Stories

A unit of behavioural change

ThoughtWorks

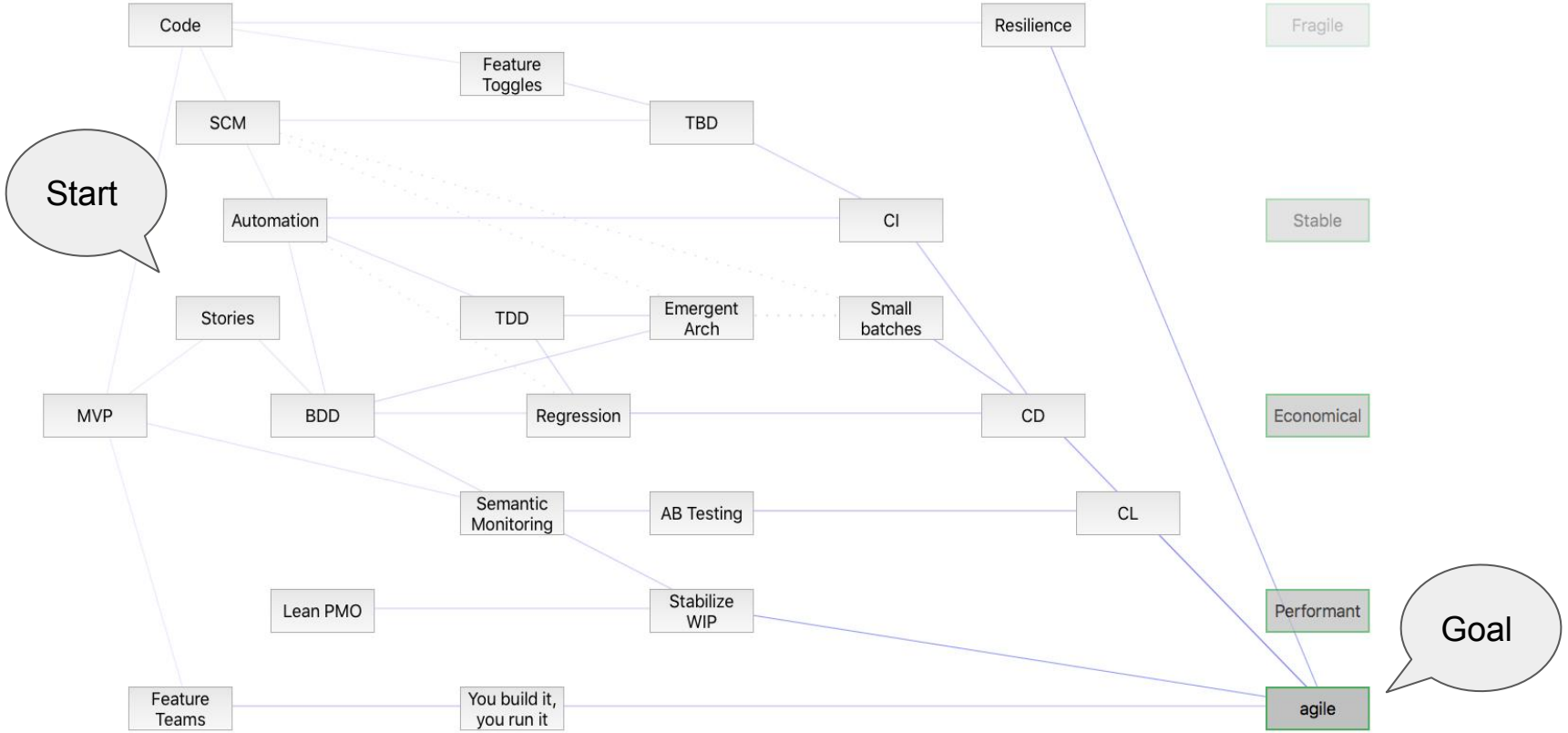
Fannie Mae July 2016

Agenda

- Foundational principles
- Epics, Features, relinquishing control
- Stories, elements, milestones, mechanics
- Do’s and Don’ts

