Title:
Keeping Software Simple - Part 1 - Introduction / Installation

Word Count:
848

Summary:
Install, Configure, Test, Use, Tune, Maintain

We get so mired in the details we tend to forget the big picture. software that has moved from the development phase to either alpha, beta, or general availability involves six things.

1. Install
2. Configure
3. Test
4. Use
5. Tune
6. Maintain

These items will certainly require iteration. For instance, we typically do not know how to tune something optimally until we begin using it in its intended environment. Then o...

Keywords:
software development, software installation

Article Body:
Install, Configure, Test, Use, Tune, Maintain

We get so mired in the details we tend to forget the big picture. software that has moved from the development phase to either alpha, beta, or general availability involves six things.

1. Install
2. Configure
3. Test
4. Use
5. Tune
6. Maintain

These items will certainly require iteration. For instance, we typically do not know how to tune something optimally until we begin using it in its intended environment. Then once we tune it for a particular environment we use it some more. And of course tuning usually requires going back and tweaking the configuration, which in turn requires us to re-test.

We certainly need to drill down on each of these entities and provide more detail, but before doing so we need to come to consenus that these are the actions items involved in any piece of software.

Without getting too wrapped up for the moment in the "how" let us try to answer the following questions.

First up: Installing Software

QUESTION
Is there any software out there that you know of that does not need to be installed? Somehow it just magically appears and we can begin using it :-). Of course not.

MISSION STATEMENT
With the exception of configuration, we cannot test, use, tune, or maintain software until we have figured out how to install it. So might it be a good idea to put together a procedure for installing the software?

SOME NORMS

* We should always strive to install software in a manner that can be fully automated. No, I did not fall off the turnip truck and realize that this is not achievable for all software. But, if this is not doable you better have some compelling (in writing) reasons why not.

* If we cannot install software via a fully automated process then we should achieve to install software in manner that we could train a chimp to do. In other words the procedure should be so easy to understand that anyone could pick it up and accomplish it. Of course if we can achieve this then we are pretty much back to the first bullet.

* Staffing a configuration management team costs money and resources. The more complex the software the more of this you need. The more complex and/or convoluted your software development methodology is the more of these people you need. Frankly, a good software development methodology should seeks to minimize this function. Sorry, I have nothing against configuration management. It certainly is a needed and vitial function. My point is that it largely exists to

deal with the crap and kludge, and general mess the development staff has lobbed over the fence.

* If your configuration management team is unable to move and manage files, directories, and do builds through an automated manner that can be accomplished by toolsets then it is likely that you have 1) hired a bunch of doofs to serve this role, or 2) are having some degree of crap being lobbed over the fence by your development staff. If you do not like these two choices you had better be prepared to defend in writing, in detail just what the problem is. The goal is reduce variance. By reducing variance we reduce installation complexity. As a trivial but common example to this point I work with a product that in each of last five releases the development staff have relocated the same set of files that make up our runtime application api. They have had absolutely no compelling reason for doing this. Does this causes our configuration management group to have to redo tool and redo deployment procedures and processes. I am not advocating that there is never a reason to do something like this, I am merely stating that if you do you need to have 1) a compelling and bone-fide written reason for doing this (as in an engineering order, or new design requirement); and 2) you had better articulate this up front so everyone can prepare for the impact this change is going to have downstream for everyone that will be accepting these changes.

* For software installation that requires configuration during the installation process itself you will need to ensure that your procedure contains a pre installation requirements section. For example, oftentimes one is required to create a default directory, or set a default location, enter a url, select a communcation protocol, have a password and login, a license key etc, etc... . Make sure the person doing the install has all of this information before they begin. Seldom have I seen where an installation does not actually requires a decision point (a fork in the road). Generally you can provide the answers to these forks in the road up front. For example, installing SQL server askes you to select a communication protocol. This is a decision point that should be known up front and provided to the installer.

* Installing software on different operating systems or different version of an operating system can be quite variant. As a general rule of thumb if the variance is more that 10% (collectively, no more than!!!) create a separate procedure.