User Story

I read all the various comments about user stories and sense there is a serious lack of consensus regarding just what a UserStory is. English speakers typically think of a **story** as some kind of narrative that describes a WHO, WHAT, and WHY scenario. So, from that perspective, it is doubtful to me that a story is going to be at the brief, functional level a programmer needs to develop code.

Therefore, I suggest that the **USER STORY** is indeed a narrative that a user tells from her own perspective. This story typically describes some kind of scenario or situation that the user finds herself in. The programmers must then work with the user to derive from that narrative the specific **USER TASKS** that the user must to perform with the software to successfully accomplish the story.

For example, here is a real **USER STORY**:

> A project manager needs to set up a web-based location where all the members of her team can access engineering drawings, models, research findings, meeting minutes, and schedules associated with their latest project. This will help the team stay in synch regarding the progress of the project. It will also prevent the current problem of storing data in different places that not everyone can access.

And here are **USER TASKS** we derive from the story: 1. Create a project on the web. 2. Create folders in the project. 3. Transfer files to those folders.

And if the user makes any mistakes regarding the name or location of a project, folder, or file, she must be able to: 1. Rename a project, folder, or file. 2. Move a folder or file. 3. Delete a project, folder, or file.

I think you can see that it is the user **TASKS**, *NOT* the **STORY**, that represent those chunks of functionality that can be estimated, prioritized, and then implemented by the software programmers.

What is your take on this Kent? Is there anyone else out there who is comfortable with this concept of **User Story** and **User Tasks**?

-- Martha Roden, Usability Engineer and Interaction Designer

What you have above are all sensible things to say, but they aren't a capital-S Story. A capital-S Story is:

- Testable -- You can write automatic tests to detect the presence of the story.
- Progress -- The customers side of the team is willing to accept the story as a sign of progress toward their larger goal.
- Bite-sized -- The story should be completable within the iteration.
- Estimable -- The technical side of the team must be able to guess how much of the team's time the story will require to get working.

As iterations have gotten shorter (I always start with one week) and teams have gotten smaller, the scope of a single story has shrunk. I certainly feel the need for larger-scale structuring mechanisms. Some folks use "theme", others "big story" or "initiative".

Re: the details above. What is missing from this is "why". Why does the user want to do this. Story writing should be an opportunity for the customers to reflect on and refine their craft.

Kent

I like the idea of the "User Tasks", though in practice, many users (and developers) might initially frame some as "System Tasks". For example: "Require entry of a userID." versus "Enter UserID." I've been casting some user stories as use case scenarios, so my "User Tasks" end up being called "Scenario Steps".

Huet Landry

I am following this discussion on User Stories. But at this time I am a bit confused and want to see a UserStoryAndUseCaseComparison. For example, do we only document the success scenarios in a user story or do we also cater for failure scenarios or alternate flows as in use case? Without documenting the success and failure scenarios both, how can we get a complete picture of the work for the unit and therefore how can we write tests and estimate them?

Please give me some information on how to write user stories and what exactly to include in them?

Thanks

Samudra Gupta

Thanks a ton for that link on user stories and use cases comparison. One more thing I want to know is the use of coloured cards with extreme programming and agile methods. Any pointer on that please? I am seriously embarking on this on my next project.

Thanks

Samudra Gupta

In ExtremeOpenBusiness, a UserStory is given by making all UserInteractions with the real market transparent, inspired by DaveBrins TheTransparentSociety. By transparently collaborating with OpenSource programmers (this is much more than "only" restricting programming to piecewise open releases of software sources to the public), the programmers get their real time-input from the users: e.g. EOB users, who practise SocialDomaining, by exposing their domain portfolios, as public Google Spreadsheets or EditGrid Spreadsheets, via wiki pages. The users describe their needs on these wiki pages, e.g. we want a pay button in each row, that lets pop up a dialogue window, which initiates a transaction of domains and funds.

- the Buyer inputs the addresses of their open domain portfolio and their e.g. paypal account.
- A funds transfer is initiated from buyer to seller.
- The source and target domain portefolios are adapted accordingly.
- DomainRegistrarTransfer is initiated.

