Quick and Dirty Skinny on Agile Software Development

Agile Principles

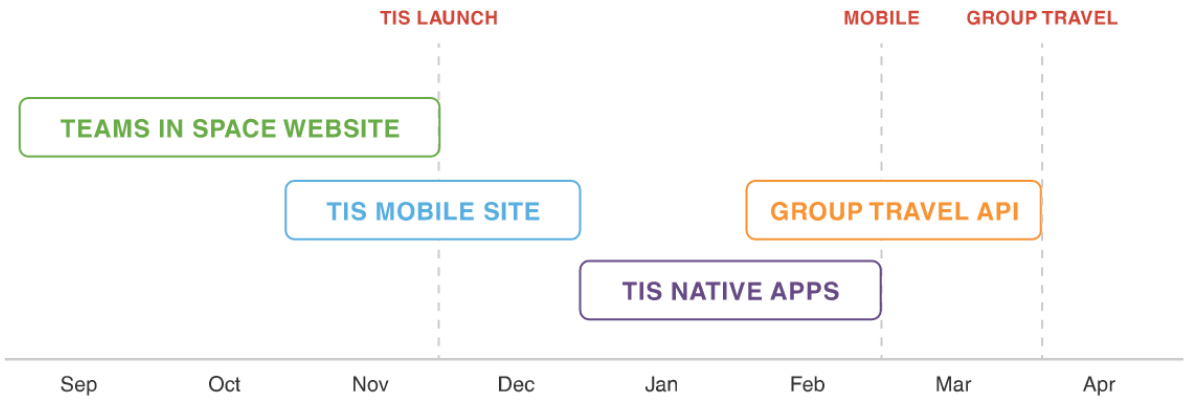
- Agile **versus** Traditional Contract Implementation
- Focuses on individuals **versus** processes and tools
- Working software/code **versus** documentation
- Customer collaboration **versus** contract negotiation
- Ability to respond quickly to change **versus** sticking to a project plan

What does this mean for contracts for Agile managed projects? It means that ideally, contracts for these projects are best structured as smaller (modular) pieces of a whole to allow flexibility for any needed changes and ensure delivery of working product/software/code.

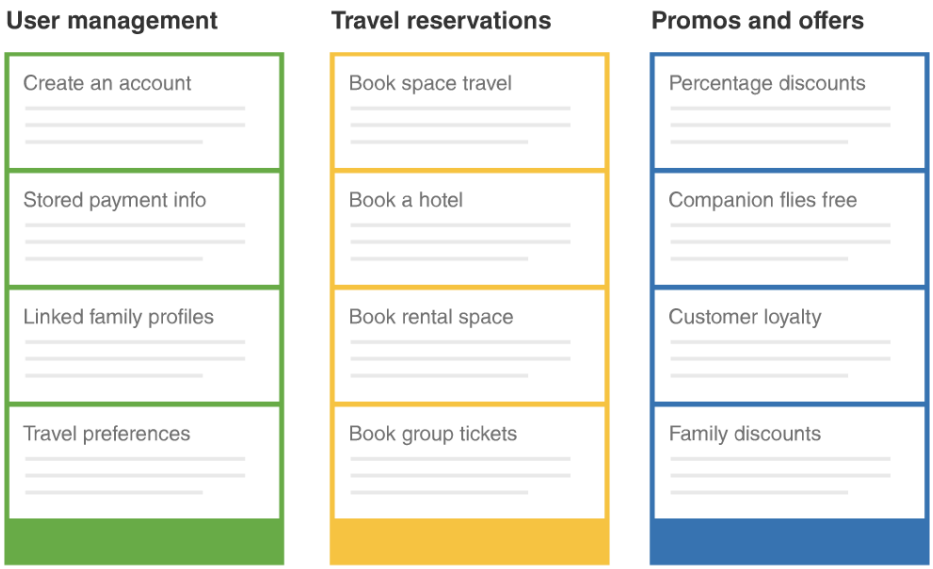
How are Agile Projects Managed?