Foundational principles

Their place in our quest



Value proposition

Have fast feedback
in order to survive

Especially if it’s negative

Agility

Facilitate adaptive planning

Economics

Make it economical to introduce change

Extracting value

Apply stressors to business assumptions

Measure all things

Introduce unbiased change

Domain language

Enhance understanding by using a common language

Perspective

“Your job isn’t to build more software faster:
it’s to maximize the outcome and impact you get
from what you choose to build” - Jeff Patton

Epics, Features, Stories

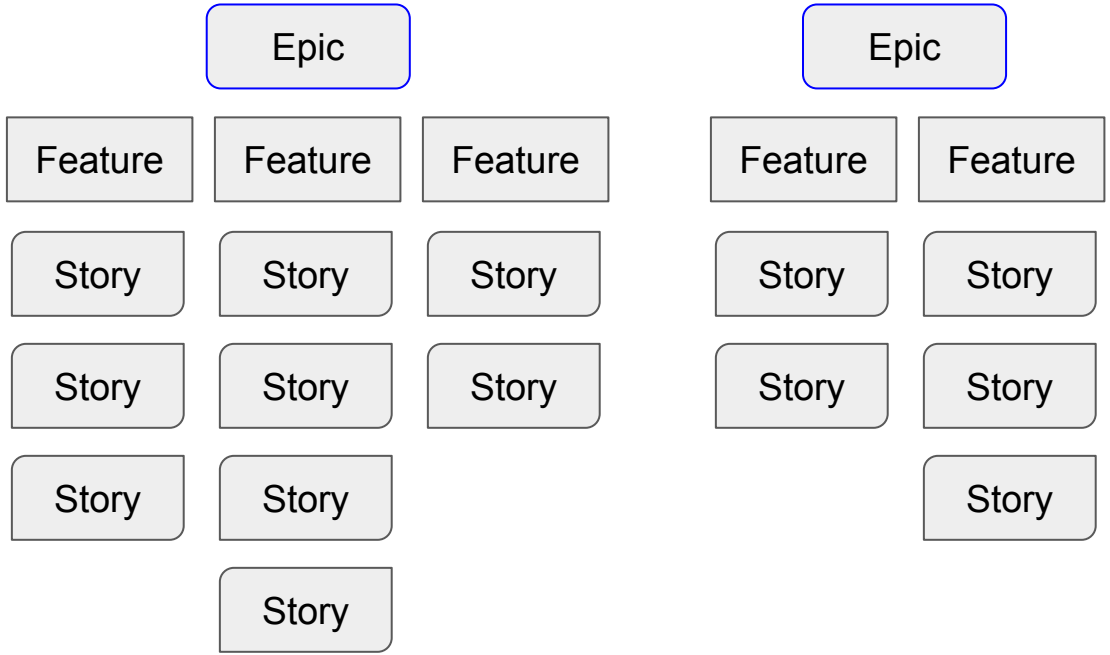
Mental model hierarchy

Our business goal is written as an “Epic”