In a human dialogue, the wants are refined until everything is crystal clear to the programmers. In this open process now, some users want e.g. to have attached to the spreadsheet a MakeOffersList, which for XP-programmers takes a matter of minutes to code this additional feature. Now some users want to pose this list into a domaining forum, say NamePros, where there is an established community, who trade domains on an internal currency. As all these user-actions are already occurring real-world transactions, these transactions tell the story. These sketchy remarks can be refined in the public. The author leaves a backup of his conceptual input on the following WikiTrailmark (http://trailfire.com/fridemar/marks/224077). -- FridemarPache

---

CategoryStories

In ExtremeProgramming a UserStory is a story about how the system is supposed to solve a problem or support a business process. Each UserStory is written on a StoryCard, and represents a chunk of functionality that is coherent in some way to the customer. [?]

These written cards are preserved through the entire planning and

These written cards are preserved through the entire planning and development process (PlanningGame, ReleasePlan, IterationPlanning). They are given priorities by the customers, they are given estimates by the developers, they are broken down into EngineeringTasks at the time that they are scheduled for development. There are one or more AcceptanceTests, owned by the customers, to give them confidence that the UserStory has actually been completed.

Can someone PLEASE, for the love of all that is right and holy, swap the contents of the UserStory and UserStoryDiscussion I know that some of the information here should be included, however the naming of these two pages is atrocious. -- JoshuaDrake

Nothing's stopping you from doing that.

The way I explain it now is that stories don't have to represent business value to the customer team, but they do have to represent progress. Only the customer team knows what it will consider progress, so they have to do the slicing. -- KentBeck on XpMailingList

Can you give an example of a story that does not represent business value to the customer? Why would customers consider something progress if it does not deliver value to them? -- Chandrashekhar

---

See also UserStoryExamples, UserStoryDiscussion, LimitsOfUserStories, UserStoryAndUseCaseComparison, HowUserStoriesAreExtractedFromUsers, QuestionsAndAnswersAboutUserStories, UserAntiStories, QuestionsAboutXpStoriesAndTasks.....

I think the content of this page and UserStoryDiscussion should be swapped. And continue on in UserStoryDiscussion towards a healthy definition, something like PlanningGame (which is excellent - till then see UserStoryTemplate). Folks reading the conversation below may deduce that UserStories are either not ready for primetime or too vague to be useful.

(Which I would heartily disagree with - UserStories are (at the very least) one concrete instance of a scenario, while UseCases are an abstract overview of all courses of a scenario. And doesn't the brain work better imagining our friend Marsha trying to get thru one task, than imagining what all accountants would want to do in this module for the next X years? Build lots and lots of UserStories, which are easy, and you'll see the UseCase. This paragraph is just me. -- ChristopherGaltenberg)

Now that I've read a little more and asked a few more questions I'm getting a view of a UserStory that's more involved than what's been presented on this page so far. **A UserStory is also a work unit and an estimation unit. In this it's actually considerably heavier - not lighter - than a UseCase.** A UserStory as an estimation unit is inadequate if it can't be properly quantified in the PlanningGame. A UserStory as a work unit is inadequate if it isn't split up into EngineeringTasks, each of which must be properly regression tested and refactored to make the simplest 100% working system. Oh, and as a work unit it has to have an acceptance test too. So UserStory is a word with considerable process baggage in XP, not just a 2-bit UseCase.

So I have a question about the thing as an estimation unit. EngineeringTasks, we're told, have to be able to be estimated as no more than 5, and preferably less than 3 ideal engineering days. **Is there a similar criterion for a UserStory?** -- PeterMerel

From RonJeffries: In CommitmentSchedule, we ask the users to split a story if we estimate it at more than about 3 weeks, on the grounds that we don't understand it. We combine stories willy-nilly if they don't add up to at least a week, for estimation only. (That is, we don't intend to implement them in that order, we just batch them up in piles of a week.)

From KentBeck: Remember also that the role of a story as a work unit comes considerably after the role of a story as a priority/scope decision unit (for which it also needs to be an estimation unit).

I was trying to use your words. I'll try again. The story is used by the customer to make informed decisions about the scope of the next release, decisions informed by the estimated cost of the story and the velocity with which the team can develop. When the story's iteration rolls around, it becomes a work unit by being split into tasks (or not on smaller developments) and a test unit by being coded as a bunch of acceptance tests.

