Dr.Dobb's
THE WORLD OF SOFTWARE DEVELOPMENT

The Non-Existent Software Crisis: Debunking the Chaos Report

An alarmist report that's become a universal reference in discussion of development practices obscures a much less dire reality.

February 04, 2014
URL:http://www.drdobbs.com/architecture-and-design/the-non-existent-software-crisis-debunki/240165910

The software development success rate published in the Standish Group's Chaos Report is among the most commonly cited research in the IT industry. Since 1995, the Standish Group has reported rather abysmal statistics — from a rate of roughly one-in-six projects succeeding in 1995 to roughly one-in-three projects today. Ever since the Chaos Report was published, various Chicken Littles have run around warning about how this "software crisis" will lead to imminent disaster. However, this supposed "software crisis" is complete and utter hogwash, it always has been and I suspect always will be. Sadly, that doesn't stop people who should know better, or at least should be able to think for themselves, to continue quoting this nonsense.

Since 2006, I have organized an almost-annual survey for *Dr. Dobb's* that explores the actual success rates of software development projects. The most recent was conducted in November and December of 2013 and had 173 respondents. The original questions as asked, the source data, and my analysis can be downloaded for free at 2013 IT Project Success Rates Survey results. The survey was announced on the *Dr. Dobb's* site, on the Ambysoft announcements list, my Twitter feed, and several LinkedIn discussion forums. The results of this study are much more positive than what the Standish Group claims. They still leave significant room for improvement, but they certainly don't point to a crisis.

The success rates by development paradigm are summarized in Table 1. As you can see, all paradigms (even an Ad hoc approach) fare much better than a one-in-three success rate. In this study, we asked people to judge success based on criteria that was actually applicable to the team at the time. In this case, a project is considered:

- *Successful* if a solution has been delivered and it met its success criteria within a range acceptable to the organization;
- *Challenged* if a solution was delivered, but the team did not fully meet all of the project's success criteria within acceptable ranges (for example, the quality was fine, the project was pretty much on time, but ROI was too low);
- *Failed* if the team did not deliver a solution.

Paradigm	Successful	Challenged	Failed
Lean	72%	21%	7%
Agile	64%	30%	6%
Iterative	65%	28%	7%
Ad hoc	50%	35%	15%
Traditional	49%	32%	18%

Table 1: Software development success rates by paradigm.

Each paradigm was well-defined. Respondents were first asked if their organization had any project teams following a given paradigm, and then what percentage of projects where successful, challenged, or failed. A weighted average for each of level of success was calculated. A selection of 91-100% was considered to be 95%, 81-90% as 85%, and so on. A selection of 0 was considered as 0, and answers of "Don't Know" were not counted. I then had to normalize the value because the weighted averages didn't always add up to 100%. For example, the weighted averages may have been 60%, 30%, and 20% for a total of 110%. To normalize the values, I divided by the total, to report 55% (60/110), 27% (30/110), and 18% (20/110).

Defining the Paradigms

The paradigms in this survey were defined as follows:

- **Ad hoc.** On an Ad hoc software development project, the team does not follow a defined process.
- **Iterative**. On an iterative software development project, the team follows a process that is organized into periods often referred to as iterations or time boxes. On any given day of the project, team members may be gathering requirements, doing design, writing code, testing, and so on. An example of

an iterative process is RUP. Agile projects, which are defined as iterative projects that are performed in a highly collaborative and lightweight manner, are addressed later.

- **Agile**. On an Agile software development project, the team follows an iterative process that is also lightweight, highly collaborative, self-organizing, and quality focused. Examples of Agile processes include Scrum, XP, and Disciplined Agile Delivery (DAD).
- **Traditional**. On a traditional software development project, the team follows a staged process where the requirements are first identified, then the architecture/design is defined, then the coding occurs, then testing, then deployment. Traditional processes are often referred to as "waterfall" or simply "serial" processes.
- **Lean**. Lean is a label applied to a customer-value-focused mindset/philosophy. A Lean process continuously strives to optimize value to the end customer, while minimizing waste (which may be measured in terms of time, quality, and cost). Ultimately, the Lean journey is the development of a learning organization. Examples of Lean methods/processes include Kanban and Scrumban.

