Technology Selection: Choosing a Software Development Methodology

A methodology defines the work practices in software projects - including how to set technical and organisational direction, achieve project milestones and distribute workloads among the developers. The goal of this technology selection is to choose which software development methodology to use within this organisation.

1.1: What do we really want?

There are four important goals that the chosen methodology needs to help us achieve:

- Responsive to rapidly changing customer demands - Customers are no longer prepared to wait more that a year for a particular piece of software functionality to arrive (anything that takes that long will probably be out of date by the time it arrives). Software teams need to be able to respond to new requirements with updated software in a couple of weeks/months
- High productivity – it is a highly competitive marketplace, and the software engineering team needs to produce as much software as possible. All activities of developers must directly lead to innovative software creation
- Quality driven – Comprehensive testing strategies are needed, at unit, integration and system test levels, so that what is delivered works first time for the customer
- Interesting for developers – As the recent recession fades and skilled developers are again in high demand, it is critical that the organisation evolves policies to keep talented staff. Developers like interesting software engineering and do not like bureaucracy, so the more time that developers can devote to the former and the less to the latter, the better

1.2: What do we already have?

The software engineering team currently has no official documented software engineering methodology. It can be considered to use an “ad-hoc” methodology with some requirements, architecture, coding, manual testing and documentation. There is no consistency in artefacts produced and work practices among the different projects.

Positives that we wish to keep include:

- Close customer contact – developers interact directly with customers, so they build up a good understanding of the customers’ domain and the issues that are important to the customer
- Flexibility to customers’ changing requirements – customers needs are fast changing and

developers strive to absorb the new/altered requirements as they become available

- Dedication to quality – the development team goes to great length to provide a quality product and, though their testing strategy needs to change from manual to automated, this quality focus needs to be maintained

Specific problem areas include:

- Scheduling not hit often – the target dates for project milestones are not been consistently met.
- Issues with quality assurance (QA) – there are no automated unit testing/acceptance testing and sometimes bugs slip through, and manual QA is taking too long
- No/little documentation – As developers move from project to project, it is clear new developers on a project are taking too long to "get up to speed" (part of this is the widely different technologies used on each project – which is not relevant to the methodology used, but part is to do with how the methodology is set up for new project team members)
- Reluctance to change things, in case "we break anything" – on the larger projects, there is a general reluctance to make large architectural changes, for fear of how the existing code base will be affected (this is directly linked to the lack of automated testing)

1.3: What are our realistic options?

The five principal software development methodologies available to us are: "ad-hoc", waterfall, eXtreme Programming (XP), Agile Process and Rational Unified Process (RUP). [Note that we use the generic term "Agile Process" to identify the substantially similar ideas covered in Crystal, Lean, APM, AM, SCRUM , etc.)]

We define "ad-hoc" as the mixture of a bit of everything, with concepts taken from many different methodologies. It is the basis of software engineering in a surprisingly large number of existing software organisations. It does deliver software, but at a higher cost, longer timescales and the resulting software is fragile when subject to changes. It also involved much more complex project management, as there is no documented set of steps to guide the project as it evolves. Using "ad-hoc" techniques is not advisable.

The Waterfall methodology consists of rigid sequential steps, involving requirements, analysis, design, coding, integration and testing. Waterfall was popular in the 1980s and early 1990s but has fallen out of favour now. It is not responsive to rapidly changing customer demand. It has a "late breakage" problem - when the various components are integrated together near the end of the project, serious problems are only then discovered (much too late). It lacks ongoing customer involvement – they participate at the beginning of the project (requirements) and near the end (acceptance testing) but have no role in the middle – often leading to significant work being completed that customers don't actually want. Using the waterfall model is not advisable.

XP, Agile Process and RUP are all part of a continuum of modern methodologies that encourage iterative delivery of value to customers, capturing most requirements as use cases/user stories and

advocate a test-driven approach. We will treat them as three separate approaches, but acknowledge they have plenty in common. Each of these is a realistic option and needs to be considered in more depth.