The stories are really the artifact at the heart of the continuing dialog between what is possible and what is desirable. -- KentBeck

Which is exactly as it should be in my humble opinion, and one of the most attractive features in XP. Other "OO" methodologies are not really OO; they're work/data flows. Specifications become Analysis becomes Design becomes Code becomes Deliverables become Tests become Deployed Systems. XP says to me, lookit, this thing you're asking for is what we're estimating and what we're estimating is what we're doing and what we're testing and what we're maintaining and delivering - they're all just methods on this one thing, "User Story". Or maybe I'm going off the deep end, but that's how it seems to me.

I should say that I'm in a bizarre methodological context at the moment; trying to relate XP concepts to RUP and MSF concepts without doing violence to any of 'em on behalf of a certain very large company who shall remain nameless for now. So my perspective is skewed and I may be quite mad - please deal harshly with me. -- PeterMerel

---

From AlistairCockburn: That is a very interesting set of observations, Peter. I am willing to make the assertion that that "considerable process baggage" you are seeing represents the set of practices ChryslerComprehensiveCompensation has put into place to make use of User Stories, and is not intrinsic to the term User Story, nor even to XP proper. I figure that by wording my assertion this way, Kent will find the right place to disagree with me.

I go back to Kent's original line, up above, "Tell me the stories of what the system will do. Write down the name of the story and a paragraph or two." Its intent is to capture some information, specifically not tying the user down to matters of, "Can it be implemented in 3 weeks?", "Is it normalized in terms of the simplest definition of what is needed?", or "Does it have engineering tasks and a regression test?" Those latter things are ways the team looks at and uses the UserStory to work out their plan.

I'll test both my characterizing matrix and my understanding, here. I wrote in http://members.aol.com/acockburn/papers/usecases.htm that one can characterize the many uses of UseCases according to 4 questions:

1. *Purpose*: Is the purpose of use cases to gather user stories, or build real requirements? (the values are stories, or requirements). To answer "requirements" implies that consistency, completeness, non-overlap, non contradiction are major tests of the result. I don't get that these are mandatory elements of User Story. To answer "stories" implies that the purpose of the exercise is to get a view of the world from the user's standpoint, warts, wiggles, and all. In the case of UserStory, the answer is "stories".

2. *Contents*: Are the contents of the use case required to be consistent, or can they be self-contradicting? If consistent, are they in plain prose or are they in a formal notation (the values are contradicting, consistent prose, formal content). My guess is that two user stories would be allowed to contradict each other, if the users disagreed during their interviews. I would further guess that Kent or Ron would chase those users around until they came up with clarifications to their stories that made them non-contradicting. So possible values are "contradicting" and "consistent prose."

3. *Plurality*: Is a use case really just another name for a scenario, or does a use case contain more than one use case? (the values are 1 or multiple). Answer is 1. UserStory does not distinguish multiple scenarios the way Jacobson's or my use cases do.

4. *Structure*: Does a collection of use cases have a formal structure, an informal structure, or do they form an unstructured collection (the values are unstructured, semi-formal, formal structure). Answer is "unstructured". At the end, there is a heap of index cards.

All of which gets me back to the assertion that how C3 chose to manage the User Stories is separate from "What is a User Story?". How did I do? -- AlistairCockburn

---

I guess when I say "heavy" I mean "concrete". A UseCase isn't quantified in terms of the famous four variables; a UserStory is. A UseCase doesn't correspond to a work or testing unit; a UserStory does. A UseCase has no essential granularity constraints; a UserStory does. I see a UserStory as a concrete description of a feature, and the process of developing UserStories as the process of developing by feature; UseCases I see as input to a BigDesignUpFront analysis/synthesis modelling exercise, about which as you know I feel great skepticism.

