Stories are for telling

In the late 1990’s Kent Beck had a simple idea to solve one of the biggest challenges in software development: communicating the details of what to build. By simply getting together and “telling our stories” we could build shared understanding in the minds of everyone involved.

In the conversation we’d focus not only on what to build, but who would use the software and why. Our goal is to identify the most valuable thing we could most economically build.

Stories get their name from how they’re used and not how they’re written.

Use cards as the tokens for the conversations you’ll use to plan, design, describe, construct, and validate your product.

**Card**
Write your product ideas on cards, one per card.

**Conversation**
Discuss your ideas with others. Let them ask lots of questions. Work together to come up with ideal solutions. The goal is to build shared understanding.

**Confirmation**
Bring models, personas, UI drawings or whatever you like into the conversation. Identify ideal solutions and draw new models. Work towards agreement on what to build. Record that agreement as a set of confirmation tests.

**Construction**
Developers, testers, and others equipped with information from conversations and the shared understanding that comes with it build and test the software.

**Consequences**
Now we’ve got working software to learn from. Those who originally asked for it and the builders evaluate. But, the software was likely for other users. You’ll need to test the working software with them to see if it meets their needs. The goal is learning. And your ideas for improvement start the cycle again.

**“Talk & Doc”**
You’ll have many discussions around stories with team members in a variety of roles. Draw pictures and record details as you do. Bring models like workflow models, use cases, UI designs, or anything else that helps you explain the story. But, be prepared to modify it during the conversation. Draw on whiteboards, model with post-it notes, or record on flipchart paper during your discussions. Keep models from your discussions as mementos to help you remember the details discussed.

**Before you build, agree on what you’re building**
Before the team makes a commitment to build software described by a story, agree on acceptance criteria for the software. Record the answers to these questions:
- What will we test to confirm that this story is done?
- How will we demonstrate this software at a product review?

**What’s on the card**
Use story cards, or items in backlogs they way you might cards in a library card catalog. Write just enough information on them to help you find the rest of the details when you need to. Use the card or list item to organize stories, prioritize, and plan.

On a typical card you’ll find:

- **Short title**
  One that’s easy to read in backlogs and easy say in standup meetings.
  If you catch yourself referring to the story by it’s number, stop it.

- **Description**
  If the title isn’t enough, write a description. Try to include who, what, and why. The template could be handy here.

- **Meta-Information**
  - Estimated development time
  - Estimated value
  - Dependencies
  - Status

**Vacation Photos**
The information, drawings, and models you record during conversations are mementos that help you remember many more details than you can capture. People that weren’t there won’t remember — just like they wouldn’t recall anything when seeing your vacation photos.

**Shared Understanding**
When we all read the same document or hear the same discussed, we often imagine different things. It’s describing our understanding with words and pictures, and then combining and refining our ideas that leads to shared understanding.

**Shared documents are not shared understanding**

©2013 Comakers LLC, www.comakewith.us, youshould@comakewith.us for more info
Stories: Concept to Delivery
Progressively split and refine stories as you move them from vague idea through to working software

Opportunities
Create an opportunity backlog from product ideas, and customer, user, and stakeholder requests.

Discovery
Use discovery to elaborate, design, and validate product ideas. Your goal is to identify the smallest viable product you can. Discovery work results in a product backlog.

Delivery
During delivery you’ll focus on designing, decomposing, and describing backlog items.

Validation
Review finished software with the team and stakeholders. Validate product parts with customers and users.

Release
After your software is released, continue to measure the product’s performance relative to its target outcomes. The most valuable opportunities come after seeing the product in use.

Opportunity Assessment
Before spending time going into details on any idea, discuss who the product, feature, or improvement is for, what benefit it will bring by building it, and how much it could cost if it’s similar to other solutions we’ve built. Use the results of this conversation to prioritize opportunities, and to make go/no-go decisions.

When splitting stories, think cake
Use each story to describe an aspect of software you can “taste.” That is, once you’ve built it, you should be able to learn something from having done so. Whole features may have value to customers and users. But, it often takes a few stories to add up to a whole feature. The steps for making software are development tasks. Demonstrate, testable software is the result of those tasks. If the software doesn’t have user interface, you’ll need to find another way to show that it works.

Stories
Stories describe something you can deliver and evaluate.

Delivery Tasks
Delivery tasks give the “recipe” – describe the work someone needs to do to create the story.

Product Team Planning
The product team meets routinely to discuss release progress, select stories for upcoming sprints/iterations, and plan the work needed to get stories ready for the delivery team.

Story Workshop
Product team members meet with delivery team members regularly to work through story details and agree on acceptance criteria. Some call these workshops backlog refinement or backlog grooming meetings. But they’re really the story conversations we need to have.

Work like da Vinci to finish on time
When managing a release budget, split larger stories into “opening game,” “mid game,” and “ending game” stories. Try to get the “big picture” as soon as possible. Early versions that are fully formed but immature allow early functional and performance testing. They allow earlier validation that your concept is right.

Development Strategy
Try using a story map to slice a while product or feature into a series of viable releases.

Release Strategy
During Discovery, try using a story map to slice a while product or feature into a series of viable releases.

Stories often have similar recipes, just less of any one ingredient. For example all stories will have some testing, smaller stories should take less time than larger stories.

Smaller stories often have similar recipes, just less of any one ingredient. For example all stories will have some testing, smaller stories should take less time than larger stories.