Nick Alexander
10/19/2015

Mythical Man Month

In nature, one will find that a tar pit can pose a serious problem for ever the largest of creatures. Metaphorically speaking, the field of software engineering is littered with tar holes. As a project progresses, things can – and likely will – go awry, and the entire project will come to a halt as though its stuck in a tar pit. This problem has existed since the dawn of large computer systems. The author notes that the production of OS/360 encountered the same. When asked if tar pits still exist in modern systems, the answer is an indisputable affirmative. Projects get delayed and some even get cancelled do to unforeseen snags. It can be personnel issues, administrative issues, budget issues, time issues, or a score of other equally devastating problems. Even while pursuing an undergraduate degree in computer science students often encounter tar pits. The problem is every bit as potent today as it was in the 1960s and will remain equally as potent for the largely foreseeable future.

Intercommunication. Intercommunication is the root of many problems in the production of large computing systems. Imagine a machine with 100 moving parts each of which perform a dependent, critical function. If even a single part fails, the entire machine will fail immediately. A software project with 100 team members is similar. If communication falls short, portions of the project will be lacking and may fall behind. With 100 minds coming together to build a single system, it must be perfectly clear what each of the 100 minds are doing to change the system.

Regardless of how much we study and strive to avoid it, projects will be delayed and some may even fail. Despite the disparaging truth, an adept manager can work to provide better schedules and better estimations of resources that prevent projects from being delayed as often or for as long. The author presents three reasons why most projects are delayed. First, the estimation techniques are, ultimately, nothing but assumptions that every moving part will never fail. Managers must always be aware that things can go extremely wrong and should always allot a certain amount of "padding" time in the estimation. If the padding was not needed, the project will be completed earlier than estimated which looks good on the development team. If the padding time is needed, it is much less likely to delay the project.

Secondly, managers are often times unwillingly or incapable of sternly requesting clients to simply "be patient." With a demanding client, a manager is much more likely to underestimate the time required. Then, when the deadline comes and there is not a finished product, delays are required. As a remedy, a manager needs to be completely forthcoming and needs to, to the best of their ability, be realistic with a client. A manager should be intimated to decrease the required time simply because the client threatened to pull out of the deal. Losing the client on the spot and losing the client after 3 project delays are not much different overall.

Finally, when a project begins to look more and more unfinished while approaching the deadline, the naturally immediate response is to add manpower. If 3 people take 2 days to harvest a field, then 6 people should only take 1 day to harvest a field. However, according to

Brooks’ Law, this is far from the truth. Adding new team members requires training and time for the new members to become acquainted with the project. The time and resources required can easily cause the project to become more delayed. When faced with this situation, a manager must carefully weigh the pros and cons of adding new team members. Only if one is absolutely certain that new members will help should more members be acquired. Otherwise, the manager’s best course of action can often times be to request an extension.

The theory behind the Mills’ surgical team was creating a team of the least number of people that possessed the required skill to create a functioning system. Though this construct was proposed decades ago, the general concept is still valid. With modern day technology, some roles may have become obsolete even. For example, the use of a distributed version control system, such as Git, allows the surgeon to maintain all records of system changes. Also, the tool smith does not sound as useful in modern day than in the late 1900s. With the wealth of instantly available information, it may well be faster for the surgeon to find tools and services himself instead of ‘outsourcing’ that work to another individual and waiting for the returns; cut out the middle man, as the saying goes. Otherwise, the remaining members of the team still seems high critical to the success of the whole.

The only modification to the surgical team that feels necessary is an additional role that is higher than the surgeon. The surgeon is acting as the chief architect and the lead engineer, and I believe those two tasks should be separated. I propose the Architect role who is responsible for designing the system as a whole on the highest level. As a database engineering metaphor, the Architect would be responsible for the conceptual data model, and the surgeon would maintain the responsibility for the logical data model. I believe that this will prevent one single person from having total control over the system. It is evident throughout history that a single person with total control can have ghastly outcomes.