Delivery Techniques

Story Writing

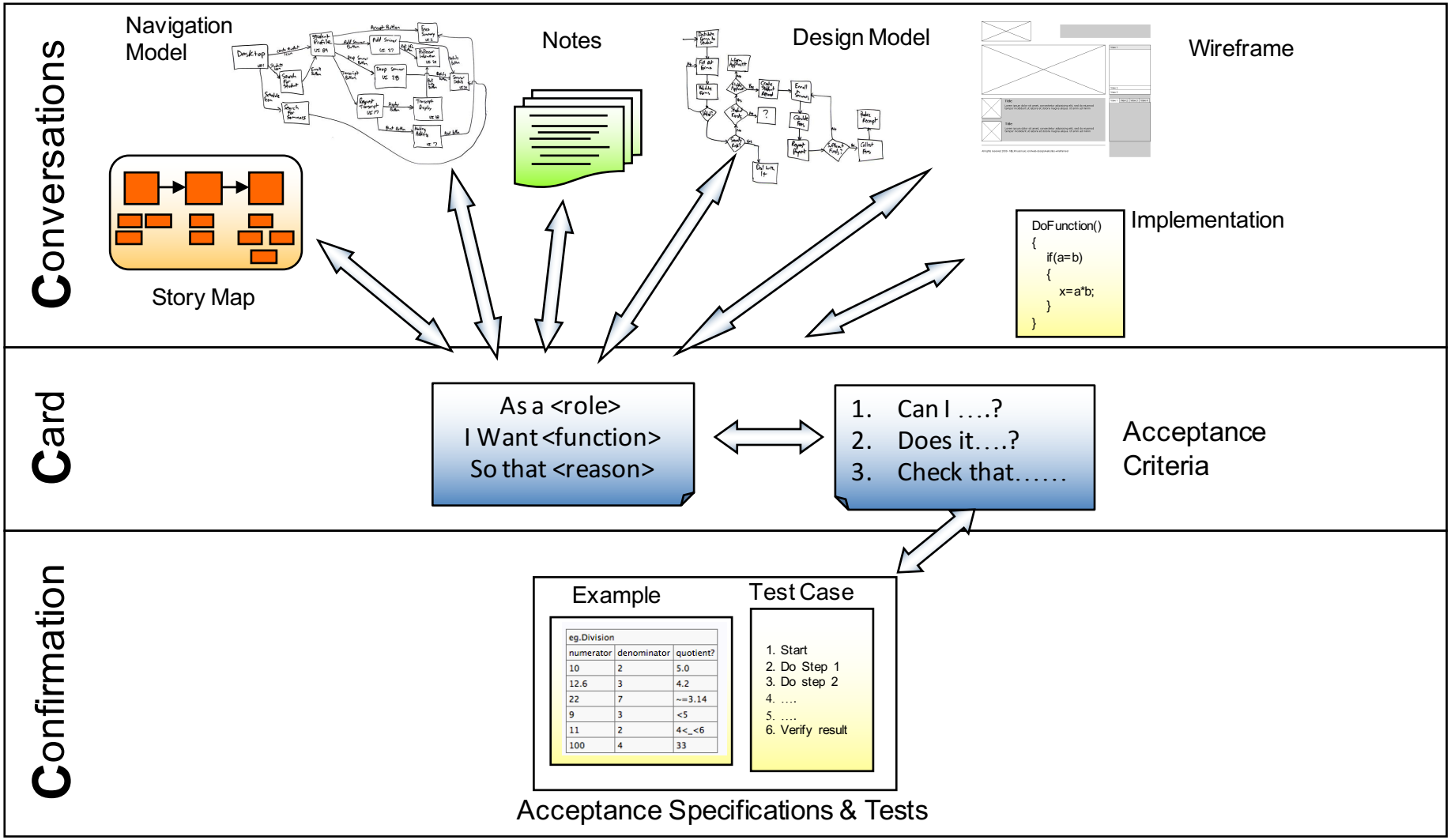
What is story writing?