1.4: Deciding how to decide

The specific selection criteria are as follows:

- Easy to implement – the chosen methodology should require little training and be clear to all how it works
- Low-cost/low infrastructure – we do not wish to have to deploy expensive tools to provision the methodology
- Productivity-enhancing – the methodology should help developers in their work, not become an obstacle with many formal administrative steps
- Quality-enhancing – comprehensive automated testing needed
- Buy-in from developers - the developers themselves must agree it is the best methodology to help them complete software projects
- Practical project management – clear visibility of project status; facilitates transfer of developers between projects (fast ramp-up of developers on new projects)
- Broad industry acceptance – the chosen methodology must have widespread industry acceptance, which will likely lead to an active research community (books, conferences, innovations) and knowledge of the methodology becoming a popular skill among potential recruits

1.5: Weighing the options

The three software development methodologies we need to examine are RUP, XP and Agile Process.

1.5.1: The Rational Unified Process (RUP)

Rational (https://en.wikipedia.org/wiki/Rational_Software) is a software engineering subsidiary of IBM. Rational is promoting a software development process, known as the Unified Process, and a product, known as the Rational Unified Process (RUP) to help developers use the Unified Process. The product consists of detailed information, templates, checklists and guidance on how to engineer each artefact. Most developers who use the Unified Process also have the Rational Unified Process product, so we treat them as synonymous.

The three most important books defining the Unified Process are:

- “The Unified Software Development Process”, Ivar Jacobson / Grady Booch / James Rumbaugh, ISBN: 0-201-57169-2, Addison Wesley, 1999

(the definitive "must read" book on the topic)

- The Rational Unified Process – An Introduction", Philippe Kruchten, ISBN: 0-321-19770-4, Addison Wesley, 2004 (entry-level coverage of topic)

- "The Road to the Unified Process", Ivar Jacobson, ISBN: 0-521-78774-2, SIGS Books, 2000
  (series of interesting essays giving the backup on the evolution of the Unified Process)

Positives about RUP are:

- Backed by a substantial company that will still be there in ten years time

- Successful in large engineering environments

- Successful where safety issues/high-cost engineering is involved

- Extremely comprehensive

- Availability of RUP product

Negatives about RUP are:

- Too many steps and too many job roles – leads to micro-management of project work

- Considered a high-ceremony process, with lots of methodology (can get in the way of the main goal – writing software)

- Costs - and this is just for the methodology information – it is not a full blown integrated development environment. (Rational also sells these, for a few thousand dollars a seat). The cost factor on it own is not that great- the problem is the other methodologies, XP and agile process – are free, and hence far more people are using them. This is important when new people are being added to the team or when projects are being outsourced and suppliers will be charging us for any extra work in learning non-popular methodologies

1.5.2: Extreme Programming (XP)

XP refocuses the developer on programming and minimises "ceremony" concerning other aspects of software projects. It includes the following strategies:

- Planning Strategy – The "XP Planning Game"; advising on the exploration (building stories), commitment (agreeing to the scoping and scheduling of releases) and steering (keeping plan up to date with reality and guiding development) phases.

- Development Strategy – Continuous integration (e.g. every few hours) leads to regular new builds with more functionality

- Design Strategy – emergent design encourages simple design (design just for today)

- Testing Strategy – Tests describe how the system should (and should not) work. It is obvious they should be written before the code (as they clearly indicate what we need to

code).

The definitive guide to XP is “extreme programming explained”, Kent Beck, ISBN: 0-201-61641-6 Addison Wesley, 2000 (the white book). Another good book on the topic is “Planning Extreme Programming”, Kent Beck / Martin Fowler, ISBN: 0-201-71091-9, Addison Wesley, 2001 (the green book). A good book on user stories (how XP tackles requirements) is “User Stories Applied”, Mike Cohn, ISBN: 0-321-20568-5, Addison Wesley, 2004

Positives about XP are:

- Significantly reduces ceremony
- For projects that are not fixed scope (the set of requirements can be prioritised, and only the most important must be completed), XP can lead to very dynamic delivery of functionality
- The first methodology to bring unit testing to the fore – the role of testing (both unit and acceptance) is well thought out
- Very close user involvement (a sample user is on the development team) ensures the code as it evolves genuinely satisfies actual user requirements

