Whose View is it Anyway?
Addressing Multiple Stakeholder Concerns

Summary

The debate around prioritization of stories based on business v/s technical value is probably too skewed. Functional stories get priority over the quality attributes of a system such as maintainability & modifiability, which get pushed out to later releases leading to expensive refactoring. In this session, we present a method that addresses multiple stakeholder concerns and transforms these into features realizing business value, and how to organize these stakeholders’ perspectives into a release plan.

Background

Any software system consists of different kinds of abilities, Functionality being one of them. Other abilities include its modifiability, configurability, maintainability, performance etc. The general tendency is to develop the functional abilities first, while leaving these for later iterations.

The challenges in addressing these other abilities (or quality attributes) of the system are:

1. It is difficult to turn them into stories. For example, how do we create a story to build modifiability into the system?
2. How and when do we say that the story is "done" - e.g. when do we say that modifiability is done, tested and ready?
3. How do we associate business value with these stories, so that we can justify the cost of building them?

4. In which iteration or release do we place these stories? When do we say that we are building modifiability into the system?

This session addresses some of these questions by introducing a method to turn these quality attributes into INVEST-able stories, and use of multiple story maps (views) to help organize them in a release plan.

Supportability

Let us start with the story template - "As a [role] I want [feature], so that [value]". If we take manageability, and try to turn it into a story, the obvious question is - who is the stakeholder? Who are the stakeholders in our system who want manageability. If we can do that, then we can also ask them to define what it means to them. For example to the operations team, it might mean that the system should support SNMP protocol so that they can manage the system using tools they are already familiar with. The business value to them comes from the fact that this was a productivity concern for them. By having SNMP support, it could save them $X per month on operational cost.

It turns out that many of the quality attributes are actually productivity related concerns of some stakeholders or the other. Chances are, that these stakeholders were not active end users of the system, such as the operational and development stakeholders. As we identify them, we articulate exactly what feature they want in the system to address their productivity concern, and the business value they derive out of it.
In this part of the session, we will take a few examples and illustrate the method of supportability to convert these other abilities of the system into concrete user stories that can be placed in the backlog and iteratively built.

**Benefits of this approach are:**

1. Other stakeholders who are affected by the system, (but did not have a say in how it is built) are converted into active users of the system. The system now has some features which are meant specifically for them, and they are sponsors for these.

2. The ambiguous quality attributes are converted into concrete stories that can be taken up for development.

3. Since each such story is basically a feature into the system, its progress can be tracked, it can be tested using conventional QA techniques and we can say when it is done.

4. Team can estimate the effort, and since the business value is also articulated, we can decide when to build it. We can now say that the story titled "Support for SNMP" will be built in iteration #4, rather than saying "modifiability" will be built in iteration #4.

There are of course some quality attributes which are actually constraints on functionality, such as performance, or fault-tolerance. These can be specified as part of acceptance criteria of the concerned functional story. The architecture frameworks that need to be built to ensure that these constraints are met (e.g. cache) can be specified as technical stories on behalf of the architect.

**Multiple Views - Smaller Story Maps**

We still haven't answered the question - "when do we take these other stories up for development?" As we discussed earlier, the tendency is to develop the functional stories first, pushing these for later iterations/releases. The consequence is, building these into the system later will require a lot of effort in new development and refactoring, regression testing and so on. This is in a sense similar to the accrual of technical debt.
If these stories are developed incrementally throughout the iterations/releases, it results in a balanced system that is sound on all its abilities, not just functionality.

So the question is - how do we organize and balance the stories? A single large story map will find it difficult to slice the stories evenly. We found it useful to arrange these into multiple smaller story maps. Each story map is a collection of stories that are relevant from a particular stakeholder's (or a group of stakeholders) point of view. In each view, the stories are sliced into releases. Across views, the stories that are part of the same release form a coherent group, the overall ability of that particular release (functional and quality).

The concept of views comes from architecture views (& viewpoints). Each view frames a set of concerns that it addresses. Picking the right set of views serves as a framework to the architects & other stakeholders that they have covered all relevant aspects of the design. Likewise, these views serve as a checklist to ensure that we have covered all stories across various stakeholders of the system. The specific set of views to be used generally depends upon the project - the product owners are most likely to be the people who would own that responsibility. The table below presents a collection of views and the type of stories that would be part of the view.