But I confess I haven't dug far into your particular UseCases Alistair, so forgive me if I'm doing you an injustice. I'm far from a methodologist - I really only learn these things when someone else says, "Okay, I know just how we're going to do this. I read it in this book ..." I want to understand, but at least so far no one's tried to lumber me with Cockburnology or whatever acronym you dub your particular flavour of OoAdSeMtFla :-) -- PeterMerel

---

Alistair- I find myself at odds with your description at what I think is a pretty deep philosophical level. The ghost of Descartes runs through every point you make above. I blessed and released that fine gentleman some time ago. To be more specific:

1. Purpose. The purpose is different at different times. First it is to get the user to tell me what the system has to do first, and what it doesn't need to do at the moment. "the purpose of the exercise is to get a view of the world from the user's standpoint"- I don't believe there is a particular view of the world from the user's standpoint that I can somehow participate in if I just sacrifice a chicken the right way. The user is learning. I am learning. We are learning together. So to say, "...get a view..." is the wrong metaphor. I'm not sure what the right metaphor is, but I'll think about it.

2. Content- "formal notation" doesn't imply "consistent". Every program I've ever written demonstrates that. There is certainly an interesting question to ask about the intersection of notational style, expressiveness, and politics (sometimes I see software engineers using formality to maintain control).

3. Plurality- I don't understand this point at all. Is this the same as asking if the user needs one or more than one acceptance test to be confident that the story is done?

4. Structure- "At the end..." There is no "end". This point seems to be assuming that requirements gathering is a phase, at the end of which you have some artifact that allows you to do something else in the next phase. Requirements gathering is the tangible fruits of the customer's learning, and the learning had better continue as long as the system is alive. This isn't to say that there aren't events along the way, like when you know you have too much stuff for the first release and Development is ready to commit to estimates. But "phase"? Yech...

-- KentBeck

Thanks, Kent. After hearing your gardening analogy, I understand more of what you are saying. Being aware that it was you, and not I, who introduced the idea of a 'requirements phase', I can restate the structural part at least. "Does a collection of use cases have a formal structure, an informal structure, or do they form an unstructured collection?" The collection of user stories forms an unstructured collection, there is never a point at which they are organized into other than a flat structure.

What I mostly notice is that what PeterMerel thinks a User Story is and what KentBeck thinks a User Story is are quite far apart. Since Peter raised the question, I'll step back out and watch you two form a reconciliation. -- AlistairCockburn

---

Hmm. Kent, are we far apart? It's your process and I'm really only conjecturing. I emphatically bow to your knowing what the hell you're talking about, which I don't - I'm just trying to get a grip.

Alistair I did find your comments useful, but I suspect we're coming from different angles on this. Let's hope Kent can recombobulate us without too much difficulty. -- PeterMerel.

Kent here - I don't think Peter and I are at all far apart. What evidence do you see of that, Alistair?

Regarding structure: *The collection of user stories forms an unstructured collection, there is never a point at which they are organized into other than a flat structure.* They are sorted into three piles by priority. They are sorted into three piles by risk. They are sorted into two piles- this release and the future. They are sorted into three week iterations. How is that "flat structure"? There isn't anything like uses and extends, or even prerequisites, but there is a whole lot of structure to me. -- KentBeck

---

Peter says,

- *A UserStory is quantified in terms of the famous four variables (Resources, Time, Scope, Quality, if I recall correctly),*
- *A User Story has essential granularity constraints.*
- *A UserStory is a work unit and an estimation unit. A UserStory as an estimation unit is inadequate if it can't be properly quantified in the PlanningGame. A UserStory as a work unit is inadequate if it isn't split up into EngineeringTasks, each of which must be properly regression tested and refactored to make the simplest 100% working system. Oh, and as a work unit it has to have an acceptance test too.*

Kent says, *'Tell me the stories of what the system will do. Write down the name of the story and a paragraph or two.' ... I blessed and released that fine gentleman (Descartes) some time ago....The purpose is different at different times. First it is to get the user to tell me what the system has to do first, and what it doesn't need to do at the moment.... Requirements gathering is the tangible fruits of the customer's learning, and the learning had better continue as long as the system is alive.... There is no "end".*

Well, I can't reconcile those two sets of statements, I'd never guess they were talking about the same thing. -- AlistairCockburn

They do sound dissimilar, but I think they're just two sides of the same coin. I still leave it to Kent to clarify, but I think I'm just recasting what I've read, not adding anything new. -- PeterMerel

I agree with all three of you, even [mostly] with Peter. Seriously. I gotta think about this. -- RonJeffries

I am shocked and horrified! -- Peter :-)