Negatives about XP are:

- XP is not suitable for fixed price/fixed scope projects, because scope (as in detailed requirements) only becomes known as the user stories are implemented
- Very low tolerance to tuning by participants – if they do not adhere to all practices in XP (pair programming, refactoring, collective ownership, etc.), likely to end up with an un-successful project
- Carried to its extreme, the only thing that results from an XP software project is the code – which is fine for the current release, but over time as developers leave and new developers come on board, this becomes an obvious problem (“here is a million lines of code” is not an appropriate start for a new developer on a project team)

An interesting book that discusses problems with XP practices is “Extreme Programming Refactored: The Case Against XP”, Matt Stephens / Doug Rosenberg, ISBN: 1-59059-096-1, Apress, 2003 (we would not agree with all that is in this book, but it certainly highlights what can go wrong with XP projects).

Agile Process

We use the generic term Agile Process to denote the growing family of methodologies that emphasises a lightweight but sufficient approach to running software development projects. The goals of agile are succinctly described in the Agile Manifesto at http://www.agilemanifesto.org/principles.html. When considering software development methodologies, we use the term “ceremony” to refer to how much organisational artefacts (meetings, documents, team member roles, etc.) are required. Agile can be considered a continuum – (measured by how much “ceremony” is used) - with XP at one end (minimum ceremony) and RUP done in a certain way at the other extreme. There are a number of variations of mainstream

agile that are fall between these two, with sufficient attention to ceremony so that projects succeed, but not too much that would hinder productivity. Though they have different names (Crystal, Agile Modelling, Agile Project Management, Lean, Scrum) they are substantially similar in concept – we call them all by the term Agile Process – which we define as something more than XP and less than RUP (though acknowledge there is overlap). Good books include: "Crystal Clear: A Human-Powered Methodology for Small Teams", Alistair Cockburn, ISBN: 0201699478, Addison Wesley, October, 2004; "Lean Software Development", Mary & Tom Poppendieck, ISBN: 0-321-15078-3, Addison Wesley, 2003; "Agile Project Management", Jim Highsmith, ISBN: 0-321-21977-5, Addison Wesley, 2004; "Agile Project Management with Scrum", Ken Schwaber, ISBN: 0-7356-1993-X, Microsoft Press, 2004. "Agile Modeling", Scott Ambler, ISBN: 0-471-20282-7, Wiley, 2002.

Cockburn states the key agile techniques:

- "Seat people close together, communicating frequently and with good will;
- Get most of the bureaucracy out of their way and let them design;
- Get a real user directly involved;
- Have a good automated regression test suite available;
- Produce shippable functionality early and often."

The advantages of the Agile Process:

- Allows methodology to be customised for specific projects – you may pick and choose from the recommended actions (e.g. if you need fixed cost/fixed scope, then full blown requirements are needed, otherwise lighter user cases will suffice)
- Encourages initial design with ongoing rearchitecture – we can use what we know at the beginning of the project to come up with an appropriate initial design, and evolve this through the iterations
- Provides sufficient design documentation – for experimental projects, perhaps little or none; for larger production projects, need more consideration of subsequent developers
- A high-productivity approach that delivers real business value early and often

The disadvantages of the Agile Process are:

- More responsibility passed to developers to do the right thing – developers experienced with older methodologies will be used to being told in minute detail what to do at every stage – with agile, there is more leeway
- Needs local customisation – agile is not a rigid set of operations that suit every project all the time – its it flexible approach that needs some work to implement correctly for a specific company and project type
- It is a new approach – large-scale success stories have yet to appear in great numbers

1.6: Decision

Considering all the issues, we decide to select Agile Process as the most appropriate software development methodology. RUP can be considered a high-ceremony approach; XP as a low/no-ceremony approach (that is not tolerant of local tuning and does not set up for the next project) and the Agile Process is in the middle as a sufficient-ceremony approach.