- Story writing is a requirements elicitation technique that embraces the fact that not all information about the requirement is known up front
- It may also be considered an Agile approach to business analysis
- Story writing is carried out in a story writing workshop ideally by the business experts (not proxies)
- The outputs of a story writing workshop is a set of user stories written in enough detail to provoke further conversations

User Stories

- Basic unit of scope from a customer point of view
  - (**Who**, **What** and **Why**)
- A "promise" for a more detailed conversation later
- The complete Story consists of 'three Cs'
  - **Card** – short-hand physical token
  - **Conversations** – between team and customers about the detail
  - **Confirmation** – Acceptance Criteria and Acceptance Tests
- Created by Business Ambassador/Business Analyst (or other business role)
- Stories can be used beyond software e.g.
  - Stories for research work, document creation etc.
  - Coaching Stories for Agile Enablement work

Evolution of a story



Example Story Card - Front

Maintain Policy Bank Details

As a Member Service Advisor

I want to be able to maintain policy level bank details

So that we can collect monies owed from the correct bank account

- Who: the role that needs the functionality
- What: the functionality required
- Why: business reason for wanting the functionality

Example User Story

User Story examples

- **As a** student, **I want** to find my grades online **so that** I don't have to wait until the next day to know whether I passed

An Abuser story

- **As a** hacker **I want** to capture users personal details **so that** I can steal their identity.

Important facts about stories

- There is NEVER enough detail in a story for the development team to develop from
  - The story card is only part of the Story – remember the 3C's
- The technique is designed to encourage conversations
- Acceptance criteria help provide more of the detail required by the testers and developers

Acceptance Criteria … a reminder

- These details should include business 'assumptions'
- Basic criteria of when a Story is 'done' from functional perspective
- Not a complete, exhaustive set of tests