<table>
<thead>
<tr>
<th>VIEW</th>
<th>DESCRIPTION</th>
<th>EXAMPLE STORIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional View</td>
<td>Contains the functional stories of the system, primarily relating to the end-users</td>
<td>Check account balance, Funds transfer etc.</td>
</tr>
<tr>
<td>Operational View</td>
<td>Contains the stories of operational stakeholders such as NOC center</td>
<td>Creating a new branch, migrating the data, data backups and restore scripts</td>
</tr>
</tbody>
</table>
**Development View**
- Contains the stories of development stakeholders such as CTO/CIO, architect, maintenance teams, project manager
- Transaction logs for debuggability, Replication for reporting database, Archival and maintenance plans, Build automation

**Events View**
- Important dates, events and milestones and stories related to them
- Upcoming demos, Demo data generation, Proposal for Agile India conference etc.

### Benefits of this approach are:

1. A carefully chosen set of views serves as a checklist to ensure that all stakeholders’ concerns are covered in the roadmap
2. Each view draws attention to that particular set of concerns - and makes it easy to balance them in the roadmap
3. Serves as a peek into the project for the concerned stakeholders, they can see how they will be affected by it
4. Such views can go beyond development stories - e.g. they can include events, conferences and other activities
5. Since each view is essentially a story map, it offers the same flexibility and agility in evolution of the system

In the session, we will take a few examples on how to create such views, and share our experiences in using them across different types of projects, challenges we faced and the learnings from there.

<table>
<thead>
<tr>
<th>VIEW</th>
<th>DESCRIPTION</th>
<th>EXAMPLE STORIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development View</td>
<td>Contains the stories of development stakeholders such as CTO/CIO, architect, maintenance teams, project manager</td>
<td>Transaction logs for debuggability, Replication for reporting database, Archival and maintenance plans, Build automation</td>
</tr>
<tr>
<td>Events View</td>
<td>Important dates, events and milestones and stories related to them</td>
<td>Upcoming demos, Demo data generation, Proposal for Agile India conference etc.</td>
</tr>
</tbody>
</table>
Session Plan

This session focuses on two simple techniques, presented mostly in the form of a dialogue between multiple stakeholders of a project, trying to get their view across. The session is organized into two parts:

1. **Supportability [30 mins]** - We illustrate how to use supportability framework for translating concerns of other stakeholders of the system such as operational and development teams, relating to quality attributes such as maintainability and modifiability and so on into concrete features and user stories.

2. **Using Multiple Views [30 mins]** - We present our experiences in using multiple views (smaller story maps) to manage planning complexity of a typical large scale project, from different stakeholders’ perspectives. We discuss what worked for us and what did not, and present examples of these views.

During the session, we will use a bottoms-up approach - we will start with participant exercises, leading up to the concept of supportability before we present the method.

Learning Outcomes

1. Articulate business value of architectural “abilities” such as maintainability, manageability, extensibility and so on

2. Identify features and stories that support the other stakeholders of the system such as operational and development teams

3. Organize the release plan into smaller story maps (views) that frame the concerns of respective stakeholders

4. Choose a set of views so that it serves as a checklist that we have captured other stakeholders concerns as well
Speaker Profiles

**Feroz Sheikh**

Feroz has more than 11 years of experience in the IT Industry. He is an architect and a polyglot programmer. He has held various technology, development and support positions.

He was the CTO of Veloz Global Solutions, India, a pioneering software product development startup, where he successfully followed Agile practices for product development over a multi-year roadmap spanning 5 years. He is a speaker at industry forums such as the IBM Software Universe and TiE Bangalore chapter. He is the co-author of the paper on Supportability Framework published at WICSA 2011.

He is a consulting architect at Canopus Consulting, Bangalore.

He can be reached at [feroz@canopusconsulting.com](mailto:feroz@canopusconsulting.com)

**Nagaraju Pappu**

Nagaraju has about 20 years of experience in both academics and industry. He held various research and engineering positions both in India and in USA. He started his career as a research engineer at IIT-Kanpur and he was later with Oracle India, Fujitsu Software Corporation, InfoDream as senior engineer and architect. He specializes in large scale transaction processing, semantic computing and software architecture. He is a pioneer in cultural informatics in India.

Nagaraju has several patents and publications to his credit. His papers are published in top-tier international journals and conferences. He has conducted several public and academic courses and is a highly sought after speaker at CSI events, CII and other industry forums.

He is a visiting faculty to IIIT-Hyderabad and IIT-Kanpur. He is the co-founder and chief solutions architect at Canopus Consulting.

He can be reached at [pnr@canopusconsulting.com](mailto:pnr@canopusconsulting.com)