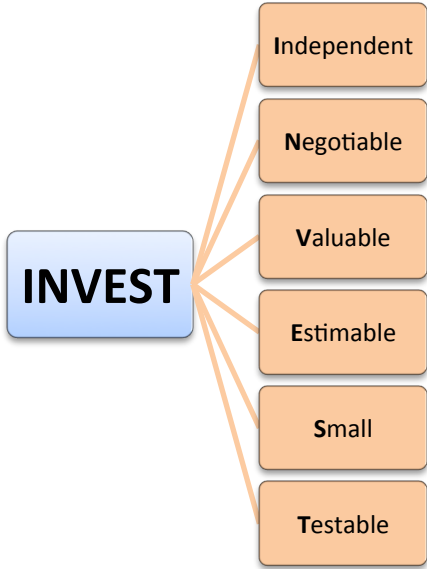
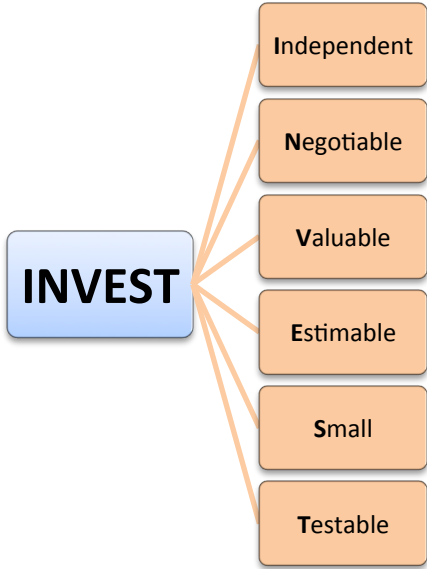
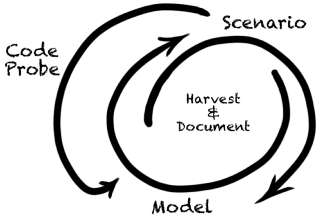


Stories for Design and Delivery

What is a story?	Characteristics of useful stories	Why stories?
<p>A story describes <i>product functionality</i> from a <i>customer's perspective</i>. It is a collaboration tool - a <i>reminder to have a conversation</i> about what the customer needs so the team can design it well & deliver it quickly.</p>		<ul style="list-style-type: none"> • Supports satisfying the customer through early and continuous delivery of valuable software • Shifts the focus from writing to talking - words are imprecise • Supports and encourages participatory domain-driven design (DDD) and user experience design (UXD): involve users, domain experts and stakeholders (i.e. customers) in a creative, iterative, collaborative design process • Describes concrete business reference scenarios, equally understandable by developers and customers in a common, shared language • Customer ranks stories in team's work queue according to relative business value, and team designs iteratively & delivers incrementally • The right size for planning – level of detail is based on implementation horizon
What is a helpful pattern to follow?		
<p>"As a <user role>, I want to <goal> so that <benefit>"</p> <p>e.g. <i>As a nurse practitioner, I want to add an appointment to my schedule so that no patient has to wait more than 5 minutes for treatment</i></p>		
Start first with benefits and goals		<ul style="list-style-type: none"> • Supports satisfying the customer through early and continuous delivery of valuable software • Shifts the focus from writing to talking - words are imprecise • Supports and encourages participatory domain-driven design (DDD) and user experience design (UXD): involve users, domain experts and stakeholders (i.e. customers) in a creative, iterative, collaborative design process • Describes concrete business reference scenarios, equally understandable by developers and customers in a common, shared language • Customer ranks stories in team's work queue according to relative business value, and team designs iteratively & delivers incrementally • The right size for planning – level of detail is based on implementation horizon
<p>Consider each user role and identify the goals that user role has for interacting with the software. <i>Identify stories as core, supporting or generic subdomain</i>. Focus on <i>what</i> is needed and <i>why</i>, not how.</p>		
Model exploration		
<ul style="list-style-type: none"> • Avoid design fragmentation when splitting stories by doing model exploration when needed • Exercise the ubiquitous language in stories. • Look for key business examples in user stories to source as reference scenarios for modeling 	<h3>Augment when appropriate</h3> <ul style="list-style-type: none"> • User stories are not the end-all, be-all for representing "requirements" • Augment them as appropriate with: <ul style="list-style-type: none"> ○ Design documents/sketches ○ Proof of concepts, code probes ○ Photos, screenshots, mockups ○ Examples of inputs and expected result (be specific!!!) ○ Business rules, data dictionaries, use cases, glossaries, diagrams, spreadsheets 	<h3>Why user roles in stories?</h3> <ul style="list-style-type: none"> • Broadens scope from looking at only one user • Allows users to vary by: <ul style="list-style-type: none"> ○ How they use the software and for what ○ Background ○ Familiarity with the software/computers • Used extensively in usage-centered design (c.f personas) • Advantages <ul style="list-style-type: none"> ○ Users become tangible – software solves the needs of <i>real people</i> ○ Incorporate roles into stories