Seems like Peter is trying to define something concrete and Kent is just talking about ongoing application of heuristics... With heuristics though I still have fear of the blank page, maybe I wouldn't have fear of the blank card. I mean what is the practice and is it productive of good software? To define a definitive practice may be like putting the cart before the horse. Does it produce the focus required to make the software? There seems to be an assumption here that you do a practice in order to create the focus in order to do great software. I don't know if that assumption is valid. Heuristics can't by

themselves create focus. A tiger chasing you creates perfect focus. You're hanging off a cliff by a plum on the branch of a tree, crocodile waiting for you below, that plum is gonna be sweet. Can practices produce focus or is it just a fantasy? -- AnonymousCoward

---

TheInmatesAreRunningTheAsylum recommends something similar to UserStory, with two differences.

The "story" is not a description of a feature in a program, but the underlying real world problem that the software is designed to solve. For example, in the case of a traffic estimation system, the problem may be where to put stop signs. The solution may be a display of a map with dots at busy intersections. But the feature is proposed later in the process than the "story".

The "user" may be fictional, based on a prototypical user, as determined by market research. An example might be "Pamela", a 35 year old interior decorator, who needs software to help her visualize color combinations. Besides being fun, the purpose is to help the user focus on the needs of a segment of users without being misled by the quirks of a particular customer.

-- CayteLindner

Is there any reason why user stories couldn't be as detailed as the users want them to be? *Or should this sort of detail go into the acceptance tests? If the users are technically competent, such as in the case where the users are developers and the product is some sort of development utility or library, are the users still limited to two bits of precision? -- DonaldMcLean*

Now there's an interesting and unusual question: Why can't we write more, if we want to? Usually one has trouble getting anybody to write enough, and that's where the user stories, as *promissory notes* for future conversation, come in handy. They are also unstructured, not requiring any particular form, or the failure coverage of 4-bit use cases.

I can't think of why you couldn't write more, if you want to. I'd be careful, because some engineers can't help but put their implementation concepts right into the requirements story, where they don't belong. To me, UserStory mostly means, simply, "I'll be using the system, and I want to do THIS." -- AlistairCockburn

---

A UserStory is a story, told by the user, specifying how the system is supposed to work, written on a card, and of a complexity permitting estimation of how long it will take to implement. The UserStory promises as much subsequent conversation as necessary to fill in the details of what is wanted. The cards themselves are used as tokens in the planning process after assessment of business value and [possibly] risk. The customer prioritizes the stories and schedules them for implementation. -- RonJeffries

I like this definition, but it doesn't really answer my question. I work in a "Tools" organization that develops libraries and sundries for other developers. Our customers are experienced developers, who may have very clear and specific ideas on what they want. I haven't seen anything in your definition or on the LimitsOfUserStories page that would prevent one of my users from writing a user story like this:

I need a class called "event log file". One of the constructors needs to take a full path name. If no file with that name currently exists, create the file and open it for writing. If a file does exist with that name, verify that it uses the log file format. If it is not a valid log file, throw an "invalid format" exception. If it is a valid log file, open it in append mode.

This user story doesn't specify anything about HOW to implement what is needed, but is very specific about what is needed. -- DonaldMcLean

Interesting... it seems as though the customer (the experienced developer) in the above is perfectly justified in their well *specified user story... it is the business logic which he has control over. For example (from UserStoryExamples):*

> *Employees who are sick more than 3 days go on DAP (Disability Absence Plan). They are paid from their full pay for 190 working days, and then 70% pay up through 270 days. DAP dollars paid must be kept separate from regular pay dollars, for accounting purposes. Entry to DAP is indicated by the JL30 transaction. These are often sent to us late, after the employee has already been paid. The system must retroactively make it look as if the transaction was received on time.*

-- *WilliamUnderwood*

I thought I understood this, with a UserStory being essentially a UseCase with unnecessary (due to working conditions) details omitted. But a UseCase is most emphatically a description of a particular type of interaction with the system, while some comments by RonJeffries sound as though he believes that every kind of requirement can be written as a UserStory, and that no other requirements need exist. Am I confused? Is Ron? Are we simply talking past each other? -- RussellGold

I set up LimitsOfUserStories to try to coalesce and sharpen the discussion that is now scattered over UserStory, UserAntiStory, TheExtremeProgrammingWayToHandleUserAntiStories, and ThereAreNoUserAntiStories. -- AlistairCockburn

---

I have been using the Task Object Model technique as described by Ian Graham. These are **structured** in lattices and include not only user interactions with the system but also the tasks the user performs that is not to be computerized (yet?). This allows me to see how the computer fits into the user's business. The computer system seems to always play a small part of the real business goals of the user. (My area is support of the legal profession.)