Ad hoc development (no defined process) and the Traditional approach had statistically the same levels of success in practice. Our previous studies have also shown this result. However, when you take team size into account, Ad hoc is superior to Traditional strategies for small teams, yet the opposite is true for large teams. Agile and Iterative strategies had similar results on average, which we've also found to be true in the past, regardless of team size. For the first time, Lean strategies for software development, supported by both Kanban and the DAD framework, were notably more successful than the other four paradigms we explored. Food for thought.

Time to Get Lean?

For each paradigm, we also looked at several potential success factors. The questions we asked were:

- Time/Schedule. When it comes to time/schedule, what is your experience regarding the effectiveness of [paradigm] software development teams?
- Budget/ROI. When it comes to effective use of return on investment (ROI), what is your experience regarding the effectiveness of [paradigm] software development teams?
- Stakeholder Value. When it comes to ability to deliver a solution that meets the actual needs of its stakeholders, what is your experience regarding the effectiveness of [paradigm] software development teams?
- Product Quality. When it comes to the quality of the system delivered, what is your experience regarding the effectiveness of [paradigm] software development teams?

For each success factor, respondents were given options of Very Effective, Effective, Neutral, Ineffective, Very Ineffective, and Not Applicable. For each question, we calculated a weighted average by multiplying the answers by 10, 5, 0, -5, and 10 respectively (answers of "Not Applicable" were ignored). By doing so, we're now able to compare the relative effectiveness of each paradigm by success factor, which you can see in Table 2. It's interesting to note that Lean approaches were perceived to provide better results on average than the other paradigms. Also, the three development paradigms of Lean, Agile, and Iterative were significantly better across all four success factors than either Ad hoc or Traditional (waterfall). In the vast majority of cases, only cultural inertia can justify a Traditional approach to software development these days.

Paradigm	Time/Schedule	Budget/ROI	Stakeholder Value	Product Quality
Lean	5.7	6.0	5.5	4.8
Agile	4.3	4.2	5.6	3.8
Iterative	4.9	4.2	5.6	3.8
Ad hoc	0.0	-0.4	1.9	-1.4
Traditional	-0.7	0.5	0.1	1.9

Table 2: The effectiveness of each software development paradigm.

How Is Success Defined?

The survey also explored how people define success, which I argue is exactly where the Chaos Report falls apart with respect to measuring project success rates. Similar to what I found in the 2010 study, there was a robust range of opinions when it came to defining success:

- Time/schedule: 16% prefer to deliver on time according to the schedule, 39% prefer to deliver when the system is ready to be shipped, and 42% say both are equally important
- Budget/ROI: 13% prefer to deliver within budget, 60% prefer to provide good return on investment (ROI), and 23% say both are equally important
- Stakeholder value: 4% prefer to build the system to specification, 86% prefer to meet the actual needs of stakeholders, and 10% say both are equally important
- Product quality: 10% prefer to deliver on time and on budget; 56% prefer to deliver high-quality, easy-to-maintain systems; and 34% say both are equally important

What I find very interesting is how the Agile and Lean philosophies toward time, money, value, and quality are given greater importance than traditional

ways of looking at those success factors. For example, when it comes to time/schedule 58% (16%+42%) want to deliver on time according to schedule, whereas 81% (39%+42%) believe in delivering when the solution is ready to be shipped. With respect to budget/ROI, 36% of respondents want to deliver on or under budget, whereas 83% want to provide the best ROI. For stakeholder value, 14% believe in building the system to specification compared with the 96% who prefer to meet the actual needs of stakeholders. When it comes to product quality, 44% want to deliver on time and on budget, whereas 90% want to deliver high-quality, easy-to-maintain systems.

