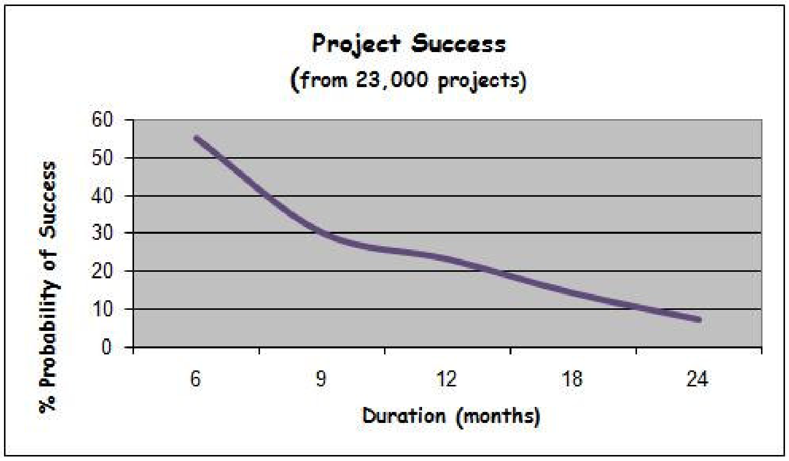
Large Project Risks: Chop those large projects down to size

When I started out as a PM and had a few successful projects under my belt, I wanted to manage larger and larger projects. Now I'm looking for ways to make them smaller and limit the functionality initially tackled. This is not just laziness or a lack of ambition on my behalf, instead it is a realization that delivering business value comes from successful projects and that large projects are inherently risky and more likely to fail.

Jim Johnson of the Standish Group presented some interesting metrics at the PMI Global Congress conference in Toronto a couple of years ago that confirmed my suspicion. From a study of over 23,000 projects it was found that the success rate dropped as project duration increased.



From the graph we can see that most (over 50%) of the 6 month projects surveyed were deemed successful, dropping to 23% of 12 month projects, and less than 10% of 24 month projects. Now, we need to understand what "Sucess" means here. The criteria was quite strict and defined as within 15% of cost and schedule and to customer defined functionality and quality standards.

I expect a large proportion of these "failed" projects merely missed the 15% cost or time criteria and the picture would not be so bad with a wider margin. Predicting costs over long periods is notoriously difficult as even labour rates and inflation predictions elude the best market ecconomists. However the trend is still true, larger projects carry a much higher probability of failure for a number of reasons we will see.

During the description of these numbers Jim was keen to stress that it is not valid to extrapolate the downward trend and deduce that no projects over 30 months are successful. Some companies (NASA, Boeing,IBM, etc) have a good track record of delivering large projects, yet these companies are the exceptions, most companies struggle with large projects.

As a project manager with a self interest in having successful projects, I like the odds on short projects a whole lot more than long ones. That's why my recommendation is to chop big projects into small ones if you can. Before giving some strategies on how to do this, we will dig a little deeper into the reasons for these failures.

Why are large projects so likely to be deemed a failure?

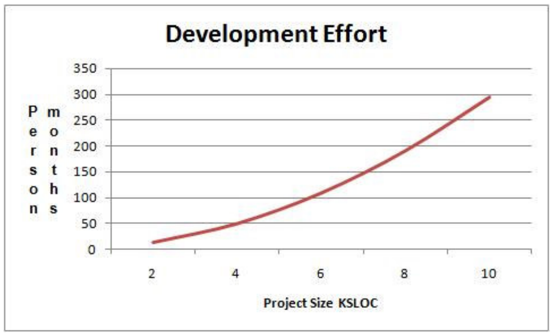
Complexity Increase is Non Linear

As projects grow in size the effort to create them grows at a faster, non-linear rate. In other words it takes more effort to write one 20,000 line application than two 10,000 line applications due to the interdependencies, difficulty maintaining the big picture, and additional co-ordination effort. This phenomenon is recognized by effort estimation software. The basic COCOMO (Constructive Cost Model) estimation formula is:

Effort = Size^ $^{\text{Penalty}}$ * Productivity Factor * Productivity Adjusters

The portion of this formula I want to focus on is the Penalty. Penalty is an exponent that adjusts the effort in an exponential way the larger the project gets. The default value of Penalty in the COCOMO formula is 1.030 for web applications. Then project adjustment factors add to or subtract from this value. For instance “Poor Team Cohesion” adds 0.0264 on, “working with a new process” adds another 0.0469.

