Requirements, Product Ownership, and Other Misunderstood Concepts in Agile Development

Jeff Patton
jeff@jpattonassociates.com
twitter: @jeffpatton
I have too many points to make... we’ll likely need to cut scope to finish on time

1. Building software isn’t the point
2. The best requirements aren’t written down
3. You can’t just talk about what to build
4. It won’t all fit on a card
5. Shared understanding is only a means to an end
6. There’s no “correct size” for a story
7. Breaking down stories too soon is a bad idea
8. It’s a bad idea for the product owner to make all the decisions
9. Backlogs don’t help you understand your product
10. You’re likely building the wrong thing
11. You’re estimates won’t be accurate
Building Software isn’t the point
IDEAS!
- Products
- Features
- Enhancements
- Specifications
- Requirements

Now

OUTPUT

LATER

Release!

OUTCOME
Minimize This

Now

Output

Later

Ideas!
- Products
- Features
- Enhancements
- Specifications
- Requirements

Maximize This!

Impact

Outcome

Release!
Stories are an antidote to “requirements”

Software development has been steered wrong by the word ‘requirement,’ defined in the dictionary as “something mandatory or obligatory.”

The word carries a connotation of absolutism and permanence, inhibitors to embracing change. And the word ‘requirement’ is just plain wrong.
Agile Development isn’t about building more faster
Maximize outcome and impact
(that’s what we call value)
The best requirements aren't written down
Specifying in writing doesn’t work well

http://www.cakewrecks.com/

Cake Wrecks, book by Jen Yates,
Specifying in writing doesn’t work well

Jen Yates’ Cake Wrecks: www.cakewrecks.com
Specifying in writing doesn’t work well

Solang Starr
Alicia (in Purple)

Jen Yates’ Cake Wrecks: www.cakewrecks.com
Specifying in writing doesn’t work well

Jen Yates’ Cake Wrecks: www.cakewrecks.com
Specifying in writing doesn’t work well

Jen Yates’ Cake Wrecks: www.cakewrecks.com
Specifying in writing doesn’t work well

Jen Yates’ Cake Wrecks: www.cakewrecks.com
Specifying in writing doesn’t work well

Jen Yates’ Cake Wrecks: www.cakewrecks.com
Specifying in writing doesn’t work well

Sometimes mistakes are less funny
Kent’s idea to get better requirements:
Stop it.
Stop exchanging documents.
Tell me your story.

If we we could just talk about this, we could figure it out together.
Tell stories, don’t just write stories

What I was thinking of was the way users sometimes tell stories about the cool new things the software they use does:

“I type in the zip code and it automatically fills in the city and state without me having to touch a button!”

I think that was the example that triggered the idea. If you can tell stories about what the software does and generate energy and interest and a vision in your listener's mind, then why not tell stories before the software does it?
The original idea of a story was simple: use it to facilitate a conversation.

I've written on the card what I want.

We'll need to discover the details of what I need to build today?
Stories have a simple lifecycle

* Ron Jeffries coined the 3 C’s in Extreme Programming Installed
Something special is going on during that conversation
When we share and sign off a document we may believe we understand

I'm glad we all agree.
But, we all take away different things based on our knowledge and preconceptions.
When we externalize our thinking with words and pictures, we detect differences.
When we combine and refine, we arrive at something better
Afterwards, when we say the same thing, we actually mean it.
Shared understanding and alignment are the objectives of collaborative work

Shared documents aren’t shared understanding
Words and pictures help everyone build shared understanding
To build shared understanding, use sketching and recording on walls and whiteboards.
Shared Understanding and collaboration at Atlassian
Shared Understanding and collaboration at Atlassian
What you record during conversations works like a vacation photo.

Looking at it helps you remember details that aren’t in the photo.
What you record during conversations works like a vacation photo.

Looking at it helps you remember details that aren’t in the photo.
If you replace a conversation with a document, you’ve stopped using stories.
Effective story conversations are about building shared understanding.

