Chapter 14: **Iteration Planning**

“It is a capital mistake to theorize before one has data.”

– Sherlock Holmes, *Scandal in Bohemia*

**Release Plan**: High level view of what is to be built

**Iteration Plan**: More focused view

Selecting user stories to be implemented

**Iteration Planning Meeting**

Product owner, analysts, programmers, testers, DB engineers, User IxDesigners, etc.

Anyone involved in the design and development

Stories defined by a set of tasks (Table 14.1)
“… using note cards during iteration planning”

Why would you not use the “technology” … simply enter the tasks into a spreadsheet? Better question… why would you?

Iteration Planning is a collaborative exercise

All involved… each sees what everybody else sees.

Arrange the cards (one for each story and one or more for the tasks required to design and implement what the story requires)… on the wall or on a table for all to see.
User stories of the Release Plan are decomposed into tasks.

The tasks are then estimated in hours.

Note. Collaboration is facilitated by use of whiteboard... visually recording the tasks – developed collaboratively.

Visual, so all can see, debate and come to an agreement.

### Table 14.1 Iteration Plan as a Simple Spreadsheet

<table>
<thead>
<tr>
<th>Row</th>
<th>Task</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>As a coach, I can assign swimmers to events for a meet.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Determine rules about who can swim in which events</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Specify acceptance tests to show how this should work</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Design user interface</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Code user interface</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>Add tables and stored procedures to database</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Automate tests</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>As a swimmer, I can update my demographics</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Specify acceptance tests</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>Change view-only demographics page to allow edits</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
### Example (Fig. 14-1)

**Iteration Planning: Note cards on a table or whiteboard display**

<table>
<thead>
<tr>
<th>Story</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a coach, I can assign swimmers to events for a meet.</td>
<td></td>
</tr>
</tbody>
</table>
Determine rules about who can swim n which events.  
Specify acceptance tests to show how this should work |
| Design user interface | Code user interface. |
| As a swimmer, I can update my demographics |  
Specify acceptance tests  
Change view-only demographics page to allow edits |
Tasks are *not* allocated during iteration planning

- Promotes “all in this together” attitude
  
  No individual ownership going into an iteration
  
  Members can (should) help one another when needed

- Individuals sign-up for tasks once the iteration begins

- Signing up for only one or two tasks…

- Work on a new task begins only after selected tasks are completed
Table 14.2 Primary Differences between a Release and an Iteration

<table>
<thead>
<tr>
<th></th>
<th>Release Plan</th>
<th>Iteration Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning horizon</td>
<td>3 to 6 months</td>
<td>2 - 4 weeks</td>
</tr>
<tr>
<td>Items in plan</td>
<td>User stories</td>
<td>Tasks</td>
</tr>
<tr>
<td>Estimated in</td>
<td>Story points</td>
<td>Ideal hours</td>
</tr>
</tbody>
</table>

Stage 1

Release Plan

Vague concerning specific order in which user stories are to worked-on

Stage 2

Iteration planning

Refines understanding of user stories (more detailed)
Team knows more than was known during the Release Planning
Team discusses product design and software design
Velocity-Driven Iteration Planning

(the 1\textsuperscript{st} of the author’s two recommended approaches)

Fig. 14.2
Sequence of steps in velocity-driven iteration planning
(Story Points per Iteration)
Adjust Priorities

Dealing with changes

Iteration (Sprint) Review Mtg. (usually 30 to 60 minutes)
- Held after the iteration is done
- Demo of functionality & capabilities
- Potential Product Owner (representing customers & users) may have changes that preempt previously determined high-priority items

Prioritization Mtg.
- Two days before the end of the iteration
- Discuss unfinished work
- Chaired by the Product Owner

Next Iteration Planning Mtg. (4 hours)
- Prioritize for the next iteration … factoring in changes agreed to in the Iteration (Sprint) Review Mtg. and Prioritization Mtg.)
Determine Target Velocity

• *Velocity* in next iteration should equal the *Velocity* in the most recent iteration

• Alternative: Reset the *Velocity* to the moving average calculated for previously completed iterations

**Identify an Iteration Goal**

• Describes what team would like to achieve (doesn’t have to be specific)

*SwimStats* Iteration Goals

- Make progress on reports
- Finish all event time reports
- Get security working
Identify an Iteration Goal

What the team hopes to accomplish during the iteration.

Example

Iteration goal for *SwimStats*

(what will be worked on in the iteration):

- Make progress on reports
- Finish all event time reports
- Get security working
Select User Stories

- Product owner & team select stories to meet goals
- Again, SwimStats

Goal: “All demographic features are finished”

User Stories

  As a swimmer, I can update my demographics
  As a coach, I can enter demographic data on each of my swimmers
  As a coach, I can import a file of all demographic data
  As a coach, I can export a file of all demographic data

Prioritize the order of work

  (assigning tasks occurs once the Iteration begins)
Split User Stories into *Tasks*

User Story:

“As a coach, I can assign swimmers to events for an upcoming meet”

List of tasks:

- Determine rules that affect who can be assigned to which events
- Write acceptance test cases that show how this should work
- Design the user interface
- Get user interface feedback from coaches
- Code the user interface
- Code the middle tier
- Add new tables to database
- Automate the acceptance tests

