SCRUM Product Owner Schulung

Agenda

Was ist eine User Story?

Fallstricke und Vorteile

Nichtfunktionale Anforderungen und User Stories

SRS und User Stories

User Stories und Use Cases – kompatibel?

Akzeptanz-Kriterien

Akzeptanzkriterien & DoD

The User Story

User Stories sind Teil des agilen Ansatzes, der dazu beiträgt, den Schwerpunkt vom Schreiben über Anforderungen auf das Sprechen darüber zu verlagern.

Die User Stories bestehen aus ein oder zwei Sätzen und einer Reihe von Gesprächen über die gewünschte Funktionalität.

User Stories sind kurze, einfache Beschreibungen eines Bedarfs, die aus der Perspektive der Person erzählt werden, die die neue Funktion wünscht, normalerweise ein Benutzer oder Kunde des Systems.

Sie folgen in der Regel einer einfachen Vorlage:

Als <Typ von Benutzer> möchte ich <ein Ziel>, damit <ein Grund>.

The User Story

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

User Stories werden oft auf Karteikarten oder Haftnotizen geschrieben und an Wänden oder Tischen angeordnet, um die Planung und Diskussion zu erleichtern.

Dadurch verlagert sich der Schwerpunkt stark vom Schreiben über Funktionen auf die Diskussion darüber.

In der Tat sind diese Diskussionen wichtiger als der geschriebene Text.

The User Story

User Stories sollen anhand der 3C guidelines (Ron Jeffries) angewandt werden:

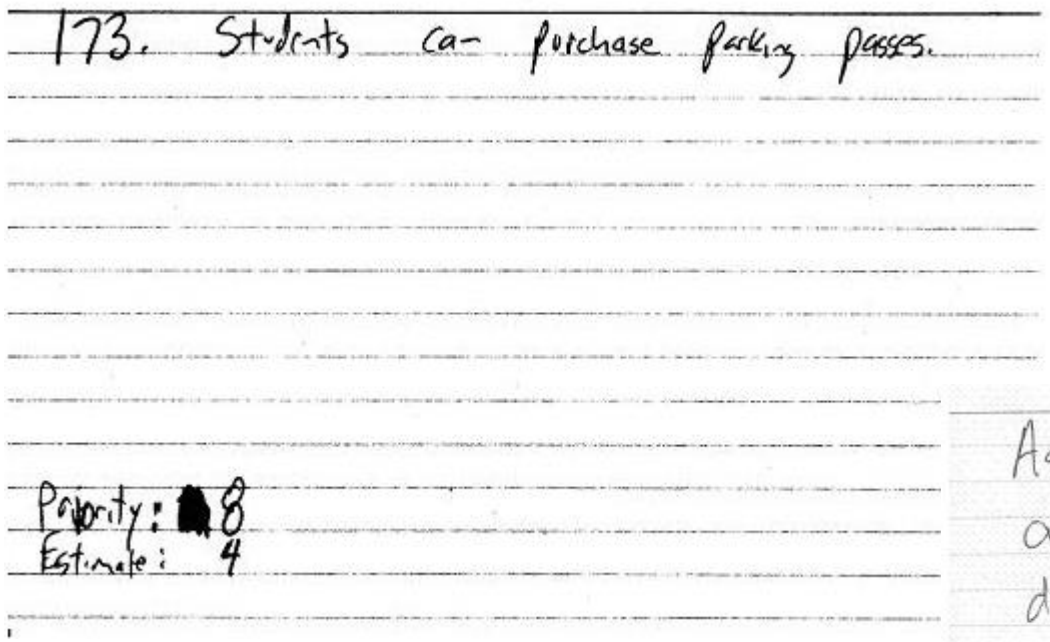
Card: index cards oder sticky notes, in kurzer Form