You need a roadmap! Roadmap n. *[rōd-map]* A plan of action for how a product or solution evolves over time. The key to a well-managed agile project is a solid roadmap.

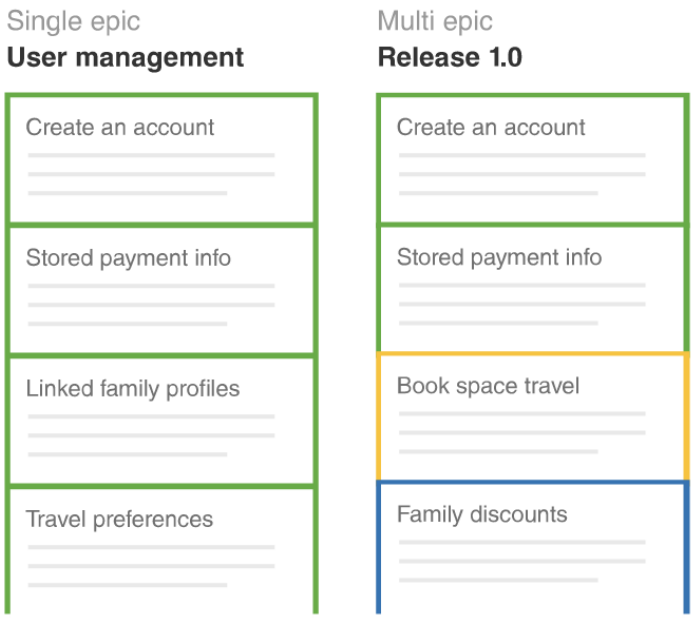
Roadmaps are built by product owners, and include large areas of product functionality, and when features will be available. To build a roadmap, product owners take into account market trajectories, value propositions, and engineering constraints. Once these factors are reasonably well-understood, they are expressed in a roadmap as initiatives and timelines. Below is a very simple roadmap for a product team. The initiatives are in blue and timelines are indicated by the milestone-markers in red.



Each part of the roadmap will be broken down into smaller parts called Epics, which will have User Stories *(both explained later).* Since the Teams in Space website is the first initiative in the roadmap, we'll want to break down that initiative into epics: e.g. Travel Stories, Travel Reservations, Promos… and then into User Stories for each of those epics: e.g. for Travel Reservations: Book Travel, Book a hotel...



The product owner then organizes each of the user stories into a single list for the development team. The product owner may choose to deliver a complete epic first (left). Or, it may be more important to the program to test booking a discounted flight which requires stories from several epics (right). See both examples below.



What may influence a product owner's prioritization?

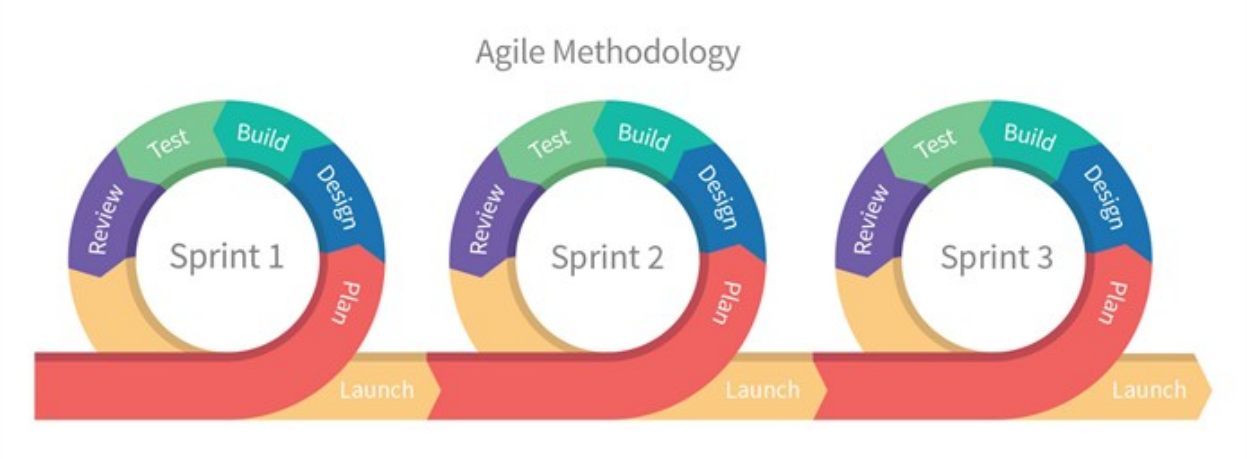
- Customer priority
- Urgency of getting feedback
- Relative implementation difficulty