The best documents help us recall our conversations, they don’t replace them.
You can’t just talk about what to build
Good conversations aren’t easy
It’s hard to break out of the “requirements” mindset
User Stories

That’ll help people remember to talk about users and outcomes, right?
The name helped, but people still talk about features and not problems to solve for people.

output

we build this

outcome

we want this
Say “Hi” to Rachel
Rachel and her team at Connextra have a clever conversation starter

As a Connextra employee, I want to know how to write good stories so that I can submit cards to the planning game that are clear and will be accepted in the next iteration.
The template helps us remember to talk about who, what, and why

As a

I want

so that

Idea!
BUT
What beginners learn isn’t what experts do

When learning to ski, beginners often start with a snowplow. The goal isn’t to master the snowplow but to break away from it.
Really discuss the who’s, what’s, and why’s

**Who** are the users and what benefit do they get?
- Discuss many possible users.
- Discuss stakeholders and others that may not directly use the software.
- Discuss elements of the system, or services

**What** will users do in the future using your software?
- Discuss what users will do with the feature or changes being
- Discuss what the system will do, especially if we’re talking about a backend service

**Why** should your organization build the software?
- Discuss who benefits - it may not be the user
- Discuss why users would benefit
- Discuss why the organization would benefit
Focus discussion and collaboration around who will use and benefit, and how they will “later” after delivery.
It won’t all fit on a card
The original idea of a story was simple: use it to facilitate a conversation.

I’ve written on the card what I want.

We’ll need to discovery the details of what I need to build today?
Stories need to support lots of conversations across lots of project roles.

How do I describe to you what I want?

What are the details of what I need to build today?

How do I describe to you what I want?

What are the details of what I need to build today?
Stories need to support lots of conversations across lots of project roles

What are the details of this feature I need to specify?

What are the features that create ROI?

What are the details of what I need to build today?

How do I describe to you what I want?

How do I verify this work is done?

How do I schedule this work and track its progress?

How do I understand users and their needs?
Cards, or items in lists, are valuable organizing tools.

Your volumes of story information will be:

- written on flip chart paper, whiteboards, or stickynotes
- stored in wikis, hard drives, or agile lifecycle management tools

Friends don’t let friends use Sharepoint.
What’s really written on the card

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 its 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

Story tests
- Before you the story is built, agree on how you’ll test the story to confirm it’s done
- Discuss how you’ll demonstrate the story at sprint review
- Discuss what outcomes you’d expect and measure after the software is delivered

Details
- Attach any other details that help explain the story: UI drawings, business rules, or technical details
- You’ll gather a lot of these details along the way as you have story discussions. “Talk & doc”
A lumberjack comes across a man in the woods trying to cut down a tree with a hammer...
Tools can support 3 activities:
1. Collaboration
2. Retention
3. Tracking
Don’t mix those things up.
Cardboard supports remote collaboration

www.cardboardit.com
Atlassian’s Confluence supports retaining the results of online and offline collaboration.

www.atlassian.com

Jeff Patton & Associates, jeff@jpattonassociates.com, twitter@jeffpatton
VersionOne is great for tracking and analyzing
Keep informal documentation to help recall conversations and agreements

Be cautious about using tools to replace collaboration
Shared understanding is only a means to an end
working software
over
comprehensive conversations
Communication begins the cycle, build and learn to continue the cycle.
Plan to learn something from everything you build
There’s no “correct size” for a story
Standard agile jargon for “big”

Epic
A story that’s large, and usually needs some more details filled.

Theme
A group of stories that work together

Whoa! This is way too big to build in two weeks. Dude, it’s EPIC!

Just talk about it, and break it into smaller parts that work together.
The natural “unit of measure” for stories varies by conversation.

- **Capabilities that generate business impact**
- **A feature that differentiate us in the market**
- **Enough software to help me meet a need**
- **A business rule, a screen**
- **Something I can complete & evaluate in a couple days**

- **Business leader**
- **Product manager**
- **User**
- **BA or UI Designer**
- **Developer**
There’s lots of right sizes

Right-sized for BUSINESS

Right-sized for USERS & CUSTOMERS

Right-sized for DEVELOPMENT
It’s not always clear what’s an epic
Stories are like rocks
Rocks are still rocks regardless of size

- Boulder?
- Rock?
- Pebble?

they're all rocks, no matter how precise we are in our naming.
Stories are still stories regardless of size

They're all stories, no matter how precise we are in our naming.
The right size for a story is the right size for the conversation
Breaking down stories too soon is a bad idea.
User Stories shrink in size and grow in detail as they travel through a funnel

