What does your Software Development Process do for you?

Graham Stone, Dunstan Thomas Consulting
http://consulting.dthomas.co.uk

Contents

Introduction

In a recent survey it was found that of companies who claimed to use a software development process, 48% of companies followed an in-house process as opposed to a branded process such as RUP (Rational Unified Process), XP (Extreme Programming), DSDM (Dynamic System Development Method), etc. However, no details were provided as to what these in-house processes looked like or what they provided for the users.

In this article, we look at how you can decide whether your in-house process is really a recipe for completing documents or whether you really are on the cutting edge of modern software development techniques.

Capability Maturity Model

There are several ways to assess the sophistication of an organisations software development methods and techniques. Of these, perhaps the most widely used is the Capability Maturity Model or CMM. The following table provides a definition of the various level of sophistication and allows us to measure one organisation against another:

Level	**Description**
1. Initial	Software process is ad hoc and occasionally chaotic. Few processes are defined & success depends upon individual effort and heroics.
2. Repeatable	Basic project management processes are established to track cost, schedule and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3. Defined	The software process for management and development activities is documented, standardised and integrated into the organisation. All projects use an approved, tailored version of the process.
4. Managed	Detailed metrics of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

5. Optimised	Continuous process improvement is enabled by the quantitative feedback from the process and from piloting innovative ideas and technologies.

Table 1

As a rough guide, most software development organisation would like to be at or would be satisfied with level 3; few ever reach Level 5. However, based on observations, most organisations that *do* have a process flip-flop between levels 2 and 1. Those with no process at all would undoubtedly remain at level 1, with the emphasis on individual effort and heroics for any degree of project success.

In assessing an organisation, it would be unusual to find that a particular organisation was, for the sake of argument, CMM level 3 for all aspects of software development. For example, it might be very strong at testing but poor at managing requirements. In other words, it might be CMM 3 for testing but CMM 1 for requirements management. Furthermore, depending on the details of an individual project or customer[1], an organisation might be CMM 3 or one project but CMM 2 on another. It goes without saying, therefore, that it is the average that counts and the constant striving for improvement that makes the real difference.

Another way to assess an organisation might be to measure that organisation against standards such as ISO or against the key quality drivers in published process such as RUP, XP or DSDM. In other words, if one compared an in-house process against another process/standard, one could measure how well that process addressed issues commonly regarded as best practices in the published process/standard. In order to do this, it is probably first necessary to state the objectives of that in-house process and why it exists (the answer to which is more than likely "We've always done things this way in this company"). In doing so, we ought to be able to conclude whether the in-house process is doing what it is supposed to do e.g. drive risk from the project (a key RUP goal) or perhaps improve time to market (a more XP-type goal).

The Rational Unified Process claims to represent the "best practices in software engineering". XP, on the other hand, claims to represent a faster, more efficient way to write software with fewer rules and less documentation. Since these two methodologies probably represent two ends of the current process spectrum, it is probably appropriate to compare your in-house process with these two. The following section highlights the goals of RUP and XP and is written in the form of a check-list. How well does your in-house process measure up? Does it come up short in some areas? Do you notice any adverse effects in real life?

[1] It's a well-known fact that the level of experience or sophistication of the client can greatly influence the development organisation. For instance, if the client refuses to sign-off requirements or state stakeholder needs, the subsequent adverse effect on the project will result in the same effect as if the development organisation lack skills in requirements management.

dunstanthomas

Rational Unified Process – Best Practices in software engineering

RUP Best Practices	Covered by in-house process?
Develop Iteratively	
Manage Requirements	
Use component architectures	
Model Visually (UML)	
Continuously Verify Quality	
Manage Change	

Furthermore, RUP has 9 Disciplines acted out by Roles taking part in the project.

RUP Discipline	Covered by in-house process?
Requirements	
Analysis and Design	
Implementation	
Testing	
Deployment	
Configuration and Change Management	
Project Management	
Environment	

The Rules and Practices of Extreme Programming.

The following tables give a summary of the rules and practices of XP. Naturally, the full version of the process provides significantly more detail than space allows for here. How similar to your in-house process are they?

Planning	Covered by in-house process?
User stories are written.	
Release planning creates the schedule.	
Make frequent small releases.	
The Project Velocity is measured.	
The project is divided into iterations.	
Iteration planning starts each iteration.	
A stand-up meeting starts each day.	

Coding	Covered by in-house process?
The customer is always available.	
Code must be written to agreed standards.	
Code the unit test first.	
All production code is pair programmed.	
Only one pair integrates code at a time.	

Integrate often.	
Use collective code ownership.	
Leave optimization till last.	
No overtime.	

Testing	**Covered by in-house process?**
All code must have unit tests.	
All code must pass all unit tests before it can be released	
When a bug is found tests are created.	
Acceptance tests are run often and the score is Published.	

Summary

Comparing like-with-like is clearly not easy when is comes to software development methodologies. Theoretically, if RUP and XP are to work at all, they must share a considerable amount in common. However, a quick glance at each would suggest otherwise. Don't let this put you off: Look deep enough and you will find the common ground. It's really a question of balance and priorities – RUP will appeal more to "engineers" whilst XP will appeal to "programmers"

So how does your in-house process measure up? Does it contain more in common with RUP or XP? If it contains both *What?* and *How?* it's probably more closely resembles RUP: On the other hand, if it is less prescriptive, you may see more similarities with XP since XP is clearly less formal and, to be fair, more aimed at developers than project managers.

What is perfectly clear is that both of these processes contain aspects that could easily be used in any process – take Pair Programming for example. Although this is not a concept found in RUP, in principle there is no reason why it couldn't be used with great effect. From this perspective, this article is probably useful even if all it is able to do is provide an insight into how others deal with certain problems. Provided your process can accept some changes, you should be able to supplement it with any of the good things found in any other process. After all, that's the whole point of defining "best practices".

Never forget: The aim of the game is to develop software that means the needs of the customer – the process is there to help avoid making mistakes more than once[3].

[2] Perhaps one day there'll be RUXP aimed at…developers? This might help those who cannot abide hacking but feel overwhelmed by formal methodologies.

[3] Developing software is too difficult to avoid making mistakes. The cardinal sin is to make the same mistake more than once, even if someone else made it first!