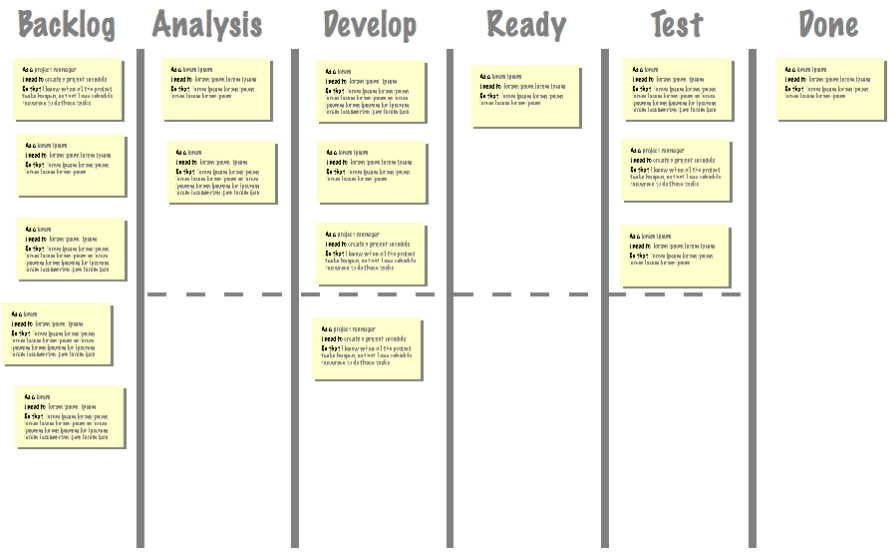
A Primer on

User Stories, Acceptance Criteria

&

Acceptance Test Driven Development (ATDD)

and a little bit of Specification by Example



The Problem with Specifications

The traditional way of building software is to write down everything that the customer wants in a document called a specification or requirements document.

It doesn't work very well.

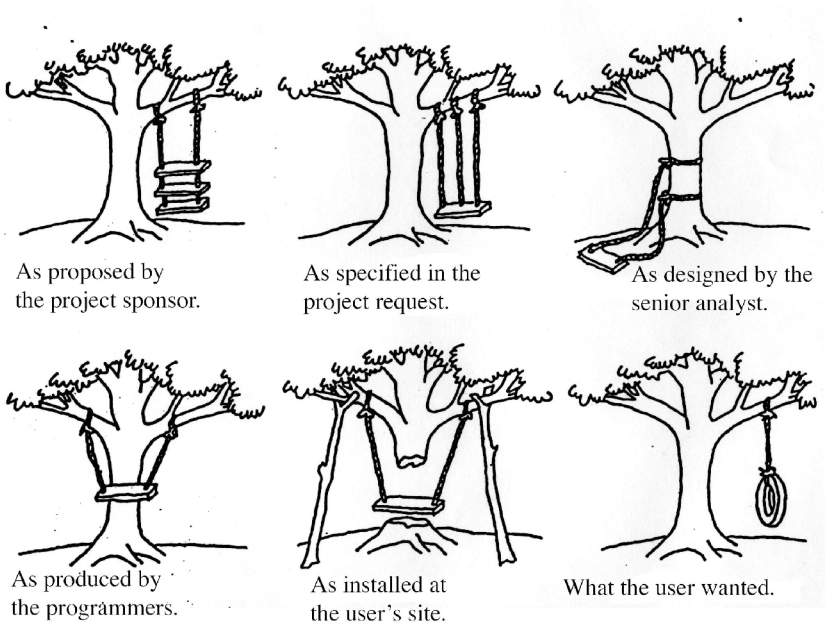
Here's something I wrote way back in 2005-06 :