<table>
<thead>
<tr>
<th>Opportunities</th>
<th>Backlog items</th>
<th>Working tested software</th>
<th>Minimal releasable software</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Name</td>
<td>• Priority</td>
<td>• Meets the team’s definition of done</td>
<td></td>
</tr>
<tr>
<td>• Target customer or user</td>
<td>• UI design</td>
<td></td>
<td>• Generates value from its use</td>
</tr>
<tr>
<td>• Value</td>
<td>• Business rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Size</td>
<td>• Acceptance tests</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Release stories**
  - Target release
  - Size as budget
  - UI sketches & prototypes

- **Split & refined stories**
  - Acceptance tests
  - Small enough to complete in an iteration

- **Validated product parts**
  - Vetted with customers and users
  - Evaluated for release readiness
User Stories shrink in size and grow in detail as they travel through a funnel.
Use conversation and collaboration to break down stories just in time
It’s a bad idea for the product owner to make all the decisions.
A cross-functional product ownership team directs discovery

Product owners lead teams through discovery and work together to tackle day-to-day product ownership responsibilities.
Look to product owners for leadership
Look to balanced cross-functional teams to help write and elaborate stories
Backlogs don’t help you understand your product
A Story Map helps organize discussion about user’s experience with our product
A Story Map helps organize discussion about user’s experience with our product. 

**product goals (why build the product)**

**users (what are their goals)**

**backbone (gives structure to the map)**

**workflow (from the user’s perspective)**

**details**

- smaller steps
- alternative steps
- UI details
- technical details

Gary Levitt, owner & designer of Mad Mimi
By arranging tasks written on cards spatially, we can tell big stories.

Place user’s tasks in the order you would explain what someone does when using your product.

Don’t get too uptight about the order.

---

Jeff Patton & Associates, jeff@jpattonassociates.com, twitter@jeffpatton
Look for groups of tasks that make up activities

Look for tasks done by similar people at similar points of time and usually directed at a common goal

Label these as larger activity-centric stories
Consider alternative tasks and other details to fill in the map.

Overlap user tasks vertically if a user may do one of several tasks at approximately the same time.
The map shows decomposition and typical flow across the entire system.

Reading the activities across the top of the system helps us understand end-to-end use of the system. (Talk through just these when talking with people with short attention spans.)

Below each activity, or large story are the child stories that make it up.
Build story maps in small collaborative groups
Use the map for continuous discussion
Discussions drive out more details, validate, and build shared understanding.

Talking through the map with multiple users and subject matter experts helps test it for completeness.
Map stories now and later

Now Story Map
Sometimes called a Journey Map

Later Story Map

OUTCOME

IDEAS:
- Products
- Features
- Enhancements

SPECIFICATIONS
- Requirements
- Specifications

IMPACT

Pains

Questions

Ideas

Jouys or Rewards

OUT PUT

LATER

Release

Jeff Patton & Associates, jeff@jpattonassociates.com, twitter@jeffpatton
Plan by slicing the map into holistic valuable releases
Plan by slicing the map into holistic valuable releases
Plan by slicing the map into holistic valuable releases
Your job is to build LESS software

MVP: Minimal Viable Product
- The smallest product you could build that reaches your target outcomes

It’s NOT - the crappiest thing that could possibly work

It’s a hypothesis until you test it

If it dies in the market, it’s wasn’t viable

Target outcome
Outcome-centric roadmap

MVP

Jeff Patton & Associates, jeff@jpattonassociates.com, twitter@jeffpatton
Use story maps to understand your product or feature’s experience

Use mapping to break down big stories into small viable solutions
You’re likely building the wrong thing
BAD OLD DAYS

Smart person → Big idea → Big solution → Ship it!

Smart person

Big idea

Big solution

lots of time passes...

Customers & users react
Is it as simple as building only the features people will use?

“Clippy” - Booed off the Microsoft Office stage as seldom-used and often despised.

It seemed like a good idea at the time....
Your guesses about the future are probably wrong

Typically about 50% to 80% of all software we ship fails to accomplish it’s objectives.

