ThoughtWorks®

Story Workshop

Writing Stories

- Use the As an X I want to Y so I can Z format
- Always write it from the business perspective
  - As a credit app approver, I want to view a person's bank balances so that I can gather info for a credit approval
- It should start and end at the UI (we call this vertically slicing a story)
  - It is never 'Record transaction' or 'Add field to DB'
- Keep it intentionally vague enough that's it's obvious more information is necessary
  - Do not give detail – it gives a false sense of accuracy or completeness
- Keep it implementation free
  - It is not 'I want to call this method to…'

Good Stories

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

Independent

ThoughtWorks®

- Combine stories if they can't stand alone, or break them out if they contain dependencies that will block the card from getting done in a single iteration.
- As a guide stories should be written around an action to get the right level of independence (and help testability) (e.g. Create an Invoice)

Independent

ThoughtWorks®

- **Story Anti-Pattern**
  Get invoice information from the Invoice database
- **Story Pattern**
  As an Accounts agent, I want to create an invoice so that my company can request payment from the Customer
- **Explanation**
  "Get invoice information from the Invoice database" doesn't really stand on its own. The customer really wants a printed invoice so they can send it to the customer. Combine it with the card that says 'Print invoice' (I'm sure it was there). If getting the info from the db and printing all of it on the invoice is too much in a single iteration, split the card vertically. Get a subset of information from the db, get it to print, and then play another card to get more information, print it, until the invoice is done.
- Never split a card horizontally (don't just 'save to db' or 'retrieve from db'). There are many reasons, but the best is that cards should have business value and be showcase-able to the customer, and horizontal cards usually don't / aren't. A good way to know if you're slicing horizontally is if you don't end up on the UI at the end of the story.

Good Stories

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

Negotiable

ThoughtWorks®

- Keeping stories negotiable really means leaving room for flexibility. They should be succinct, but they shouldn't be precise.
- Leaving room for what the detail of the story is gives the customer, analyst and developer yet another reason to have a conversation - what does the story really mean?
- Stories that sound too precise give a false sense of accuracy - e.g. if it has so much detail, it must be exactly what we need to do. That isn't and shouldn't be the case. It's a placeholder for a conversation and for negotiating the details when you it gets close to development.

Negotiable

ThoughtWorks®

- **Story Anti-Pattern**
  - Display summary results for a patient including date, result status and results indicator (normal or abnormal).
- **Story Pattern**
  - As a physician, I want to see summary results for a patient so I can quickly see if the results are normal
    Notes: Actual info to display needs more analysis. Will include date, result status and result indicator (normal or abnormal). There are no other integration points.
- **Explanation**
  The anti-pattern actually reads pretty well, doesn't it? You might pick up this story and think it's close to be ready for development. But that wasn't the case on the real project.
- The story actually required the display of a lot more information than what was described on the anti-pattern, and that data needed quite a bit of analysis. The story actually went to three iterations because the team was forced to replay the story twice in order to get all of the result information they needed to display.
- Stay away from adding detail to the story. Use the notes to write down what you've learned and make it clear that isn't the whole enchilada.

Good Stories

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

Valuable

ThoughtWorks®

- The story should add value to the business - it should be related to something the business needs to achieve, and is often action oriented (like allowing a customer to order an item from the website).
- Business value is relative, and some cards will add more value than others, but as you'll see in the below examples, when we say valuable, we're saying focus on what the story should be helping accomplish. Do not focus on what the story should be 'doing'

Valuable

ThoughtWorks®

- **Story Anti-Pattern**
  - Turn font red if payment is overdue

- **Pattern**
  - As an Account Manager, I want to see a visual indicator on the customer's screen of any invoices that are overdue so that I can send an overdue notice
- **Explanation**

There are many things wrong with the anti-pattern. The most obvious is that we don't know who needs the font red or why. It may not even be that they want red font - it's just what the system currently does.

- Do the following to help:
  - Isolate the functionality needed (e.g. see a visual indicator for overdue invoices)
  - Identify the role(s) that are interested in the action (often roles have different needs)
  - Identify the reason (the business value) for the request

  This enables you to:
  a) get consensus on what indicator everyone wants to see
  b) allows you to understand why the indicator is needed.

