



Victor Szalvay
In 2000, Victor Szalvay co-founded Danube Technologies, Inc., a Bellevue, Washington-based software consulting company focused on software process training and coaching. A Certified Scrum Trainer with experience implementing Agile frameworks for software development teams in the public and private sector, he currently serves as Product Owner for Danube's flagship product, ScrumWorks™ Pro.



Danube Technologies, Inc.
520 SW 6th Ave, Suite 940
Portland, OR 97204 USA
1.888.5.danube
T: +1.503.248.0800
F: +1.503.248.0801

Hierarchical Requirements and Scrum

Posted by Victor Szalvay on December 8, 2006 at 10:47 a.m.

As the Product Owner of [ScrumWorks](#), I'm often asked why ScrumWorks does not support hierarchical nesting of backlog items (i.e. user stories, PBIs, etc.). The answer is simple, but highlights some profound differences in the way requirements are managed in Scrum and Agile as opposed to other methods.

The short answer is *prioritization*. Scrum is predicated on the fact that all work is listed in a linear, one-dimensional list and prioritized relative to other items on that list (see [Agile Software Development with Scrum](#) for more on the Product Backlog). The relative priority of items is always clear and there are no confusing situations like two backlog items with the same priority (e.g. #1!) that both need to be done (e.g. now!). Rather, the Product Owner has the hard job of determining which items are more important than others.

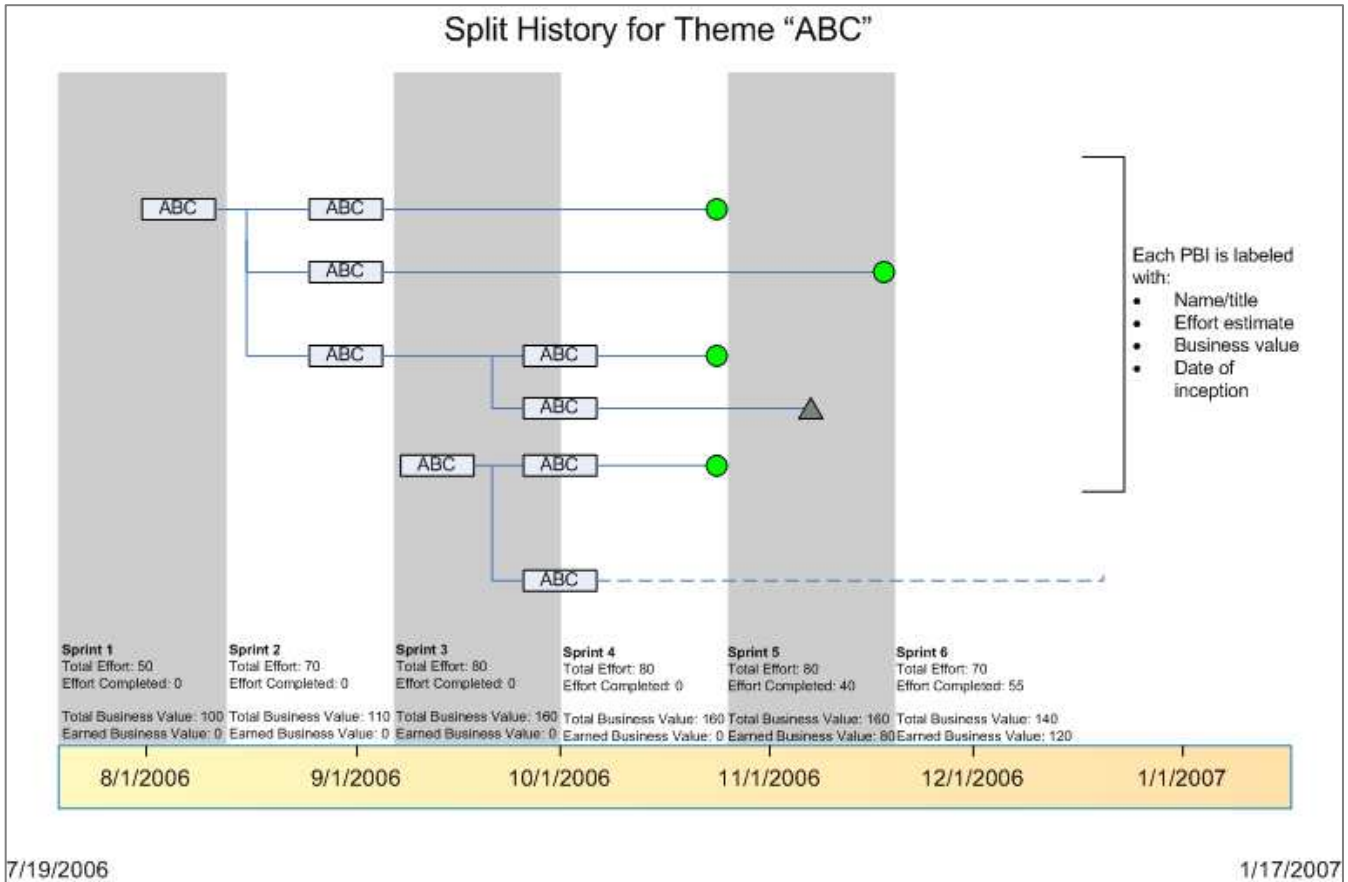
So why can't backlog items be organized into tree-like hierarchies? You can easily prioritize the top level item, but often the top-level of the hierarchy represents a large amount of effort — so much so that the item itself may span multiple sprints or even several releases. So how does one prioritize such a large item that will be done over several sprints and possibly as part of several releases? The problem is that your hierarchy actually contains individual backlog items with differing priorities. Some might be completed in the first few sprints, while others are not as important (gold-plating) and might be done in a later sprint or even a later version of the product. In this case, the hierarchy is obscuring the true priority of the individual items.

There is a second reason why taking a hierarchical approach to requirements is less desirable in Scrum: It encourages Product Owners/Managers to prematurely define detailed requirements. With a hierarchical system, the Product Owner is encouraged to break work down into very granular detail long before the feature defined in the requirement is needed in the product. This runs counter to the lean principle that champions a just-in-time approach to requirements, thereby encouraging Product Owners to define the real requirements only when they are needed for a sprint.

Scrum and Agile propose a new paradigm: Namely, to use large, high-level backlog items (often called *Epics*) and peel off smaller items from that large Epic as the need arises and priority demands. Pretty soon, your Epic dissolves (in terms of effort points) because it is split off into many smaller items. This is why CSM courses teach that you not to split your backlog items into very granular items until they become a priority for the product. Otherwise, you will have an unwieldy backlog. Not to mention that this kind of rigid forecasting is not very Agile/lean: You're spending a lot of time and effort detailing requirements you may never actually need or build. Instead, keep distant, low priority items as large Epics so that your backlog remains manageable.

How does ScrumWorks™ handle categorization? Themes represent a free tagging system that allows users to apply keywords, or Themes, to individual backlog

items. Themes are meant to keep the association between split off items and the original Epic. Simply apply a Theme to all decomposed items to ensure they remain related. In the future, we're considering building a split-history/traceability function that would illustrate how a particular Epic was divided over time, the relationship of the splits, when items were created, finished, deleted, etc. I've attached a mock-up of what the report might look like. But, in essence, this report provides a rollup view of a requirement's evolution and its current status.



Danube Technologies, Inc.
 520 SW 6th Ave, Suite 940
 Portland, OR 97204 USA
 1.888.5.danube
 T: +1.503.248.0800
 F: +1.503.248.0801

Copyright © 2006-2008 Danube Technologies, Inc.