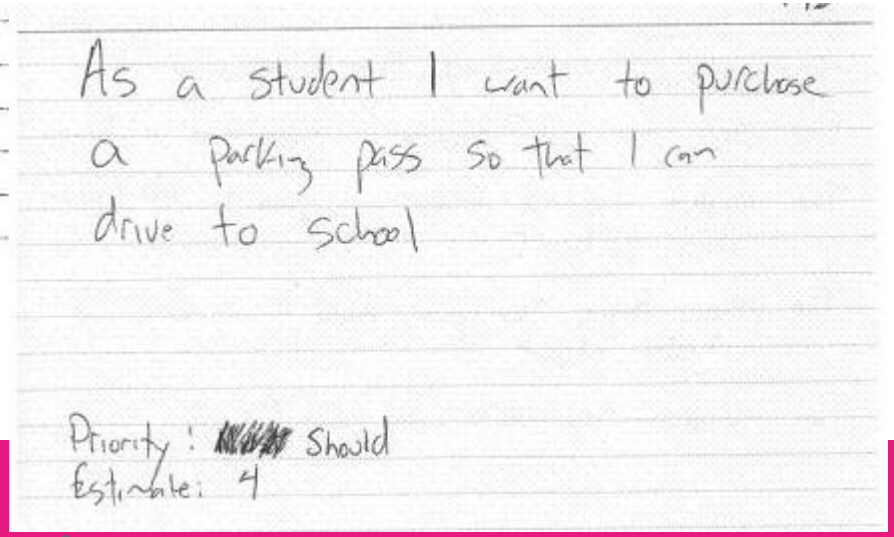
Conversation: Fokus von Schreiben zur Diskussion

Confirmation: Acceptance tests dienen zur Bestätigung das die Anforderung richtig umgesetzt wurde

User Story - example

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





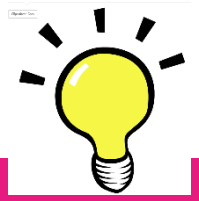
User Stories and Epics

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

Einer der Vorteile von agilen User Stories ist, dass sie auf verschiedenen Detailebenen geschrieben werden können. (siehe auch Spezifikationsstufen von Rupp et al)

Wir können eine User Story schreiben, um große Mengen an Anforderungen abzudecken. Diese großen User Stories werden im Allgemeinen als Epic bezeichnet. Hier ist ein Beispiel für eine episches agile User Story für ein Desktop-Backup-Produkt:

As a user I can backup my entire hard disk



(ToDo: find the specification level)

User Stories and Epics

Da ein Epic für ein agiles Team in der Regel zu umfangreich ist, um es in einer Iteration abzuschließen oder zu schätzen, wird es in mehrere kleinere User Stories aufgeteilt, bevor es bearbeitet wird.

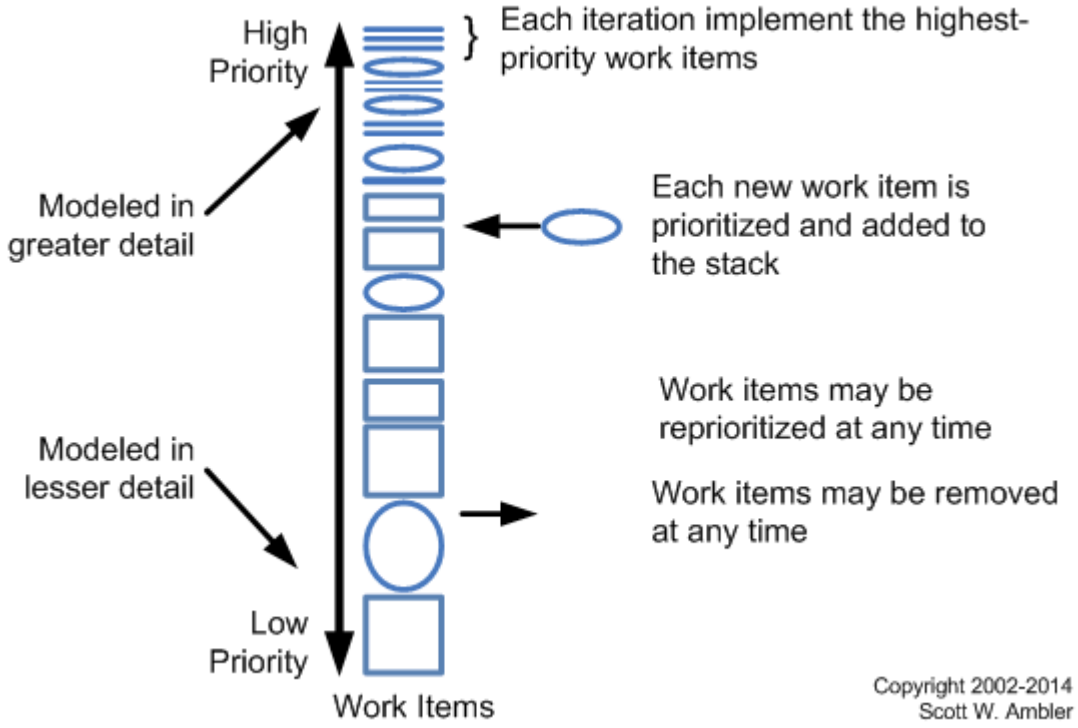
Das obige Epic könnte in dutzende aufgeteilt werden, einschließlich dieser beiden:

Als Power-User kann ich die zu sichernden Dateien oder Ordner anhand der Dateigröße, des Erstellungsdatums und des Änderungsdatums angeben.

Als Benutzer kann ich Ordner angeben, die nicht gesichert werden sollen, damit mein Sicherungslaufwerk nicht mit Dingen gefüllt wird, die ich nicht brauche.

User Stories and Epics

Disciplined agile change management process.



User Stories and Epics

As a power user, I can specify files or folders to backup based on file size, date created and date modified.

As a user, I can indicate folders not to backup so that my backup drive isn't filled up with things I don't need saved.

Define the specification level

Why are Epics important?

Why not defining User Stories instead of Epics?

User Stories and Themes

Ein Theme ist eine Sammlung von zusammenhängenden User Stories.

Bei einem Universitätsregistrierungssystem könnte es zum Beispiel Themen rund um Studenten, Kursverwaltung, Zeugniserstellung, Notenverwaltung und finanzielle Abwicklung geben.

Themes werden oft verwendet, um Stories in Releases zu gliedern oder sie so zu organisieren, dass verschiedene Unterteams an ihnen arbeiten können.

Es geht also darum, die User Stories zu organisieren und zu strukturieren.

INVEST User Stories

A good user story should be:

- "I" ndependent (of all others)
- "N" egotiable (not a specific contract for features)
- "V" aluable (or vertical)
- "E" stimable (to a good approximation)
- "S" mall (so as to fit within an iteration)
- "T" estable (in principle, even if there isn't a test for it yet)

Developed by Bill Wake

INVEST Is not SMART !

INVEST is not SMART

The INVEST checklist targets user stories.

The SMART checklist targets tasks (which decompose user stories from a technical point of view).

- “S" pecific
- “M“ easureable
- “A" chievable
- “R" elevant
- “T" ime boxed

User Stories - pitfalls

Ein klassischer Fehler besteht darin, in Teams, in denen der agile Ansatz auf das Entwicklungsteam beschränkt ist oder nach einer nicht-agilen Anforderungsphase übernommen wurde, von einem Anforderungsdokument im narrativen Format auszugehen und User Stories direkt aus dessen Struktur abzuleiten: zum Beispiel eine Story für jeden Abschnitt des Dokuments

Wie das 3C-Modell betont, ist eine User Story kein Dokument und sollte nicht wie ein Dokument behandelt werden. Es geht um Kommunikation