Good Stories

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

Estimable

ThoughtWorks®

- Writing stories that are estimable is fundamental to accurately assessing how long a project will take, and at the iteration level, whether the story can be completed in a single iteration.
- Stories need to be small enough to be estimable, but just as importantly, be at a consistent granularity. Writing similar stories where one is estimated at 10 days, and the other has been broken into five stories at two days each is confusing to the estimators, and often results in inconsistent estimation.
- For example, there is almost always overhead associated with starting a new story, even if it's psychological, so the likelihood that the five stories above being estimated at 2 days apiece is low - more likely they'll be estimated higher to account for each being a 'new' card.

Estimable

ThoughtWorks®

- **Story Anti-Pattern:**
  - As a cashier I want to take payments via credit card so I can make a sale
- **Story Pattern:**
  - Story 1a: As a cashier, I want to take payments via American Express so I can make a sale

    Story 1b: As a cashier, I want to take payments via Visa or Mastercard so I can make a sale
- **Explanation**
  The anti-pattern is just too big to estimate. Any number you throw out there isn't going to tell you anything. Spike the story a bit (analysis spike) and take a look at the interfaces.
- Notice this story has 3 credit card types but only 2 stories. Turned out the Visa and Mastercard interface were so similar there was little additional work involved in getting Mastercard working.
- This is a point toward consistent granularity. The team could have created a 3rd story 'Take payment via Mastercard' but at the master story level it would have thrown the granularity off. It made more sense to roll them up and break them down at the iteration level if needed later.

Good Stories

- Independent
- Negotiable
- Valuable
- Estimable
- **Small**
- Testable

Small

ThoughtWorks®

- Breaking stories into the right level of detail is fairly easy if you follow a few guidelines.
- A story shouldn’t span more than a single iteration. If it does, break it up.
- When you break up a story, break it up vertically, not horizontally.
  - Do not think horizontally (e.g. get the persistence layer working). Think vertically.
  - An easy rule of thumb is if there's a UI, the story should start and end at the UI.

Small

ThoughtWorks®

- **Story Anti-Pattern - Multiple Stories in One**
  - Process payment from customer
- **Story Pattern - Multiple Stories**
  - As an Accounts agent, I want to enter basic payment information and have it displayed so I can verify it has been recorded correctly
  - As an Accounts agent, I want basic payment information processed centrally and see it on the mainframe so the customer gets credit for the payment
  - As an Accounts agent, I want the customer's payment to show up when I search the existing reporting database so I know whether the payment will display on their invoice
  - As an Accounts agent, I want the customer's payment to display on their invoice so they know we have received the payment
- **Explanation**

  Large stories are often stories that are actually many stories in one. Break them up into smaller vertical slices of what's actually needed. Remember to get the right level of granularity - don't break them up too small.

Good Stories

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

Testable

ThoughtWorks®

- There are two levels where a story must be testable. First, the original story (As an X, I want to Y so I can Z) must be specific enough so that even without writing detailed acceptance criteria, you'd understand the happy path to test the card.
- The second level occurs if the acceptance criteria are added to the story itself (on the back of the card, in the story spreadsheet, somewhere). The project is usually small and the cards aren't complex enough to expand into a narrative. The acceptance criteria are simply added to the story and it goes into development.
- For both levels, what's written should be possible to automate - no fuzzy logic, no esoteric distillation of the requirements. Anyone that reads it should be able to take what's been written write solid tests.

Testable

ThoughtWorks®

- **Anti-Pattern**
  - Get external link types
- **Pattern**
  - As a user, I want to see the links to external websites that I have saved for myself display on the home page when I log on, so I can easily do research using these links.

- **Explanation**

  How do you test 'get external link types'? This story put the developers and testers in quite a bind. Rewriting it with a business focus made it a piece of cake

How can we help?

ThoughtWorks is a global custom software solutions consultancy trusted by many of the world's leading businesses with their most complex and critical systems. We deliver consulting grounded in delivery expertise, build custom applications and help organisations across all market sectors to drive IT efficiency – working to an exceptionally high standard.

Contact us

Brett Ansley
07754613494
bansley@thoughtworks.com
www.thoughtworks.com

ThoughtWorks®