ACS 567 - Software Project Management

Spring 2009 - Kalyan Govindu

SWEBOK Knowledge Area : Software Engineering Process

"Some problems with software development lifecycle"

In my experience all projects are different, some more than others. For the sake of experiment, even if you take two identical projects and keep the team constant. Once the team finishes the first project, they have gained experience and/or found an alternative solution to a problem they experienced the first time around. This changes the factors of the projects and hence ultimately the duration of the project has changed.

There are several development methodologies also known as Software Development Life Cycles (SDLC) and new ones with minor adjustments to old ones keep coming up seemingly on a daily basis. The flavors and adaptations are endless and some companies come up with their own home grown solution. This is indicative of the fact that there is no silver bullet when it comes to SDLC, while no surprise there. All the methodologies have their supporters and critics, but none mention two of most important factors that affect a methodology in my belief. These are 1) Formality and 2) Developer skill level.

First, formal level of a methodology can insist on creation of certain documents for even the simplest of projects which is actually counter productive. By sticking to this formal process the few capable developers are forced to spend valuable time creating documents that ultimately might not be necessary for projects of certain size.

Second, all the methodologies that I have come across assume an even skill level throughout the team which is not correct. Some methodologies insist that by following all the steps and producing the right kind of documentation, this skill factor can be mitigated, which I believe is not true either. A piece of document with details is not an equal match to experience and natural talent that some developer posses.

That said, there is definitely value in following a well defined methodology coupled with good requirements gathering, good design and good testing strategy. But, a certain level of flexibility or give should be built into the process for a level of success to be achieved.

Ours is a small services company called Transworks, here in Fort Wayne, primarily in the logistics sector. My team is only 9 developers in all and we recently have loosely adhered to AUP (Agile Unified Process). I say loosely because, besides all the stages that are defined by this methodology, our team has divided up the projects into categories by size and by technology/environment at a high level. So, small project <= 30 hrs, medium <= 180 hrs and large > 180 hrs are the categories by size. And, Web, GUI, Internal, External and Enhancement's are the categories by environment. For each of these categories we have created templates to show which of the agile stages or phases apply to it. We have worked on two projects since implementing this SDLC and the results have been mixed. We as a team are certainly better at estimating the amount of time it would take to complete a certain task and code is better in general because of reviews etc. But the end product satisfactory from the customer has not improve to any degree of measure and the we are still not able to handle scope creeps any better.