience.
- Getting different results by experimenting in reasonable ways.
- Thinking of software development as invention.

Focus your organization on these four areas and your software development efforts can only improve.

softwareDecisions, is a Texas Limited Liability Partnership providing information technology consulting services to the greater Houston area. *softwareDecisions* is dedicated to making software development and integration efforts succeed by addressing the right requirements, focusing on the right priorities, and building the right relationships.

Robbie Mac Iver, PMP
Owner/Principal Consultant
President, Mac Iver Management, Inc., General Partner

¹ For more information please see ArchitectureWeek's article [Engineers Explain WTC Collapse](http://www.architectureweek.com/2002/0515/news_1-1.html) at http://www.architectureweek.com/2002/0515/news_1-1.html.

² For more information please visit Nova's [Why the Towers Fell Site](http://www.pbs.org/wgbh/nova/wtc/) at <http://www.pbs.org/wgbh/nova/wtc/>.

³ The CHAOS research by The Standish Group International. http://www.standishgroup.com/chaos_resources/index.php

⁴ Craig Larman 2004. "Agile & Iterative Development – A Manager's Guide". Addison-Wesley