User Stories - pitfalls

der Detaillierungsgrad einer User Story ist nicht konstant, sondern entwickelt sich im Laufe der Zeit in Abhängigkeit vom „Planungshorizont“;

so sollte beispielsweise eine User Story, die für die nächste Iteration vorgesehen ist, besser verstanden werden als eine, die erst in der nächsten Version implementiert wird

eine User Story ist kein Use Case; aufgrund der unterschiedlichen Ausgangssituation sind sie auch nicht wirklich kompatibel - sind lassen keine Eins-zu-Eins-Zuordnung zu

eine User Story entspricht im Allgemeinen nicht einer technischen Komponente oder einer Komponente der Benutzeroberfläche: auch wenn es manchmal eine nützliche Abkürzung ist, z.B. von „der Suchdialog-Story“ zu sprechen, sind Bildschirme, Dialogfelder und Schaltflächen keine User Stories

User Stories and Details

Details können zu User Stories auf zwei Arten hinzugefügt werden:

- ✓ Durch Aufteilung einer User Story in mehrere, kleinere User Stories

- ✓ Durch das Hinzufügen von Akzeptanzkriterien

User Stories and Details

Die Akzeptanzkriterien (oft auch als Zufriedenheitsbedingungen bezeichnet) sind einfach ein übergeordneter Akzeptanztest, der nach Fertigstellung der agilen User Story erfüllt sein wird.

Beispiel:

As a vice president of marketing, I want to select a holiday season to be used when reviewing the performance of past advertising campaigns so that I can identify profitable ones.

User Stories and Details

Dieses Beispiel für eine User Story könnte durch Hinzufügen der folgenden Acceptance Criteria noch detaillierter gestaltet werden:

Make sure it works with major retail holidays: Christmas, Easter, President’s Day, Mother’s Day, Father’s Day, Labor Day, New Year’s Day.

Support holidays that span two calendar years (none span three).

Holiday seasons can be set from one holiday to the next (such as Thanksgiving to Christmas).

Holiday seasons can be set to be a number of days prior to the holiday.

Ersetzen User Stories ein SRS?

Agile Projekte verwenden ein Product Backlog die in einem Produkt oder einer Dienstleistung entwickelt werden sollen. Obwohl die Elemente des Product Backlogs beliebig sein können, haben sich User Stories als eine der besten und beliebtesten Arten von Product Backlog-Elementen herausgestellt.

Während ein Product Backlog als Ersatz für das Anforderungsdokument eines traditionellen Projekts angesehen werden kann, ist es wichtig, daran zu denken, dass der schriftliche Teil einer agilen User Story („Als Benutzer möchte ich ...“) unvollständig ist, bis die Diskussionen über diese Story stattfinden.

US and Quality of services

QoS passen ohne Probleme in die

Als .... Ich möchte ... Vorlage

Als Kunde möchte ich in der Lage sein, Ihr Produkt auf allen Windows-Versionen ab Windows 95 einzusetzen.

Als CTO möchte ich, dass das System unsere bestehende Auftragsdatenbank nutzt, anstatt eine neue zu erstellen, damit wir nicht noch eine weitere Datenbank zu pflegen haben.

Acceptance Criteria

Acceptance Criteria are the conditions that a software product must satisfy to be accepted by a user, customer, or in the case of system level functionality, the consuming system.

Acceptance Criteria are a set of statements, each with a clear pass/fail result, that specify both functional and non-functional requirements, and are applicable at the Epic and Story Level.

Acceptance criteria constitute our “Definition of Done”

Acceptance Criteria - Purpose

Acceptance Criteria's:

Define the boundaries for a user story/feature

Help the product owner answer what they need in order for this feature to provide value (typically these are the minimum functional requirements)

Help the team gain a shared understanding of the story/feature

Help developers and testers to derive tests

Help developers know when to stop adding more functionality to a story

Acceptance Criteria – good to know

Write the criteria before starting development!

If we write and review the criteria before implementation begins, we're more likely to capture the **customer intent** rather than the **development reality**.

Acceptance Criteria – good to know

Die Kriterien sollten eine Absicht, aber keine Lösung vorgeben

e.g., “User can approve or reject an invoice” rather than “User can click a checkbox to approve an invoice”.