Independent ¹	Estimable	Common patterns for splitting stories ²
<ul style="list-style-type: none"> Identify dependencies – they make prioritizing and planning more difficult “Slice the cake” – each story must have a little from each system layer <ul style="list-style-type: none"> Exercising each layer of the architecture reduces risk of finding last-minute problems in one of the layers Possible to release application with only partial functionality 	<ul style="list-style-type: none"> Don’t get hung up on estimation or traceability, focus on <i>delivering</i> A story might not be estimable if: <ul style="list-style-type: none"> Developers lack domain knowledge Developers lack technical knowledge The story is too big (see below) Stories used in project planning Developers are responsible for estimating Technique such as planning poker or planning game can be useful for making estimates Story points are a <i>relative measure of effort/uncertainty/risk</i> in implementing a story 	<p>Workflow Steps – <i>potential need for modeling here if core domain</i> As a content manager, I can publish a news story to the corporate website ...I can publish a news story directly to the corporate website ...I can publish a news story with editor review ...I can publish a news story with legal review</p> <p>Business Rule Variations – <i>potential need for modeling & UXD here</i> As a user, I can search for flights with flexible dates ...as “n days between x and y” ...as “a weekend in December”</p> <p>Break Out a Code Probe or Spike/POC – <i>in core domain (avoid otherwise)</i> As a user, I can pay by credit card ...Investigate credit card processing ...Implement credit card processing (as 1 or more stories)</p> <p>Simple/Complex – <i>is complex stuff within core domain? Potential need for modeling here. Otherwise, avoid complexity in supporting & generic</i> As a user, I can search for flights between two destinations ...specifying a max number of stops ...including nearby airports</p> <p>Major Effort – <i>supporting? maybe simple case is good enough</i> As a user, I can pay for my flight with VISA, MasterCard, Diners Club, or American Express ...I can pay with one credit card type (of VISA, MC, DC, AMEX) ...I can pay with all four credit card types (VISA, MC, DC, AMEX)</p> <p>Data Variation – <i>supporting? maybe simple case is good enough</i> As a content manager, I can create news stories ...in English ...in Japanese</p> <p>Data Entry Methods – <i>collaborate with UXD</i> As a user, I can search for flights between two destinations ...using simple date input ...with a fancy calendar UI</p> <p>Defer Performance – <i>leverage domain events & aggregate boundaries?</i> As a user, I can search for flights between two destinations ...(slow – just get it done, show a “searching animation”) ...(in under 5 seconds)</p> <p>Operations (eg. CRUD: Create/Read/Update/Delete) – <i>basic operations unlikely to be in core domain, is it supporting subdomain?</i> As a user, I can manage my account ...I can sign up for my account ...I can edit my account settings</p>
Negotiable	Small	
<ul style="list-style-type: none"> Stories are not: <ul style="list-style-type: none"> Written contracts Requirements (they express customer needs that software can help fulfill) Don’t include all details, otherwise gives impression of: <ul style="list-style-type: none"> false precision or completeness no need to discuss further 	<ul style="list-style-type: none"> Small stories for implementing in the near future, and higher-level (larger) stories for further out. Large stories (aka “<i>epics</i>”): <ul style="list-style-type: none"> Do domain distillation: separate what is core from what is supporting and generic Often hide assumptions that should be made explicit Hard to estimate and to plan - won’t fit in a single iteration, so split into smaller stories Bugs can be turned into user stories if this is helpful 	
Valuable	Testable	
<ul style="list-style-type: none"> Identify if story relates to core, supporting or generic subdomain Stories must be valuable either to users/domain experts <i>Completed</i> stories have more value. Finish stories early and often. 	<ul style="list-style-type: none"> Drive important design decisions test-first with unit tests Specify <i>acceptance criteria</i>: these tests demonstrate that a story meets the customer’s expectations, they define the conditions of satisfaction for the story <i>Use specific, concrete, actual business scenarios</i> for modeling and testing Where possible, automate acceptance tests 	

¹ User stories content adapted from Mike Cohn, *User Stories Applied* (Addison Wesley: 2004).

² Adapted from *Story Splitting Cheat Sheet* by Richard Lawrence of Humanizing Work (<http://www.richardlawrence.info/2009/10/28/patterns-for-splitting-user-stories>)

³ DDD Whirlpool adapted from <http://domainlanguage.com/ddd/whirlpool>