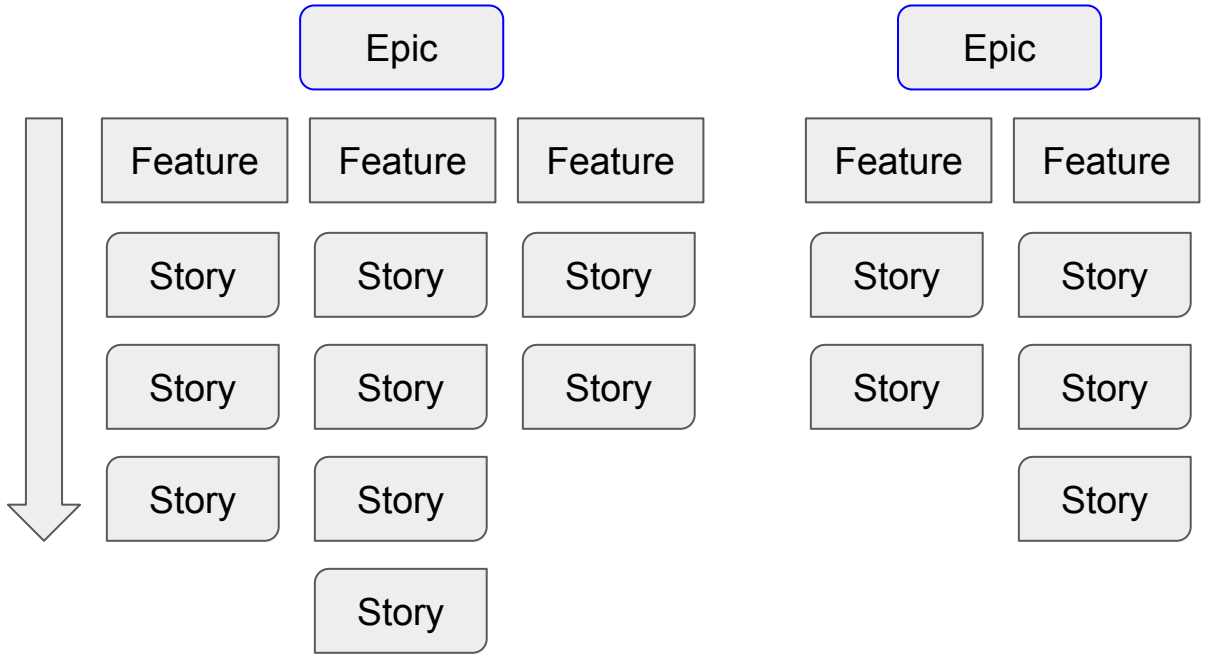
It is broken into “Features”

They are broken into “Stories”

Breaking up complex problems



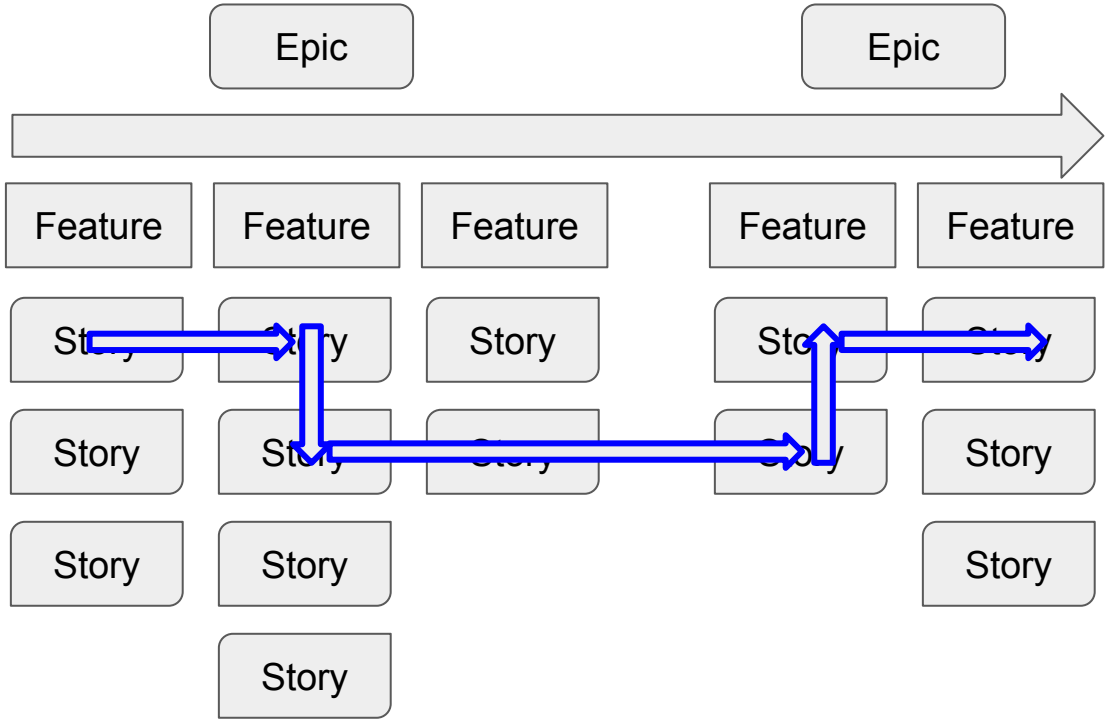
IT Hierarchy



Inverting control

Relinquish control from IT
to product owners

Business Hierarchy



Stories

Delivering value

Deliver business value,
not check-off technical tasks

Value

“Valuable initiatives produce an observable change in someone’s way of working”

- Robert Brinkerhoff, “Systems Thinking in Human Resource Development”

The drivers

- Find our personas, find their needs
- Create a story map
- Create user journeys
- Write narratives from the stakeholders’ perspectives

Specification

The product should have:
Four wheels, gas-powered engine,
steering wheel and a steel body

What am I?