The criteria should be independent of the implementation, and discuss WHAT to expect, and not HOW to implement the functionality.

Acceptance Criteria - Formats

… are often formulated like this:

Given…

When…

Then…

Given some precondition When I do some action Then I expect some result

When writing acceptance criteria in this format, it provides a consistent structure.

Additionally, it helps testers determine when to begin and end testing for that specific work item.

Acceptance Criteria - Formats

Bei User Stories auf Systemebene ist es schwierig, Kriterien nach dem Format „wenn, dann“ zu erstellen.

In solchen Fällen können Sie auch Verifizierungs-Checklisten verwenden.

Ein weiterer Vorteil von Verifizierungs-Checklisten besteht darin, dass sie bei der Implementierung von Funktionen einfach einzeln als vollständig markiert werden können.

Acceptance Criteria

As a customer,
I want to withdraw cash from an ATM
So that I don't have to wait in line at the bank.

Acceptance Criterion 1:

Given that the account is creditworthy
And the card is valid
And the dispenser contains cash,
When the customer requests the cash
Then ensure the account is debited
And ensure cash is dispensed
And ensure the card is returned.

As a customer,
I want to withdraw cash from an ATM
So that I don't have to wait in line at the bank.

Acceptance Criteria

Acceptance Criterion 2:

Given that the account is overdrawn
And the card is valid,
When the customer requests the cash
Then ensure the rejection message is displayed
And ensure cash is not dispensed.

What about details such as e.g.:

The column heading is “Balance”

The rolling balance format is 99,999,999,999.9 D/CR

We should use a dropdown rather than checkboxes

Acceptance Criteria - Formats

These kind of details normally come up in the conversation about the story with the product owner. This would be at the sprint planning meeting or when the team starts coding this particular story.

Acceptance Criteria - Formats

Gherkin Format:

```
1: Feature: Some terse yet descriptive text of what is desired
2:   Textual description of the business value of this feature
3:   Business rules that govern the scope of the feature
4:   Any additional information that will make the feature easier to understand
5:
6:   Scenario: Some determinable business situation
7:     Given some precondition
8:       And some other precondition
9:     When some action by the actor
10:      And some other action
11:      And yet another action
12:    Then some testable outcome is achieved
13:      And something else we can check happens too
14:
15:  Scenario: A different situation
16:      ...
```

https://github.com/cucumber/cucumber/wiki/Gherkin

Acceptance Criteria & DoD

For a user story or feature to be “potentially shippable” it needs to meet the expectations of the Product Owner and be of the agreed quality.

The Product Owner’s expectations are phrased as acceptance test criteria. Acceptance test criteria are conditions of satisfaction specific for each individual user story.

The user story’s (internal) quality is defined in the “Done” statement. The “Done” statement is applicable to all user stories in the project

http://nomad8.com/acceptance-criteria-and-the-definition-of-done/

Acceptance Criteria & DoD

User Story:

“As a music lover I want to be able to pay for my album by VISA card”

Example Acceptance Criteria (specific for this story):

I can purchase an album by VISA card

I cannot pay with a VISA card that’s expired

I cannot pay with a VISA card with a wrong number

http://nomad8.com/acceptance-criteria-and-the-definition-of-done/

Acceptance Criteria & DoD

Example Definition of Done (DoD) (valid for ALL stories):

0 (known) bugs

Passed unit tests

Passed automated Cucumber acceptance tests

Passed cross browser testing

Passed UAT

Code peer reviewed (if not pair programmed)

Relevant documentation updated

http://nomad8.com/acceptance-criteria-and-the-definition-of-done/

Acceptance Criteria & DoD

The “Definition of Done”(DoD) is the team/project’s quality statement for a user story or feature and ensures that a feature is 100% developed and tested.

It is a statement that no more work is left to be done on this piece of functionality (90% done doesn’t count!) and that it works without errors.

http://nomad8.com/acceptance-criteria-and-the-definition-of-done/