> *In technical software projects, clients are generally unfamiliar with not only the concepts and terminology but also with the functionality that is being described to them.*
>
> *Typically, the experts go through a lengthy and detailed design process and deliver a custom built solution, tailor-made to fit the client's requirements.*
>
> *At this point the client looks at it and says: "That's not what I want!"*
>
> *And the developer/BA/designer blurts out: "But that's what you asked for!"*
>
> *There has been a fundamental breakdown in communications and on one knows why.*
>
> *The problem stems from the fact that the project team are experienced software practitioners and the client is not. They are used to interpreting specifications and the client is not. When a developer sees "a multi-function hierarchical picklist" in a spec they understand what it means. The client sees convincing techno-babble delivered by a competent technical expert and (at first) nods vigorously.*

In recent years, the rise of Agile has spurred an interest in what might be called 'lightweight' forms of requirements documentation. This is because there has been a recognition that pursuing ever more elaborate forms of documentation does not solve the requirements gap I illustrate above.

What is needed is more elaborate forms of communication and discussion.

The richer the better.



The classic "tyre swing cartoon" from the 1960's.
See http://www.businessballs.com/treeswing.htm for the history...

User Stories & Acceptance Criteria

A *user story* or *feature* describes functionality that delivers value to a user or a customer. It tells a story about how someone uses the product and what they will achieve with it.

It is not meant to encapsulate the whole requirement – it is meant to be an easily understandable placeholder for the conversation that takes place about the requirement.

Acceptance criteria or *scenarios* are rules which confirm the completeness and correctness of a user story. How will we know when a user story is complete in the system? It is done, when the system complies with the acceptance criteria.

As a story is discussed, the acceptance criteria develop and confirm the exact behaviour of the system under certain conditions. Acceptance criteria start out simple and develop into more formal and structured statements as the requirement is better understood.

Patterns

The easiest way to learn to write user stories and acceptance criteria is by using "patterns". By using these patterns repeatedly you form a common language within your team that you use to describe your requirements; which reduces ambiguity.

As a ***<type of user>***	As a customer service rep
I want ***<a feature>***	I want to create a new quote
So that ***<I get a benefit>***	So that we can sell a new policy to the customer
Given ***<a known state>***	… Given the customer exists
When ***<I perform an action>***	And I have the customer selected
Then ***<something happens>***	When I generate the quote
	Then the quote is attached to the customer record
	And the quote is emailed to the customer
	… Given the customer does not exist
	When I generate the quote
	Then the system prompts me to add the customer

Rules of Thumb

- Each user story should have about 3-5 acceptance criteria
- Each story should be independent and stateless – it should not rely on another story. If it does, you should consider splitting the story into sub-stories.
- Avoid implementation details and focus on outcomes – don't say how a thing is to be done, but what the outcome should be.
- If you can't write any acceptance criteria for your story then it's likely that it can't be implemented!

Standardising Your Language

In order to make the best use of user stories and acceptance criteria you need to standardise your language. If you refer to a "user" in one acceptance criteria and an "end-user" in another then it won't be clear whether you are referring to the same thing.

To this end, you should create a dictionary to be use with your acceptance criteria that specifies what each keyword means. Each keyword should have only a single meaning and the same keyword should always be used for the same artefact (user, field, screen etc).

Advanced

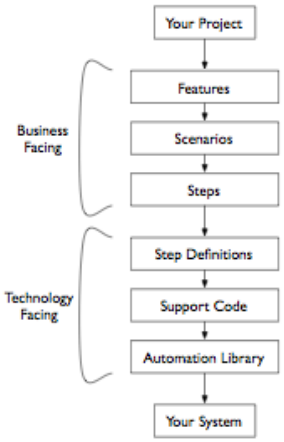
- And – you can extend each simple clause of an acceptance criteria by using "and" to specify another clause. For example "when I fill the kettle with water AND plug it into the electricity AND turn it on"
- Background – if you have a number of acceptance criteria which have common clauses you can use a "background" statement to make them easier to read. Typically the background statement contains the common "given" and "when" statements for all clauses, but the "then" (outcomes) will be unique.

Gherkin, Cucumber and other Automated Fruit

The pattern for criteria is from a language called "Gherkin".

Gherkin works with an interpreter called Cucumber and a range of other tools to provide an automated testing stack for software. By interpreting the acceptance criteria through Cucumber and then using another layer to bolt it onto your specific system you can automatically test all of your requirements.