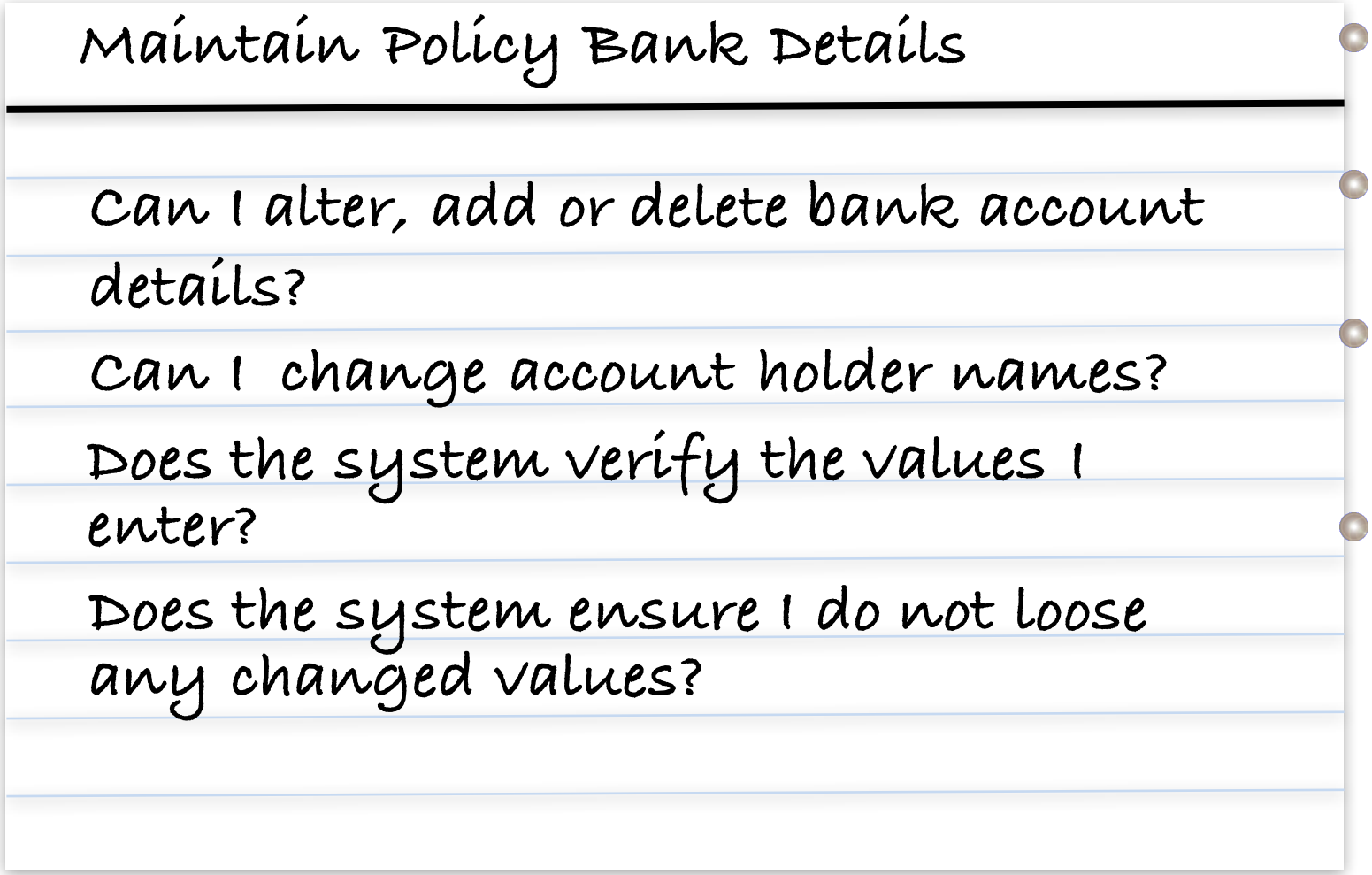
For a Story

- **As a** customer **I want** to use a credit/debit card **so that** it is easy for me to pay

Acceptance Criteria

- Can I use Visa/Delta/Electron, Amex, Master/EuroCard, Solo/Maestro?
- Does it deal with good, bad and missing long ID number?
- Does it deal with good, bad and missing 3-digit Security Number?
- Does it reject expired cards?
- Does it deal with different purchase amounts? (including rejecting one over the card limit)

Example Story card - Back



- High-level Acceptance Criteria
- Can be posed in the form of questions
- Not detailed tests – testers will come-up with those
- But must be 'testable' to show that the story has been implemented as expected

Considerations for Acceptance Criteria

- What else do the developers need to know about this Story?
- What am I assuming about how this story will be implemented?
- Any circumstances where this Story should behave differently?
- What can go wrong with this Story?
- Convey your expectations to the Developers
- Work with Team to define detailed tests

INVEST

- Stories should be written so that they are:

  - Independent of each other
    - Can be implemented independently and 'moved around' when planning
  - Negotiable
    - Start point for collaboration – not too much detail
  - Valuable
    - To customer/end user – the "so that…"
  - Estimable
    - Enough information to provide a 'rough' sizing
  - Small Enough
    - For current timeframe and, at lowest level, must comfortably fit inside a timebox with a number of other stories
  - Testable
    - So we know when the story has been fulfilled

Independent

- Interdependencies complicate prioritisation, estimating and planning
- Some interdependencies are inevitable
- But seek to minimise dependencies between Stories
  - So they don't overlap conceptually
  - So we can schedule them in any order
- Options when faced with interdependent Stories
  - Combine them into a larger independent Story
    - works if the combined story is less than 5 days effort
- Split along a different dimension to remove dependency
  - Record two estimates
    - Based on whether each story is done first or second

Negotiable

- Negotiable... and Negotiated
- Not an explicit contract for features
- Details will be co-created
  by Product Owner and other Team members
  - Through discussion and modelling
  - Joint working face to face
  - Customer, developer and tester perspectives
- A good story captures the essence, not the details.
- Over time, the card may acquire notes, models, test ideas, etc.