That product is:

Value statement

As someone living on a large piece of land
I would like a way to trim my lawn
So that I can spend time enjoying it

Attributes of a story

- Represents business value
- Written from a user’s perspective
- Placeholder for conversations
- Can be tested
- Can be quantified

Stories are not mentioned in the agile manifesto

Story scope

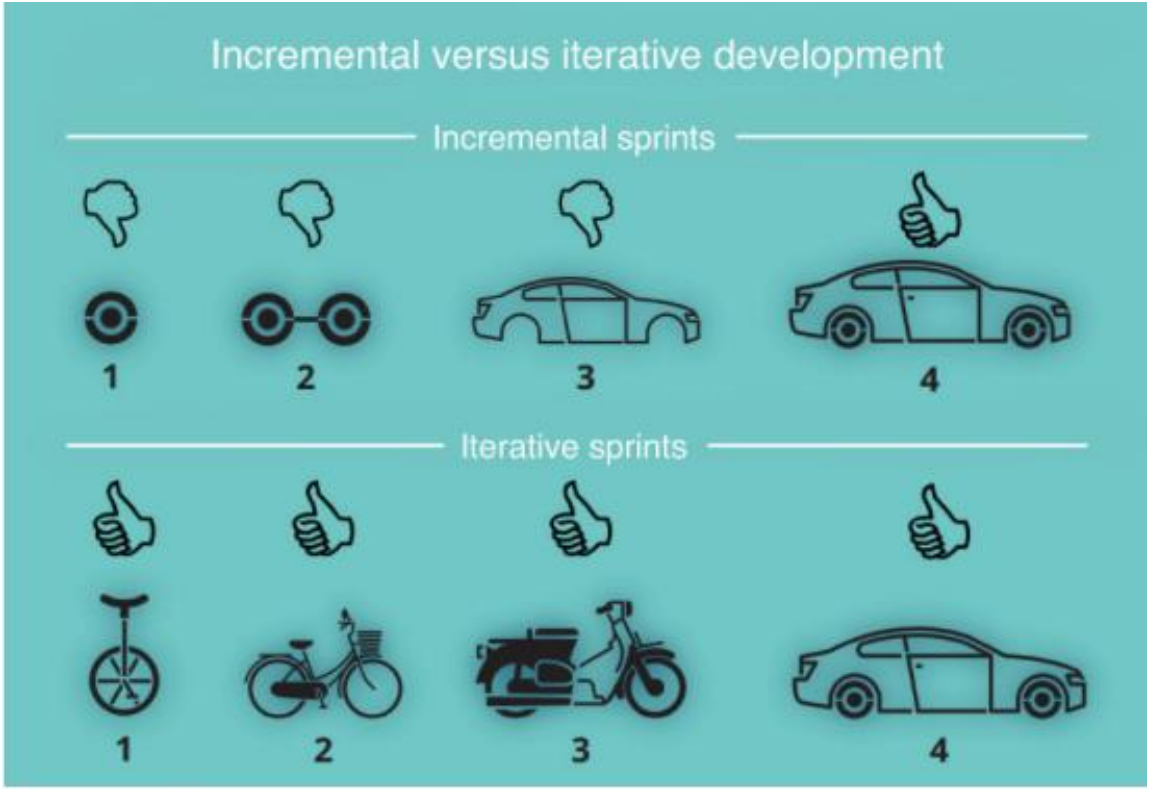
Write stories that minimise risk to the business goal