People like Marty say this stuff is hard
(Marty Cagan, author of Inspired, How to Create Products Customers Love)
Most ideas fall flat

- ~20% - Success!
- ~60% - Little or no impact
- ~20% - Failure! - releasing it hurts us
Lean Startup thinking magnifies learning

VALIDATED LEARNING LOOP

Build

Better Ideas

Learn

Measure with customers & users

Subjective & Objective Data

Minimal Viable Product Experiment

Jeff Patton & Associates, jeff@jpattonassociates.com, twitter@jeffpatton
Learn most critical things first to speed learning cycle time!
MVP = minimal viable product

1. Smallest viable product for a target market
Lean Startup Cycle

Build → Measure → Learn
Lean Startup's

MVP = minimal viable product

[experiment]

z. Smallest viable experiment to validate a product concept
Release → Build → Measure → Learn → Release
Release → Build → Measure → Measure → Earn
LS Loop Inside the Market Value

Loop

Release → Build → Measure

Earn

Learn
Explicit release step
Explicit measure step & metrics
Nothing leaves the board until there’s been a discussion on what we’ve learned

Snag-a-Job’s board courtesy of David Bittenbender
Find minimal and viable by releasing different solutions to a small test market
Eric uses his map to plan and deliver a series of minimal viable product experiments.
Use validated learning to build and release small experiments to learn if you’re building the right thing
Your estimates aren’t right
“accurate estimate” is an oxymoron
To release benefit on a schedule we’ll need to budget, and leverage incremental and iterative thinking

(What’s the difference?)
“incrementing” builds a bit at a time

Incrementing calls for a fully formed idea.

And, doing it on time requires dead accurate estimation.
“iterating” and “incrementing” builds a rough version, validates it, then slowly builds up quality.

A more iterative allows you to move from vague idea to realization making course corrections as you go.
Many organizations consider revising the same functionality as failure. Iteration is not tolerated.
Work like an artist to envision and build the product holistically

“Art is never finished, only abandoned.”

-Leonardo DaVinci
Organize work to maximize learning

The inverse of risk is knowledge

Learning earlier about delivery risks helps us finish on time

Alistair Cockburn refers to cutting the small “polishing” stories as “trimming the tail.”

Opening Game
Early stories emphasize iteration and learning. We need to be sure we’re building the right product

Mid Game
Once we’re confident we have the “shape” of the product right, we begin to pile in value

End Game
Over time the value of stories begin to diminish signaling it’s time for release

Organize work to maximize learning
Use a story map to slice out a delivery strategy
Consider these four story splitting heuristics that build up quality

**Bare Necessity**
For the feature to be minimally demonstrable – but not releasable, what is the minimal functionality

*Example: A form with only necessary fields and no validation*

**Capability & Flexibility**
What would add the ability to perform the user task in different ways? Adding in sub tasks that are optionally performed?

*Example: a form with optional fields, date lookup tools, input translation on dates*

**Safety**
What would make this feature safer to use? For both the user, and for the business paying for the software?

*Example: input validation, enforcement of business rules such as credit card validation*

**Usability, Performance, Sex Appeal**
What would make this feature easier to use? More desirable to use? Faster to use?

*Example: auto-completion, sexy visual design, speed keys*

*Adapted from Gerard Meszaros’ “Storyotypes”*
Building up quality iteratively and incrementally ships the best product possible

1. We know each story can be split into at least four parts
2. Early iterations strive to build bare necessities, later iterations build up quality
3. Evaluating readiness based on subjective quality to understand doneness

Product goal: **(in 4 sprints)** ship the best product possible
Product Owners must understand the delivery strategy that leads to a finished product.

Sculpture at various stages of completion, Musée d’Orsay, Paris.
Build up software iteratively and incrementally to release the highest quality possible on time.
What you should take back to the office and leverage

1. Minimize output, maximize outcome and impact
2. Use collaboration to build shared understanding and alignment
3. Focus discussions on who, what, and why
4. Make and keep models and information to remind you of conversations
5. Plan to learn from everything you build
6. The right size for a story is the right size for the relevant conversation
7. Use conversation and collaboration to break down stories just in time
8. Use product owners to lead small discovery teams that focus and valuable, usable, and feasible solutions
9. Use story mapping to help you understand product experience
10. Use validated learning and experiments to learn if you’re building the right thing
11. Build up software iteratively and incrementally to release the highest quality possible on time
What did you take away?
What idea or concept sticks in your mind?

Tweet something using this tag: 
#AgileRequirements

I’ll give away 5 copies of User Story Mapping to the 5 best

“Presentations are never finished, only abandoned.”

-Leonardo DaVinci

OK, he didn’t say that, but he would have if he’d have done conference speaking
Requirements, Product Ownership, and Other Misunderstood Concepts in Agile Development

You should buy this book!

You should attend Agile Day Chicago, October 10th, 2014 at the Sears Tower in Chicago

Jeff Patton
jeff@jpattonassociates.com
twitter: @jeffpatton