You can choose to execute each test manually or automatically but by doing them automatically you can quickly home in on what has changed and what might be broken and use the manual effort to diagnose the problem.



IThe Cucumber Test Stack from [The Cucumber Book](#) by Matt Wynne and Aslak Hellesoy

From "The Cucumber Book" :

The idea of automated acceptance tests originates in eXtreme Programming1 (XP), specifically in the practice of Test-Driven Development2 (TDD).

Instead of a business stakeholder passing requirements to the development team without much opportunity for feedback, the developer and stakeholder collaborate to write automated tests that express the outcome that the stakeholder wants.

We call them acceptance tests because they express what the software needs to do in order for the stakeholder to find it acceptable. The test fails at the time of writing, because no code has been written yet, but it captures what the stakeholder cares about and gives everyone a clear signal as to what it will take to be done.

These tests are different from unit tests, which are aimed at developers and help them to drive out and check their software designs. It's sometimes said that unit tests ensure you build the thing right, while acceptance tests ensure you build the right thing.

More Examples

From Dan North's Introducing BDD:

Story: Account Holder withdraws cash
As an Account Holder
I want to withdraw cash from an ATM
So that I can get money when the bank is closed

Scenario 1: Account has sufficient funds
Given the account balance is > $X
And the card is valid
And the machine contains enough money
When the Account Holder requests $X
Then the ATM should dispense $X
And the account balance should be
And the card should be returned

Scenario 2: Account has insufficient funds
Given the account balance is < $X
And the card is valid
And the machine contains enough money
When the Account Holder requests $X
Then the ATM should not dispense any money
And the ATM should say there are insufficient funds
And the card should be returned

Scenario 3: Card has been disabled
Given the card is disabled
When the Account Holder inserts the card
Then the ATM should retain the card
And the ATM should say the card has been retained

Scenario 4: The ATM has insufficient funds...
etc..

Another worked example:

Story: Ordering a book via the website
As a customer,
I want to be able to order a book via the web
So that I can get the book without going to a store

Given the book I want is available in the store
When I search for the book
Then the system displays the book and its price

Given the book I want is not available in the store
When I search for the book
Then the system displays the book and offers to order it for me from our supplier

Given the book I want is available in the store
When I complete all the fields in the purchase form
And I enter my credit card details
And I confirm I want to purchase it
Then my credit card is charged with the total cost including sales tax and shipping charges
And the order is stored in the customer database
And I am sent a receipt via email

Even More (Real) Examples

FEATURE:Add a reserve to a claim
As a Claims Manager
I want to add a reserve to a claim
So that we can report this information in our Bordereau, monitor our liabilities and also make provision for payment of invoices

SCENARIO:Adding reserve costs
Given the case is open or re-opened
When I call up the case record
And I want to add reserves for legal and settlement costs
Then the reserve calculation sheet is generated for completion

SCENARIO:Add settlement costs reserve
Given the case is open or re-opened
When I call up the case record
And add a settlement reserve which exceeds my authority
Then the reserve is created
And sent to a higher level of authority for approval
And is visible to the authority

FEATURE:Receive and Pay an Invoice
As a Claims Assistant
I want to pay an invoice on a case
So that the payee can receive payment for services provided

SCENARIO:Case cannot be found
Given I search for a case record
Then I cannot find a case record
And the payment cannot be made
And the invoice is referred to Management

SCENARIO:Case is closed so I can't pay
Given the case is NOT open or re-opened
When I call up the case record
Then the payment cannot be made
And I can refer to the Claims Manager

SCENARIO: Case has insufficient reserves
Given the case is open or re-opened
And the case has insufficient reserves
When I call up the case record
Then I cannot create a payment
Then the case is referred to the Claims Manager for review of reserve
And the payment cannot be made

SCENARIO:Case has reserves
Given the case is open or re-opened
And the case has sufficient reserves
And the payment is entered
When I call up the case record
Then the payment is listed on the case
And is now ready to be authorised