Story scope should be inversely proportional to business risk

It has to work

Stories must be in production
for us to
measure and validate their value

Incremental vs. iterative



Elements of a story

Focus on the content, not the format

Have a narrative

Talk about and describe what each persona needs and extract its value

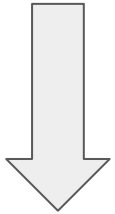
As a <persona>
I want to <action>
So that <value statement>

In order to <value statement>
As a <persona>
I want to <action>

A feature does not exist until a customer finds one

Extract value, not self-fulfilling

As a Call Centre Manager
I want call queue statistics
So I can report on the efficiency of the call centre group



As a Call Centre Manager
I want to see average wait time during peak hours
So I can staff the call centre adequately

Convey context

- Background for the story
- Assumptions
- Flow in respect to other stories
- Out of scope subjects

Define “completed”

Include Cross Functional Constraints, deploy to production, monitor and alert

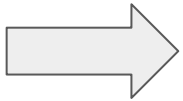
Given <state>

When <event>

Then <validate expectation>

Keep acceptance criteria at a behavioural level

Given I have an account
When I log in
And I enter my username
And I enter my password
And I click ‘Log in’
Then I see my accounts



Given I have an account
When I log in
Then I see my accounts

Living documentation, describes intent

Cross Functional Constraints (a.k.a NFRs)

They are a stressor on architecture that will lead to design with resiliency in mind

Satisfying this stressor will reduce the risk associated with cost-estimation, ideally to zero

The Elephant: Estimation

How long will it take me to get to DC?

The Elephant: Estimation

Point burndown shows that people have been busy, but not necessarily on things that bring value

Size using relative complexity

The Elephant: Estimation, if you must

“Long-term estimates give the wrong impression of precision and promote long-term commitment on scope, which eliminates the biggest benefit businesses can get from agile delivery – adaptive planning.” - Gojko Adzic

- Focus on time and budget as constraints on the stories, and craft them accordingly
- This will change the budgeting models, but those models are inherently wrong anyway

Right-sizing

Strive to size all stories equally
to help predict feature development

Show progress by delivering features

Mechanics

Story kickoff

- Ensure the analyst, dev, tester, PO have a common understanding
- Review acceptance criteria, tasks
- Answer late-breaking questions
- Agree on what, who and how we’ll build & test

Development and tests

- Use BDD in conjunction with TDD (unit, integration)
- Formulate QA examples using AC’s
- Don’t worry about documenting tasks or bugs during development

Desk-checks

- Quick feedback with necessary adjustment on work in development
- “Over the shoulder” walkthrough of the current state of work under development
- Ensures that we’re ready to test the story

Formal QA testing

- Exploratory testing in addition to automation
- Quick feedback from QA to Devs without paperwork

Story milestones

- Ready for development
- Ready for extensive testing
- Ready for acceptance

Ready for development

- Story entered and tracked in our system
- Data-sets and mappings are clearly defined
- Story narrative is complete to the team’s expectation
- Questions by dev/QA/stakeholders have been answered
- Story has been prioritized and accepted (into sprint)
- (Story has been estimated)

Ready for testing

- Desk-check agreement with QA, Dev, PO
- Code & Data have been promoted to QA environment
- Unit & Integration tests ran for newly developed code
- Acceptance criteria were met
- Code is in SCM, was built and passed all tests
- Documentation, regardless of owner, is completed

Ready for acceptance

- All QA test cases have been executed
- All deferred defects are clearly documented, including steps to reproduce and expected outcomes
- All defects that need to be fixed have been fixed and re-tested and passed by QA

Ship it!

to have fast feedback in order to survive

Do’s and Don’ts

Don’t

- Assume that everyone has the business context
- Promote tasks to stories, nor even document them
- Detail technical solutions
- Don’t split by technical convenience
- Attribute great importance to story points
- Set fixed rules, you won’t be agile (small ‘a’)

Do

- Involve everyone when discussing stories
- Write in the context of personas
- Try to describe change in their behaviour
- Use a business language, not a technical one
- Group stories by theme (and set sprint by theme)
- Prioritise by business outcomes, not technical feasibility
- Solicit feedback from real users

Thank you!

Please do not hesitate to contact your agile coaches or the ThoughtWorks team.

itamar@thoughtworks.com