Whenever you raise a number by a power greater than 1 the end result gets bigger fast. The following graph shows the person effort estimates for developing web applications of various sizes.



The effort line is not flat but instead has an upward curve, larger projects get harder in a non linear way.

Business Changes

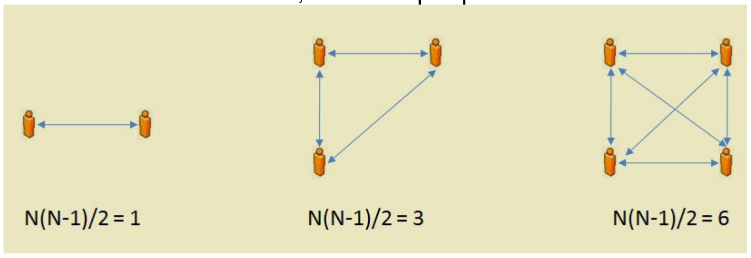
The business changes and the rates of business change are accelerating as companies compete in an increasingly global market. Companies now need to innovate and evolve faster than ever before as they face competition from around the world. Teams would need near clairvoyant abilities to anticipate the true business requirements two years down the line in many industries. While agile methods and their capabilities to cater with changing requirements are increasing in popularity, market penetration is still too low and many companies are unable to (or refuse to) react to evolving requirements.

People Change

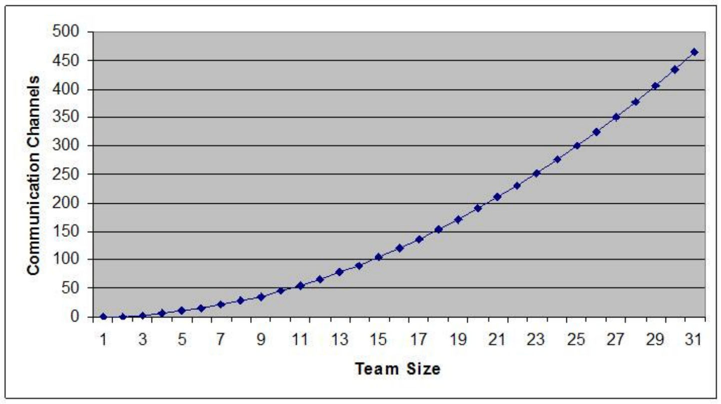
People change their jobs and their minds. What was the current CIO’s priority project could be thought of as a waste of resources to the next CIO. Or even last year’s pet project could be superseded by something new by the same sponsors. Likewise priorities are not static and people change their minds about ideas as the business and industry moves along. Team members move on too, a project of two years or more is likely to see a fair percentage of original team replaced over the project lifetime. As people leave, unwritten, tacit knowledge often leaves the project too and the delays due to relearning can be enough to push the project in Standish’s “Failed” category.

Communication Channels Increase Exponentially

As we add team members to a project the number of communication channels we need to manage to keep everyone informed rises in a near exponential fashion. With just two people on a project there is only one communication channel between them, three people have three communication channels, and four people have six.



The number of communications channels on a project follows the formula n(n-1)/2 and rises quickly.

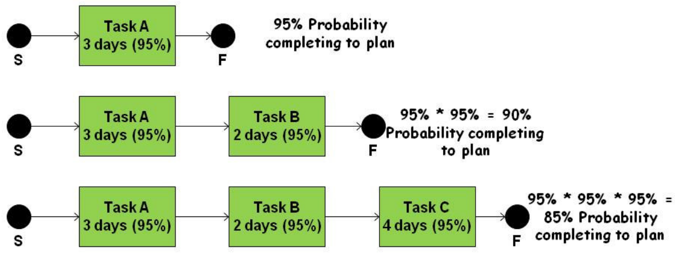


As we can see from the graph above, on small teams of 8 people the communication channels are small (16) but as team sizes reach, say 15 people then the communication channels have risen to 105 and by 30 people then have reached 435. This is why small teams can do more face-to-face meetings and need to commit less to documentation. As team size exceeds 10 people face-to-face communications tend to break down and more needs to be written which takes longer and conveys less information.