So, in my area, I get better information by ensuring I have a context in which the computer interactions and functions are used.

-- LarryWinkler

[At this point in the text, a few minor examples of UserStory-like requirements were removed by their original poster.]

Based on the examples of UserStories I have seen on wiki, no longer will I say that an XP UserStory is "just" a cut-down use case. They sure don't resemble anything I'd call a use case. -- AlistairCockburn

I would say that a UserStory is simply a brief, informal assertion of a requirement. There are no stylistic or perspective constraints on it. As a result, there should be no problem calling performance or security requirements UserStories, as Ron has been telling us. To answer my question above, I am the one who was confused. -- RussellGold

Well, I declare myself officially confused. I can't match the descriptions recently posted here on wiki with the official words from Kent here on wiki and in his book. -- AlistairCockburn

---

When I was introduced to user stories at XpImmersionTwo the first thing I noticed was how closely they mimicked actual *requirements* I've received from customers. Use cases on the other hand have not been terribly *use*ful. They have an artificial feel to them, the *uses* and *extends* thing for instance; I don't believe that customers think that way. -- JohnMerk

The uses *and* extends *relationships are just techniques for organizing the text of the use cases. You can very easily dispense with them - and no, I would not expect customers to think that way. Use cases turn out to be a really useful technique, but it is hard to learn how to do them effectively from the published literature. When Alistair's new book,* WritingEffectiveUseCases *is available, that should change. -- RussellGold*

Can't one look at a UserStories as the description of a goal for a UseCase? Then the acceptance tests written by the customer actually fully describe the

underlying use case. In other words, perhaps both models fit well together. -- DavidVanCouvering

---

Found an interesting comment at http://computer.org/seweb/dynabook/McKinnonComments.htm. TimMackinnon writes:

In our company (Connextra), during our development of the Sidewize product (which did not have a large user base in its initial development), we nominated a team of three people to [make sure TheCustomerSpeaksWithOneVoice]. These people represented Sales, Marketing and Technology. It was their job to solicit stories from people both within the company and from beta users, our software partners, and ensure they made sense and did not contradict one another. They also provided a unified voice for the priority and detail of the stories. This has worked particularly well and may give insight into the new role of business analysts.

This suggests one avenue to pursue when ScalingExtremeProgramming.

In one of the XP books it quotes Alistair Cockburn as saying we can think of Stories as promises for later conversations between developers and customers. Alistair also talks (in Crystal) about the use of "markers" in software development which are things that record what you want to remember and also inspire other thoughts later on. So, it seems to me that an initial UserStory is a marker that reminds the customer and developers what to talk about, and also inspires them in exploring some particular area of business value. A UserStory, then, contains whatever the customer thinks is necessary to jog the memories and inspirations of those who will later explore the story. Expertise in the business area will help the customer decide what are the essential "jogs" to record in the initial UserStory marker. -- Anthony Lauder

Well stated, Anthony. You certainly have captured my view there (and presumably your own). -- Alistair

---

I am developing a software system for my company. The system will allow the users to store, retrieve, manipulate and view the information about the TELECOM Companies of the world. I was introduced to XP by Prof. Johnson, UIUC in Fall 2000. Then I did a project using Smalltalk and SQL Server.

This time I am using Oracle 8i, Developer 6i. I have decided to follow XP methodology.

In my opinion a User Story should look like this:

> The Software System should store the Name, Profile and Address of the Telecom Company

I think that it's quite difficult to get a large number of User Stories at once. What happens is that you get them from time to time.

I have decided to get the User Stories, Analyze them, Design and Document, then Implement and test.

Proceeding in this fashion I will have iteration reports, which will have complete and updated information about the system and its features as the project progresses.

This is tough, but today when I saw a 100-page Software Requirement Specifications for another on-going project and learned that not even a single line of code is in place to date, I was convinced that XP is many times better.