Valuable

- Not to just anybody .... valuable to customer, user or purchaser
  - All connections to the database are through a connection pool

  is not as good as
  - As a product purchaser
  - I want 50 users to be able to log-in with a five-user database license
  - So that I don't have to spend a lot of money on database licenses
- Others (legitimate) concerns must highlight value to customer
- When splitting stories must still deliver customer value
  - Horizontal slice is not a true sample
  - A vertical slice is better

Estimable

- Can't be and doesn't need to be exact
  - Enough to help rank and schedule story's implementation
  - Enough to enable team to make a commitment to develop
- Being Estimable is a function of:
  - Size
    - Bigger stories are harder to estimate.
  - Being negotiated and elaborated
    - It's hard to estimate a story we don't understand.
  - The team
    - What's easy to estimate will vary depending on team's experience
- If an otherwise good, small Story cannot be estimated
  - Split it into a (timeboxed) "spike" of investigation to enable a good estimate, followed by the rest of the Story to implement the feature

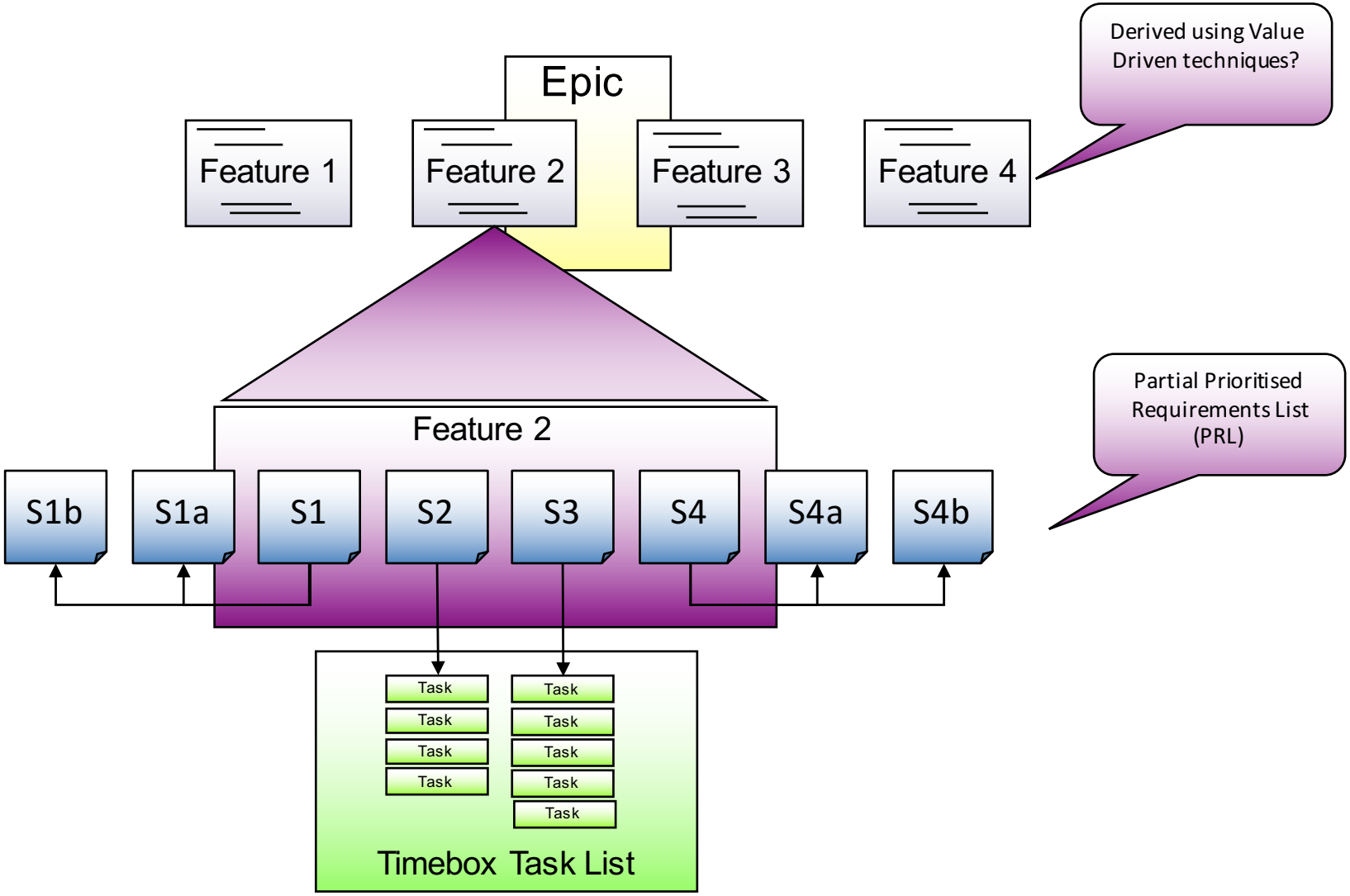
Small enough

- 1-5 days of effort – makes it clear we understand the scope
- Larger stories (a few weeks or a month) are implicitly uncertain
  - Investigate complex stories through Spikes of activity
  - Break-down compound stories into separate individual Stories
- Smaller stories tend to get more accurate estimates.
- Story descriptions must be small as well
  - A couple of sentences on the Story Card
  - A few criteria on the reverse
- Cards are tokens promising a future conversation to elaborate the details

Testable

- Writing a story card carries an implicit promise
  - "I understand what I want well enough that I *could* write tests for it."
- If we don't know how to test something, this may indicate that
  - Story isn't clear enough
  - It doesn't reflect something valuable
  - Need input from someone with specialist testing skills
- Non-functional requirements can be treated as acceptance criteria to be tested