The larger the project the higher the probability that communication gaps will occur. As Francis Hartman observes in his book "Don't park your brain outside: A Practical Guide to Improving Shareholder Value With Smart Management", *"project failures can be directly attributed to communication failures".*

Compounding Task Completion Probabilities get small fast

As we chain a series of uncertain tasks together in a project plan the overall probability of completing the whole project when originally scheduled reduces dramatically.


S
Task A
3 days (95%)
F
95% Probability completing to plan
S
Task A
3 days (95%)
Task B
2 days (95%)
F
95% * 95% = 90% Probability completing to plan
S
Task A
3 days (95%)
Task B
2 days (95%)
Task C
4 days (95%)
F
95% * 95% * 95% = 85% Probability completing to plan

In the last example above, we can see that with only three tasks (all at 95% probability) the chance of finishing on schedule has dropped from 95% to 85%. However, projects have more than three tasks and more like 300 or 3,000 tasks. Now the odds look really bad

$$P(\text{Completing to Plan}) = \prod_{n=1}^{N} P(\text{Completing Task}_n)$$

Number of 95% Tasks	Probability of completing to plan
10	60%
50	28%
100	0.6%
200	0.003%

Of course some of the under's will cancel out the over's (but guess which way the trend goes...) and we soon see how estimating large, uncertain endeavours is tremendously challenging for the PM and frustrating for the business. Being able to adjust scope to meet constraints can certainly help.

Chopping Projects Down to Size

The good news is that our sponsors (generally) want the same thing as we do as project managers, namely successful projects. By explaining the risks I have listed here they usually see how it would be in everyone's best interest to find a way to break projects up.
Some Project Chopping Options to consider include:

By Functional area – look for logical separation points and break the project at these boundaries. If a system has to capture customer orders, manage inventory, and produce financial reports consider each portion as a candidate for a separate project.

Plain Vanilla First – build the system for the 80% of transactions that are straightforward first and add the exceptions as a follow on project. This will deliver the major business benefit early to the organization and it is surprising how often the 20% of exceptions actually shrinks or can be handled off system. If you know things have to be added later keep the design open, but do not code for them now. YAGNI

By User Group – build and implement for the order entry folks first and add the management reports as a latter project. Obviously, we need to consult and ensure necessary details are captured in the first project, but simplifying the requirements set and stakeholder groups will benefit each project.

By Sub-system Replacement – If your new whizz-bang system will replace four existing systems, assess the architecture and tackle them one at a time. Issues become muddled when there are lots of competing groups and addressing them one at a time reduces finger pointing and political manoeuvring.

Some options to avoid include:

By project phase – Do not run requirements gathering and analysis as a project and then spin up another project to develop and test it. This is even worse than a single large project as there is less incentive to do a good job on the first project if the team might be different on the second. An analysis project delivers very little business value, and there is ample opportunity for the follow-on project to blame issues (rightly or wrongly) on the first project.

By financial year – This is not as bad as dividing by phase, but really misses the complexity savings of smaller projects. We want fewer participants to manage risk; merely chopping up a broad team into shorter pieces misses the main benefit.

Conclusion

So, do yourself and your organization a favour, don't take on the mega projects without considering how to make it a series of mini projects first. Some projects are huge and there is little we can do to reduce them, but many more projects are larger or more complex than then could be. I would suggest we park our PM ego at the door not our brain; and manage a collection of small, boring projects to successful completion.

Mike Griffiths is an independent consultant specializing in effective project management. Mike was involved in the creation of DSDM in 1994 and has been using agile methods (Scrum, FDD, XP, DSDM) for the last 13 years. He serves on the board of the Agile Alliance and the Agile Project Leadership Network (APLN). He maintains a leadership and agile project management blog at www.LeadingAnswers.com.