Requirements and Analysis: Techniques and Tools (article)

The Requirements Discipline

Requirements Drive Development: A Use Case-driven Process

> *As stated in previous posts and in articles like Real World Development Practices: RUP and XP , I apply much of Craig Larman's UP style and its emphasis on rightsized, "essential" use cases, which then collectively act as a lynch pin that links together the disparate disciplines of Business Modeling, Requirements, Analysis and Design, Implementation, Test and Project Management. Furthermore, achieving success with use cases is more difficult than it first appears, and many pitfalls in usage await the inexperienced practitioner. Consistent application of the techniques espoused by Alistair Cockburn's de facto standard for specifying use cases and structuring them in relation to goals, which provides a repeatable, traceable discipline for use case development and maintenance.*

Executable Requirements: Aligning Requirements and Development

> *These days I particularly like the idea of 'Executable Requirements' (XR) to capture requirements. This approach has the benefit of not only enabling the Pull method described above but they also ensure that software developed matches the specifications provideds. XR bsaically provides a mechanism where a requirement is captured in a 'pass/fail' style using an Excel or HTML table to define the*

requirements. The power of this approach is that it not only moves requirements out of the fuzzy, prose style that can plague use cases (and which is why use cases have so many sections) but also allow a team to automate a series of tests that demonstrate that a requirement has been 'fulfilled'. For those of us with a testing orientation we can immediately see the opportunity to regress through all of our tests every iteration and ensure that new changes don't break old functionality. There's a lot to this subject and something that I'll update on more in the future but there are some good reference sources for this such as the Fitnesse wiki and Ward Cunningham's Functional Integration Testing (FIT) Framework.

Managing Risk and Non-functional Requirements (ATAM, EVO)

Addressing Non-Functional or Supplementary Specifications is often a neglected component of software development. Notable references in this area are Tom Gilb's iterative EVO method, which emphasizes full and careful definition of non-functional requirements (which Gilb calls "attributes" leveraging his Planguage approach) and SEI's ATAM (Architecture Tradeoff Analysis Method) methodology. Documentation of all significant architectural decisions – a component of the ATAM approach – as a key mechanism for reasoning about and justifying choices between architectural options. This fits well with leveraging risk analysis as a major driver of iteration plans.

Early, Continuous Delivery of Business Value: Complementing the Risk Driver

The agile methods complement UP by providing an important emphasis, not only on risk reduction, but also on the early and continuous delivery of business value. Hence, a full iterative development discipline has two drivers: delivery of useful functionality and management of risks. The use case-

driven approach, when combined with non-functional drivers and the dispatching of work into developer tasks provides tangible evidence of progress to the business at each iteration's end. (See some of the XP, EVO, and FDD links for further details.)

The Analysis Discipline

From Use Cases to Developer Tasks

The Larman method takes analysts and designers through a series of simple intermediate steps leading up to operation contracts on a system or service level interface. In accordance with Agile Modeling [below], intermediate artifacts need neither be formally developed nor maintained if the ceremony level of the process does not warrant it. I also believe strongly in a "pull"-driven approach to developer task definition, a key element in Lean Programming.

Applying Analysis Patterns to Streamline Design

I encourage analysts to leverage Martin Fowler's Analysis Patterns, rather than reinvent the wheel. This emphasis provides synergy with the product line process mentioned later, and also opens the analysis up to alignment with standardized vertical models such as well defined reference models (e.g. Insurance Application Architecture). Another useful source of such patterns is Penker and Eriksson's book.

Discipline by Discipline: Requirements

As many who follow my blog entries and have read my articles know, I use the Unified Process as framework to manage projects and programs. While the phases of the UP (Inception, Elaboration, Construction and Transition) are powerful ways to manage the risk and narrow the ‘cone of uncertainty‘ of a project, I find the disciplines within the Unified Process as useful containers for ensuring roles are established and that artifacts are being developed that will support the project.

However, beyond the phases and disciplines I find most of the artifacts and activities as too abstract for effective application in most real world projects. Instead, I mix in a series of techniques that I have applied successfully and found round out the details of each of the disciplines with RUP. This first article focuses on the top of the “V” model, Requirements and Analysis.

Read More