  - Deciding how to implement such tests will help define true needs
- Tests evolve iteratively through feedback
  - To include more detail and become a full set of tests for the Story
  - Cycle of proposing, estimating & implementing Stories

Story Evolution



Value Stories

- As a starting point (high level stories) we may choose a value driven approach which derives from the product vision
- Links the Feature to the Vision by specifying the goal
  - In order to <contribute to the vision>
  - As a <stakeholder>
  - I want <some feature or capability>

In order to sell more products, as the Head of Marketing, I want consumers to order goods online and have them delivered to their door.

- We then go on to write user stories to deliver the features
- Thinking about the value first prevents us from focusing on roles and/or solutions
- This encourages a structure that helps us to iteratively reach an understanding on:
  - The business value sought (why)
  - The problem – what needs to be solved
  - The roles and product capabilities (the solution)

Story Writing Tips

- Merging small Stories
  - Story size of 1 – 5 days is best.
  - Teams will sometime list all 'tasks' as 'stories'.
  - E.g. "10 min stories" - not good
  - Re-group these tasks into Stories of 1- 5 days.
- Don't do task breakdown until Timebox Planning
- To keep Stories small, make each one a small incremental step that adds value (e.g. Build a basic version, then enhance it).
- Ensure the people who are to develop the story, understand the story (they could help write it)
- Team should expect to spend approx 5-10% of each Timebox helping to 'groom' the stories for the next Timebox

Complex and Compound User Stories

- Compound Stories
  - Story that is large (> 5 days) because it contains other stories.
  - Split it up into constituent stories
- Complex Stories
  - Story that is large (> 5 days) but seems to be no way to split it up.
  - Probably because not enough is known.
  - Use a small investigative 'spike' to find out more
  - Then split it up into constituent stories