Just-in-Time Requirements Analysis – The Engine that Drives the Planning Game

Michael Lee
Managing Partner
Kuvera Enterprise Solutions, Inc.
1750 30th Street
Suite 186
Boulder, CO 80301 USA
+1 303 638 7728
mike@kuvera.com

Abstract

Just-in-Time Requirements Analysis is an alternative approach to analysis that perfectly compliments XP development and planning. Rather than analyzing and defining requirements up front, Just-in-Time Requirements Analysis defines requirements only when they are needed – and only at the detailed required. It is an iterative process that expects and embraces change and makes it easy for requirements to evolve over time.

This paper details the concepts behind Just-in-Time Requirements Analysis (JITRA) and identifies the benefits of using JITRA on XP Projects. It outlines how JITRA is implemented in XP development and shows how JITRA fits into XP planning.

Keywords

Just-in-Time Requirements Analysis, JITRA, Planning Game, Planning Strategy, Iteration Planning, User Stories, XP Analysis, Incremental Change, Continuous Learning, Adaptive Analysis, Agile Processes.

1 INTRODUCTION

Traditional analysis processes do not work well with XP [1]. Requirements are analyzed and defined up front and then baselined before development begins. One requirement is not given priority over any other, and often requirements will sit for months and even years before being implemented. Elaborate change management processes are put in place to guard against change and it is very hard to evolve requirements over time to meet changing business needs or advances in technology.

Just-in-Time Requirements Analysis is the antithesis of traditional analysis. It defines a process where requirements are continuously analyzed and defined throughout the life cycle of a project. Requirements are only analyzed and defined when they are needed for planning or development – and only at the level of detail required.

JITRA starts the analysis process with a broad, overall set of requirements. These requirements are then iteratively refined into detailed requirements as they are needed – and only when they are needed. This allows development to begin with incomplete requirements and also provides a mechanism for incorporating feedback from actual development into analysis. The end result is a shorter project life cycle, better requirements, less risk, and an evolving baseline that meets the changing business needs of customer.

2 JUST-IN-TIME REQUIREMENTS ANALYSIS – WHAT IS IT?

Just-in-Time Requirements Analysis is a lightweight, adaptable approach to requirements analysis that complements the basic principles of XP. It is an analysis process that expects and embraces change and is distinguished from other analysis methodologies in several ways:

- Requirements aren't analyzed or defined until they are needed.
- Only a small initial investment is required at the start.
- Development is allowed to begin with incomplete requirements.
- Analysis and requirements definition is continuous throughout the project.
- Requirements are continuously refined as the project moves forward.
- Change is expected and easy to incorporate into requirements.
- Analysis tasks compliment XP planning.

3 DON'T DEFINE IT UNTIL YOU NEED IT (DDIUYNT)

Although it makes for a terrible acronym, *Don't Define It Until You Need It* is analogous to XP's YAGNI (You Aren't Gonna Need It) [2], and is the fundamental principle of Just-in-Time Requirements Analysis. It stands

in stark contrast to the monolithic approach of traditional analysis and has three distinct advantages:

- Requirements that are defined closer to implementation are better requirements.
- Requirements that have not been defined can not change.
- Requirements can change incrementally as needed.

Defining Better Requirements

As we develop, we learn more about the problem and what is required to solve the problem [3]. With traditional analysis, this knowledge isn't available because none of the work leading to the knowledge has been done yet. With JITRA, however, knowledge and understanding gained from actual development *can* be incorporated into requirements – which naturally leads to better requirements.

Eliminating "Phantom Change"

On any software development project there are two types of requirement changes: "Real" requirements change that necessitate actual modifications to implemented software; and "Phantom" requirements change that only necessitate changes to baseline requirements that have not yet been implemented. With traditional analysis, both Real and Phantom changes have an associated cost. With JITRA, however, we can eliminate most – if not all – of the cost associated with Phantom Change.

Phantom Change is a direct result of an up front analysis process where the result of the analysis – the baseline requirements – are used to define the scope of a project and estimate cost. Any changes to this baseline are then measured against the initial estimate. The delta between the first estimate and the new estimate is the cost associated with Phantom Change.

With JITRA there is no phantom baseline to change. Only implemented requirements are baselined and by deferring requirements definition until the point of implementation, JITRA ensures that the initial requirements definition is the one that gets implemented.

Incremental Change

Change is inevitable and a good analysis approach must address this fact. Most traditional approaches abhor change and have put in place elaborate change management procedures to limit and control change.

With JITRA changes only apply to real baseline requirements (i.e. requirements than have been implemented) and all changes are treated as new requirements going forward. This allows changes to be addressed on their own merits and handled just like any other requirement.

Shorter Projects and Decreased Time-to-Market

JITRA eliminates the long analysis phase at the start of a project. Instead, analysis is spread out across the entire project life cycle and is concurrent with development. In most cases – and especially on big projects - this significantly reduces the time it takes to complete development.