- Symbiotic relationships between work items (e.g. B is easier if we do A first)

While the product owner is tasked with prioritizing the backlog, it's not done in a vacuum. Effective product owners seek input and feedback from customers, designers, and the development team to optimize everyone's workload and the product delivery.

Managing Development

Epics=Iterations=Sprints (in Scrum). Each of these have user stories.



What is a user story?

User stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. They typically follow a simple template:

As a <type of user>, I want <some goal> so that <some reason>.

User stories are often written on index cards or sticky notes, stored in a shoe box, and arranged on walls or tables to facilitate planning and discussion. As such, they strongly shift the focus from writing about features to discussing them. In fact, these discussions are more important than whatever text is written.

User stories create the product "backlog," which is a prioritized list of work. The most important items are shown at the top of the product backlog so the team knows what to deliver first. The key to an agile process here is that the development team **pulls** work from the product backlog as there is capacity for

it, either continually (e.g. kanban) or by iteration (e.g scrum) instead of being assigned it based on a set schedule.

What Size Should A User Story Be?

A story should be small enough to be coded and tested within an iteration—ideally just a few days. When a story is too large, it is called an epic. Backlog items tend to start as epics when they are lower priority. User story size guides:

Too broad:

- A team member can view iteration status.

Too detailed

- A team member can view a table of stories with rank, name, size, package, owner, and status.
- A team member can click a red button to expand the table to include detail, which lists all the tasks, with rank, name, estimate, owner, status.

Just right

- A team member can view the iteration's stories and their status with main fields so that they have immediate information presented.
- A team member can view the current burndown chart on the status page, and can click it for a larger view, so they can easily view the data.
- A team member can view or hide the tasks under the stories because they need a simplified roll up view and ability to see the detail when necessary.
- A team member can edit a task from the iteration status page so they do not have to go to another page to edit.

When Do I Add Detail to a User Story?

Acceptance criteria provide the Definition of Done for the story. As details about the story evolve, capture the critical ones as acceptance criteria. The product owner should list as many acceptance criteria as possible in order to clarify the intent of the story. Regardless of how detailed the acceptance criteria are, the team should have a conversation about them and adjust the acceptance criteria to capture the results of the discussion. Once an iteration has begun, testers can formalize acceptance criteria into acceptance tests.

Who Uses User Stories?

- Creation: The customer, customer proxy, product owner, and anyone else who identifies a need for the product can contribute user stories.

- Ownership and maintenance: The product owner owns the user stories and is responsible for writing, gathering, maintaining, and prioritizing.

- Usage: Developers, testers, technical writers use user stories to be able to know what to implement and when they are done. Product owners track overall progress based on the status of the user stories. Management tends to track user stories rolled up to epics or features.

What Are The Top Mistakes That People Make?

- Too formal or too much detail. Product owners with good intentions often try to write extremely detailed user stories. If a team sees a story at iteration planning that looks like an IEEE requirements document, they often assume that all the details are there and will skip the detailed conversation.
- Technical tasks masquerading as stories. Much of the power of Agile comes from having a working increment of software at the end of each iteration. If your stories are really just technical tasks, you often do not end up with working software at the end of each iteration, and you lose flexibility in prioritization.
- Skipping the conversation. Stories are intentionally vague before iteration planning. If you skip the acceptance criteria conversation, you risk moving in the wrong direction, missing edge cases or overlooking customer needs.

DITAP Team Example