The Software Crisis that Wasn't

For years, I have actively questioned the Chaos Report findings in print, in seminars, and in conference presentations. You don't need to do a detailed study as we've done here; instead, you need only take your brain out of neutral and think for yourself. If the Chaos Report's claim that we have a one third success rate at software development is true, then roughly half of organizations must have less than a one third success rate, and roughly half must have a greater than one third success rate. How many organizations have you seen with a less than one third success rate? Any? I get around, at one point in my career I was visiting 20+ organizations a year, and I've never run into one with such an abysmal track record.

Why are the results of this study noticeably more positive than what is published in the Chaos Report? It's in the way that success is defined. The Chaos Report has never measured the success rate of software development projects. Ever. What it measures is whether project teams are reasonably on time, on budget, and are building something to specification. To be fair, the Standish Group has always been very clear about this, *but* they unfortunately have also marketed this as if it's a valid standard measure of the success of software development teams. As our study found, only a very small minority of teams seem to use these as their success criteria.

Instead of inflicting an artificial definition of success on software development teams, this study instead asked people to rate their projects in terms of the success criteria that actually applies to them. Just as important, we also explored how people define success in the first place, and not surprisingly, found that every combination of the four categories of the criteria we asked about (in this case 2^4) were valued by someone: Sixteen ways to define success, not just one. Had we also looked into team morale, something I intend to do in the next study, I suspect we'd find all 32 combinations. For example, this study found that only 25% of respondents valued both being on time and within budget on software development projects, and only 8% value being on time, on budget, and to specification. By judging teams with criteria that are applicable in only one-in-twelve situations, is it any wonder that the Chaos Report's results are so far off? The claims of the existence of a "software crisis" were built on an ill-formed definition of success. Get the definition right and, *poof* — no more software crisis.

To understand why this misinformation is still being spread, I'll ask several rhetorical questions: Why would vendors of expensive tools want you to believe that there is a software crisis? Why would expensive software consultants plying their services want you to believe that there is a software crisis? Why would research organizations selling expensive studies want you to believe that there is a software crisis? Why would public speakers pitching their strategies want you to believe that there is a software crisis?

We may never have definitive answers to these questions, but we should at least consider them.

The good news is that it seems that the Standish Group knows that its success rate claims are problematic. It does seem to be rehabilitating itself by also publishing value-based measures in addition to their "on time, on budget, to spec" measures. Perhaps giving people a choice will help them to take their traditional blinders off, time will team. Regardless, whenever someone quotes the Chaos Report's claim that only a third of software development teams are successful, please challenge them. At some point, we really do need to stand up and "call BS."

Conclusion

The results from the 2013 IT Project Success Rates Survey, including the questions as asked, the data as answered, and my analysis are available free of charge. At Comparing Software Development Paradigms, I've shared some infographics that you're welcome to use that summarize the survey results.

You can read about our previous success rate studies at:

- How Successful Are IT Projects, Really? (2011)
- 2010 IT Project Success Rates (2010)
- Software Development Success Rates (2009)
- Defining Success (2007)

Finally, I should note that the Standish Group does in fact have some interesting and valuable insights to share regarding the effectiveness of various IT strategies.

February 2014 IT State of the Union Survey

I invite you to fill out the February 2014 State of the IT Union Survey. The goal of this ongoing survey series is to find out what IT professionals are actually doing in practice. The survey should take you less than five minutes to complete, and your privacy will be completely protected.

The results of this survey will be summarized in a forthcoming article. This is an open survey, so the source data (without identifying information to protect your privacy), a summary slide deck, and the original source questions will be posted so that others may analyze the data for their own purposes. Data from previous surveys has been used by university students and professors for research, and I hope the same will be true of the data from this survey. The results from several other surveys are already posted, so please feel free to take advantage of this resource.

Surveys such as this are not only a great way for you to compare your team with the rest of the industry, they're also an opportunity for you to give back to the rest of the community through sharing your experience.

Scott Ambler is a long-time contributor to Dr. Dobb's.