Reduced Risk

JITRA uses an evolutionary approach to requirements analysis and evolutionary approaches reduce risk [4]. With JITRA only the minimum amount of analysis is performed before development begins. This allows managers and senior executives to allocate their investment where it matters most – to actual development. It also allows decision makers to focus on the most important things first and arrive at meaningful check points much earlier.

4 THE JITRA PROCESS

Just-in-Time Requirements Analysis is a simple process. Analysis starts at the highest levels of abstraction and requirements are continuously refined over the life cycle of the project. The most important things are analyzed first, and the analysis is always at the level required to meet the current needs of the project – no more and no less.

To support this process, JITRA defines four major analysis activities:

- Initial Analysis
- Feature Set Analysis
- Story Analysis
- After-action Analysis

These activities are performed continuously throughout a project's life cycle, and may over lap each other in scope and detail. The following paragraphs discuss each of these activities.

Initial Analysis

The Initial Analysis activity is performed at the start of a new unit of work. This unit may be an individual system, a system of systems, a subordinate subsystem, or individual subsystem components. The purpose of this activity is to broadly define the scope for the work ahead and to specify an initial set of features, functions, and capabilities required for the specified unit of work.

Initial Analysis Tasks

Defining Scope – The first task of Initial Analysis is to broadly define the scope of the work ahead. This allows planners to estimate the level of effort required to complete the work. On some projects this take may take a few hours or days. On other projects – especially on those projects where a detailed estimate is required (i.e. fixed price projects) – this may take longer.

Gain the required understanding so follow-on activities can move forward – The second task of Initial Analysis is to gain a better understanding of the current problem domain. This doesn't have to be a detailed understanding, but as a minimum the team must have a reasonable

expectation of success if the project moves forward.

Develop an initial set of required features, functions, and capabilities – The final task of Initial Analysis is to define a generalized list of things the new system needs to do. This list might be a simple bullet list, or it might be detailed in one or more high-level Stories.

Scaling Initial Analysis

The Initial Analysis activity is highly scalable and works extremely well on complex projects – especially those involving multiple teams and organizations.

The key to this scalability is the recursive definition of the unit of work in Initial Analysis. At the top-level, a unit of work may specify a system of systems. Initial Analysis is performed at the top level and then the unit of work is partitioned into one or more subordinate units of work (individual systems).

This hierarchical decomposition is recursive and may continue down any number of levels. At each level an initial analysis is performed and the unit of work is either partitioned into subordinate units of work, or Initial Analysis is completed and Feature Set Analysis is begun.

Output of Initial Analysis

The output of Initial Analysis is the initial set of features, functions, and capabilities for the unit of work. This may be formally documented (recommended on large or complex projects), or more loosely defined. The set may be communicated as a list, or it may be built into one or more high-level User Stories.

Feature Set Analysis

Feature Set Analysis (FSA) is performed continuously throughout the life cycle of a project. It's purpose to build User Stories that feed into iteration planning and into individual iteration development. Most of the analysis effort on any project is performed as part of FSA, and it is the heart of the JITRA process.

Feature Set Analysis Tasks

Prioritization of features, functions and capabilities - The list of features, functions, and capabilities defined during Initial Analysis feeds the analysis of Feature Sets. At the start of Feature Set Analysis, all of the features, functions, and capabilities that have not yet been implemented (or which have only been partially implemented) are reviewed and prioritized by the Requirements Stakeholders. This process allows stakeholders – and not developers - to identify what parts of the system get the most focus, and always ensures that the most important part of the system will be analyzed and developed next.

Selection of a Feature Set - Once the remaining features, functions, and capabilities have been prioritized, the development team groups the highest priority items into a *Feature Set*. A Feature Set is nothing more than a grouping of the features, functions, and capabilities that the development team estimates can be analyzed and implemented in a small number of iterations (typically 2 – 4).

Analysis and Definition of User Stories – The primary task of FSA is the analysis and definition of User Stories. After the Feature Set has been selected, Domain Experts and Business Analysts – supported by Requirements Stakeholders and the development team - perform a detailed analysis of the items in the Feature Set. This analysis is used to build and define individual User Stories that feed into iteration planning and actual development.

Scaling Feature Set Analysis

Scalability of Feature Set Analysis is not an issue. The size of a Feature Set is under the control of the development team and is always determined by how much of the remaining system can be analyzed and implemented in the next few iterations.

Output of Feature Set Analysis

The output of Feature Set Analysis are groups of User Stories. These User Stories should be detailed enough to allow follow-on iteration planning, yet they do not have to be detailed enough to implement (although they may be).