Analysis, design, user IxD, or other tasks that are necessary should be included and estimated (how much time each will take)

“All tasks necessary to go from a user story to a functioning, finished product should be identified.”
Include only work that adds value to the Project

• Exclude “email time”
• Exclude peripheral tasks … the overhead of work

Be Specific Until It’s a Habit

Identify and estimate unit testing tasks

Example: Programmer estimates that coding a new feature at 8 hours and writing unit tests at 5 hours

Once the programmer assumes that “coding” includes unit testing, only one estimate would be needed
Meetings Count (A lot)

• Identify, estimate and include tasks for meetings related to the project
  – Time for preparing
  – Include the time for all attendees

Mtg. sample estimates:
  Analyst spends two hour preparing
  One hour meeting, 7 attend
  Meeting task estimate is 9 hours

Bugs

• Programmer coding estimate includes time to fix bugs
• For Bugs found after the iteration is completed, fixing is prioritized and added to a subsequent iteration
Handling Dependencies

- One Story depends on the previous implementation of another story
- Not a problem if developed in their natural order

**SwimStats** example

Story 1: Add new swimmers to the system

Story 2: Let user view an individual swimmer’s fastest times

   Story 2 is implemented before Story 1

   Iteration planning would need to add “design data tables for info about individual swimmers” to the iteration

Will project take longer?

   Maybe not. Shifted some tasks from Story 1 to Story 2

   Even if it does, the reason for the reordering was approved by the Product Owner
Work that is difficult to split

**Example:** Small change to legacy feature

No one was confident in knowing the possible impact Sections where the code was changed would be affected
But, what of “side effects” to other areas of code? Two tasks were identified:

1. Determine what’s affected – 2 hours
2. Make the changes – 10 hours

The 1st task was a “spike”… to gain knowledge on how to approach the 2nd task
Estimate Tasks

Expressed in “Ideal Time”

– “Ideal Time” of 6 hours but took 8 hours (the “Elapsed Time”)

Reasons why task estimates should be done by the group:

1. Tasks are not assigned to specific team members during Iteration Planning

2. The task will be assigned to one person, but others can contribute

   The “one” person might be the right person, but others may have useful questions to ask about the scope of work, experience on previous tasks that were similar, knowledge that the person assigned had forgotten

3. “… hearing how long something is expected to take … helps the team identify and clarify misunderstandings about the Story or task

4. When the estimate was made collaboratively and person doing the work takes longer, the team takes the “blame” and not the person.
Some design is OK

During Iteration Planning … identifying tasks and assigning estimates to each requires understanding the design work needed.

Product owner, analysts and IxDDesigners *may* discuss design, creating the list of tasks and estimates

For example:

– How much of a feature should be implemented
– How it will appear to users
– Options of how to implement what is needed

The Right Size for a Task

Each developer is able to finish an *average* of 1 task per day

Allows work to flow smoothly through the iteration
Add stories to the iteration 1-by-1 until the team can commit to completing no more.
Ask for Team Commitment

“Commitment… key factor contributing to team success.”

• Not a “hardwired” velocity requirement
• Committed to delivering the functionality described by a user Story… the initial set of task and newly identified tasks discovered during the iteration

Summing the Estimates

• Sum the estimated time for the tasks – Determine if it is a reasonable amount of work
• Range estimates are appropriate, given the uncertainty
• May require spreading the work to other team members

(Major point: Continually assessing… not a one time commitment to a “finish” time and then wait and see…)
Maintenance and the Commitment

• Teams may also be responsible for support and/or maintenance of other systems, for example:
  – A prior version of the product
  – An unrelated product
• These are unpredictable commitments

The commitment to work for the iteration is based on how much room is left (~90%)

Support, maintenance, and other commitments (~ 10%)

• May be predictable… using past averages
• …or not
Author’s Recommendation

• **Commitment Driven** (over Velocity Driver) Iteration planning

• **Why not Velocity Driven?**
  1. **Velocity (Story Points per Iteration)** is a “coarse grained estimate”
     - Not accurate enough for planning short iterations
     - Good for estimating amount of work a team will complete per iteration
  2. Team would need to complete 20 to 30 stories per iteration for errors in the story point estimates to average out.
     - (typical where 3 to 12 stories are included in each iteration)

Note. While the averaging-out problem exists between iterations … velocity “works well for long-range release planning”.
Relating Task Estimates of Time to Story Points

Example

At the end of an given iteration, the team calculates the time per story point as 12 hours.
They assumed that one Story Point took 12 hours of work.
The time per Story Point is a Random Variable… interpreted below as a Normal Distribution.
Example: 50% require less than 12 hours, 50% require more.
Fig. 14.6
Distribution of hours to complete for 1, 2 and 3 point stories. Each distribution is represents hours as a random variables (1-point, 2-point and 3-point)
Days of the Week

Initially, iterations were started on Monday, ended on Friday

Logical? NO

If something went wrong on Friday… fixing could be done on the weekend

Team decided to run 2-week iterations, start on Friday and end on a Thursday

Fridays were time to do the Iteration Review and next Iteration Planning

Avoided “dread” of Monday meetings… Monday became the review and planning day

Friday also provided time to wrap-up last-minute work… and avoiding weekend work : )