Agile Documentation with uScrum

Joaquim Baptista
Altitude Software
Alameda Fernão Lopes, 16 – 4 andar
1495-136 Algés, Portugal
+351-21-412-9800
px@acm.org

ABSTRACT

uScrum (uncertainty Scrum) is an agile process developed by a small team at Altitude Software to manage the process of writing user documentation. uScrum manages uncertainty and the unknown, allowing writers to quickly react to changing conditions.

uScrum uses orders of ignorance to understand the difficulty of tasks, allowing the team to effectively prioritize regular work together with difficult creative work.

uScrum overbooks writers on iterative cycles called sprints, then lets the writers micro-manage their tasks to overcome obstacles. After each sprint the team decides what to publish and whether to proceed with unfinished work.

Categories and Subject Descriptors

K.6.1 [**Management of computing and Information Systems**]: Project and People Management – *Life cycle, Management techniques.* H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Training, help, and documentation.*

General Terms

Documentation, Management, Measurement, Theory.

Keywords

Agile documentation, Agile management, Orders of ignorance, Complexity measures, uScrum, Writing, Wiki, Risk, Uncertainty.

1. THE ART OF ESTIMATING TASKS

Fully understanding an issue in order to get an accurate estimate may take as long as half the total effort, by which time the team manager might as well finish the work. In other words, getting an accurate estimate for a task is just too expensive to be practical. Classifying tasks by their order of ignorance[1] allows the team manager to apply different estimating strategies to tasks of each order. The team names orders of ignorance as *0oi* to *3oi*.

0oi Writers know everything needed to complete the task.

1oi Writers know what they need to write, but the information must still be gathered from developers.

2oi Writers know the problem to solve, but the team has never solved a similar problem.

3oi Writers try to solve a hard problem but are not sure how to proceed.

For small tasks expected to last days or at most a few weeks, the team manager just guesses based on past experiences. However, *2oi* tasks guessed to take more than a week often surprise the writers assigned to the task.

For large tasks at *0oi* or *1oi*, the team manager finds some item to count (for example, screens, parameters, errors, or even pages), and estimates based on a guessed average effort to complete each item.

For harder tasks, estimates are often pointless, either because the objective is poorly defined or because the steps to reach the objective are not known. However, it may be possible to estimate the effort to complete some initial work.

2. AGILE DOCUMENTATION

uScrum prioritizes tasks into monthly iterations called sprints, based on their importance, urgency, and timeliness. To juggle longer and harder tasks with small important tasks, the team must develop a shared long-term vision of major work as well as a feeling for the importance and urgency of upcoming short-term work.

Sprints are based on the idea of overbooking writers with more work than they should reasonably be expected to complete, and then letting writers micro-manage the exact work that gets done. In a month with 22 working days, for example, a writer may have work estimated to take 18 to 30 days.

The monthly iterations allow the team to react quickly to the changing needs of internal and external customers. Requiring new "urgent" work to be prioritized at the end of the month has kept most "urgent but not important" work at bay, preventing interruptions caused by work that turns out to be useless.

2.1 Task Selection for Sprints

The team chooses what to do based on the skills required by each task, how useful the output of the task will be for customers, and the timeliness of the task.

Writers must judge the importance of each task to decide what gets done immediately and what can be safely delayed. For example, some maintenance tasks are of general use while others matter to a single customer.

Some tasks have an ideal timing, normally during or after a feature is tested. Documenting those features near their best timing optimizes the time of writers, testers, and developers.

Hard tasks may require that the team delays short-term maintenance work in some sprints.

Tasks may linger in the backlog for years without ever being started. These are nice ideas that are too hard to put into practice, small tasks that are just not important enough, or work that cannot be done for lack of resources.

2.2 Writers in Sprints

Since estimates vary with the writer assigned to the task, writers renegotiate estimates at the beginning of the sprint. The understanding is that the writer should not spend significantly longer at a task than initially estimated without calling the manager's attention to that fact.

The writer that takes a task will often be the first person to try to understand the required effort in detail, which means that the initial estimate is often inaccurate.

Writers are free to help each other during sprints, as long as the team metrics remains accurate and the total effort does not exceed the initial estimate.

2.3 Handling Unfinished Tasks

At the end of the sprint, each task may be finished, not finished, or not started. In the next sprint, the team may decide to continue or stop the unfinished work.

The team may decide to stop the work abruptly if progress is not as expected, or if there is no progress at all. This happens if writers try to document new features too soon, when the interface or the behavior are not stable.

The team may also decide to finish some of the work done and stop further work. This captures some benefit from the effort done, but allows the writer to start working at other tasks. In effect the writer performed only part of a larger task.

Unfinished tasks at *0oi* and *1oi* are typically waiting for input from a developer, either basic information or the results of a document review. Also, the writer may have started the task just before the end of the sprint, or the task may have turned out to be larger or harder than expected.

Tasks at *2oi* often reach the end of the month unfinished, since they are both harder to do and harder to estimate. Typically, there is some partially completed work to show for the effort.

2.4 Meetings and Publishing

Sprints are aligned on week boundaries and typically take 4 weeks (a lunar month), although bank holidays and company events have forced different durations. Table 1 shows the schedule of the meetings and publishing events.

Table 1. Change-of-sprint schedule in 2008

Monday	Thursday	Friday	Monday
What to publish	Publish	Sprint retrospective	Sprint kick-off
	Sprint review		

The sprint starts with a two hour *Sprint kick-off meeting* that assigns tasks to writers. During the sprint, the team holds 15-minute *Stand-up meetings* two or three times a week, replacing the weekly meetings used previously. Stand-up meetings are followed by extra *ad-hoc meetings* as needed to address identified issues.

The last week of the sprint focuses on publishing and process. The week starts with a *What to publish meeting* where writers negotiate what work to publish, especially unfinished work. Tasks can be finished and integrated into the documentation set until Thursday morning, when the documentation set is republished.

On Thursday afternoon, writers present significant work to each other in the *Sprint review meeting*. These presentations foster a sense of accomplishment and spread knowledge through the team, which is especially important for new writers. Each presentation takes at most one hour.

On Friday, the *Sprint retrospective meeting* reviews the metrics of the sprint, determines what went well and what went wrong, and proposes improvements. The next sprint starts on the following week.

3. REFERENCES

[1] P. G. Armour. The laws of software process. Communications of the ACM, 44(1):15–17, January 2001.

[2] K. Schwaber. Agile Project Management with Scrum. Microsoft Press, Redmond, Washington, 2004.

Table 2. Assumptions of Scrum[2] and uScrum

	Scrum assumptions	uScrum assumptions
Product backlog	The top of the product backlog is fully prioritized and estimated.	*2oi* tasks cannot be reasonably estimated, although the team can estimate the effort that it is willing to invest at solving a task in each sprint.
Sprint goal	Team demonstrates an objective to the product owner in the sprint review, based on a realistic sprint backlog.	Writers show results for the effort invested. Overbooked writers select what work gets done during the sprint.
Team	Work must be coordinated among team members.	Work is mostly independent of other writers.
Sprint tasks	The sprint backlog can be broken down into 4 to 16 hour tasks in the first day of the sprint.	Finding out what to do or how to do it is a large part of the work that cannot be fully anticipated.
Sprint progress	A burndown chart can track the progress through the sprint.	There is nothing interesting to count down during a sprint.
Obstacles	The team can complete its tasks because it has the necessary skills and resources from the beginning. The Scrum Master can quickly remove any obstacle.	Some obstacles simply cannot be removed or anticipated by the team, such as a developer going on a two week vacation immediately after finishing a project.