Selecting a Feature Set that is too Big

In a some cases, the team may discover that the selected Feature Set is too big to fit into a few iterations. In these situations, simply focus on the highest priority items first, and return the unanalyzed items to the list of unimplemented features, functions, and capabilities. These items will then be addressed in follow-on Feature Set Analysis.

Story Analysis

As part of iteration planning, User Stories are allocated to specific iterations. Story Analysis is then performed as part of each iteration for every Story being developed. The purpose of Story analysis is to finalize the details of each Story in the iteration and to baseline the Story at the completion of the Iteration.

Story Analysis Tasks

There is only one relevant task for this activity – finalizing User Stories. How this task is performed, however, may vary widely from project to project and is designed to be tailored to meet the specific needs of an organization.

Splitting User Stories

During Story Analysis the team may find it necessary to split one or more Stories allocated to the current iteration. In this case, the new Stories may be allocated to the current iteration, or they may be added back to the current Feature Set for implementation in follow-on iterations. The decision is left up to the Requirements Stakeholders.

After-Action Analysis

At the completion of each iteration, an After-Action Analysis is performed. This allows "lessons learned" from the previous iterations to be included in the analysis process of subsequent iterations.

As part of After-Action Analysis, new requirements may be defined. These new requirements may identify new or modified features or they may specify changes to features that have already been implemented. Regardless of the form the changes may take, new requirements are fed back into the JITRA process at the appropriate level.

5 JITRA AND THE PLANNING GAME

> "*We will plan by quickly making an overall plan, then refining it further and further on shorter and shorter time horizons – years, months, weeks, days. We will make the plan quickly and cheaply, so there will be little inertia when we must change it*." [5]

Requirements Analysis drives all planning, but the approach used for requirements analysis must match the approach to planning. For projects relying on up front planning, an up front analysis approach is required. But for a flexible, adaptive, and incremental approach to planning that evolves over time, we need an analysis approach that is also flexible, adaptive, incremental and evolves over time. Just-in-Time Requirements Analysis is such an approach and it perfectly compliments the XP Planning Cycle

How JITRA Compliments XP Planning

Initial Analysis - The Initial Analysis activity forces the team to quickly identify scope and define an initial set of high-level requirements. This allows decision makers to rapidly develop a broad overall plan that is expected to be tailored as subsequent analysis proceeds.

Don't Define It Until You Need It – A core principle in XP planning is that you only plan for what you need for the next horizon[4]. This principle is complimented by the core principle of JITRA – Don't Define It Until You Need It. This principle allows work to begin on the important parts of a system even if other areas have yet to be analyzed or defined. This greatly aids in XP planning because it allows planners to focus only on what is needed at the moment – not what may be needed in the future.

Feature Set Analysis – Feature Set Analysis provides XP planners with a grouping of User Stories (the Feature Set) that feed Iteration planning. During iteration planning, decision makers select User Stories from the current Feature Set and allocate them to an iteration. Feature Set Analysis ensure that planners always have a current set of User Stories to select from, yet it doesn't require a complete analysis of an entire system.

After-Action Analysis – After-Action Analysis allows requirements to evolve over time and allows new and modified requirements to be injected into the analysis process at the appropriate level. This gives planners the flexibility of prioritizing new or changing requirements and allows them to adapt the plan to reflect an evolving set of requirements.

Splitting User Stories – JITRA supports the splitting of User Stories at any time. New Stories following the split can be allocated to the current iteration or added back to the current Feature Set or a future Feature Set. This forces planners to always focus on the most important User Stories at any give time and to defer work on less important User Stories.

6 SUMMARY

Just-in-Time Requirements analysis significantly reduces project risk and shortens development time. It ensures the most important parts of a system – as defined by the business stakeholders - are being worked on at any given point in time and only defines requirements when they are needed. It supports the evolution of requirements and provides mechanisms for easily incorporating changes into the analysis process. In short, Just-in-Time Requirements Analysis matches the vision and promise of XP and perfectly compliments the XP approach.

ACKNOWLEDGMENTS

The author wishes to acknowledge the support and help of Kuvera's clients in the development of Just-in-Time Requirements Analysis. These clients provide timely feedback on the use of JITRA on business-critical projects and allowed the author to refine many of the concepts presented in this paper.

REFERENCES

1. McBreen, Pete: *Incremental Requirements Capture*. XP Magazine online. http://www.xprogramming.com/xpmag/incremental_req1.htm
2. Auer K. and Miller R.: *Extreme Programming Applied – Playing to Win*. Addison-Wesley. 2001.
3. Beck, Kent: Extreme *Programming Explained – Embrace Change*. Addison-Wesley. 1999.
4. Highsmith, Jim: *Agile Methodologies: Problems, Principles, and Practices*. XP2001 Conference. http://www.xp2001.org/xp2001/conference/Details/AgileMethodologiesXP2001.pdf
5. Beck, Kent: Extreme *Programming Explained – Embrace Change*. Addison-Wesley. 1999.