As per my understanding, the motto is *Get the User Stories, Analyze/Elaborate them, then Design and Code as per the chosen standards, then perform Unit, Integration and Acceptance Testing.*

-- Rahul Sharma (rsharma@npi.stpn.soft.net)

Consider this: if you tell what you have to store, instead of telling why/who needs the info, you introduce problems. For example: is the address used for phone marketing, letters or identification? Is there more than one contact possible? IS the address about a billing address or about a legal representative? The story should be "In order to identify the company stored, a postal address of the company's headquarters will be stored together with the company name." -- Bernd Eckenfels

---

I have a project in my mind, an estrategy game, which is becoming quite big. I was wondering: is there an XP way to store ideas which I consider really important for a far away future, but which cannot be estimated because they involve much functionality? I'm thinking of a hierarchy of user stories: I have this idea now, and I don't want to split it right away because I have no idea how and it will take time that should be used somewhere else. If I work with this as user stories I would end up with parent-child user stories, in which case the parent doesn't have an estimation on itself, but the sum of all its found child user stories. An example could be: "The game should be split in three editions: beginner, advanced, expert. This split main reason is to have a way of getting to the expert (full featured) game from easier editions.". Is that OK, or are there examples of how this should be or shouldn't be done? Also, how about really particular details I think are really cool, but again, they depend very much on this game workings like "The spy should get random information on the specified topic of research from the enemy city.", this when I don't even have a city designed. Here, I could come up with several of this ideas, and I would rather like to store them somewhere, should I use something different that user stories or should I try to build myself a user stories relationship system (like a 3 level abstraction level, where level 2 is actually the one that gets estimated (someday)? Sorry for any incoherence; I come from a Spanish country.

I wonder.... how do you complete the project? I understand that user stories become a check-off list for system completion, however, it sounds like there is a lot of room for the customer to change his/her original intent as the new system is implemented. This is good (short-term) for the customer and if I were in the business of selling services, good for business but I am in an IT department with limited resources and a need to get one project done and start on the next one. Any thoughts?

-- Bob Cross

The ongoing, iterative nature of XP allows for systems to evolve over time. There is always a 'delivery date' which will need to be met (or, realistically adjusted as the iterations go). What is crucial in XP is feedback from the customer on a very short time basis. Stories may change, but make sure that those stories change in manageable ways. (For example; don't leave feedback until the system is 'done'; make sure there is feedback on every iteration. Waiting until the customer gives feedback during BigBangTesting is the largest source of cost overrun, missed deadlines, and frustrated customers and employees, in my opinion. -- ChadThompson

---

I am a customer who wants to establish priorities among the user stories. Why are they regarded simply as a heap? Why can't I specify the priority and order in which I need capabilities?

Also, can I include "ease of extension" or "designed to adapt quickly if I change my mind" as components of a user story?

-- Vince Khan

User Stories are not regarded as a heap. They are prioritized. -- BrentNewhall

---

Would a UserStory be applicable to capturing a problem rather than a prospective feature or solution? For instance, would something like "Developers misinterpret requirements" be a UserStory when codifying the problem domain? I've looked at UserAntiStory, but that again seems to only be used in the context of the solution, like "User X should not be able to push button Y". Before there is ever a button Y, a user X, or a system that solves the user's problem, is there a way to describe the problems that plague an organization or process such that they can be discussed and their impact weighed? -- JamesWhite

See AnalysingTheProblemDomain

What about BusinessStories? I can't help feeling that -- since, according to recent studies, investment in IT bears little relation to improvements in business performance (which, by the way, is why the industry is in a major recession, folks) -- we are just tinkering around the edges of a much bigger problem ("building the right thing"). You simply cannot assume that just because the requirements can change, business value will be delivered. You could do 10,000 iterations and still deliver nothing of value if the customer hasn't clear objectives and the business architecture to back them up (and a little bit of luck). Agile software development does not necessarily translate into an agile business!

-- JasonGorman

If the customer has no clear objectives, then it doesn't matter what software development methodology you use; you still won't deliver anything of value. -- BrentNewhall

I'm in the other end of a problem - a customer trying to figure out what I need. User stories doesn't seem to offer much help (apart from the iterations, perhaps). Any comments?

-- Jan Frelin

The name "user story" seems to be very confusing. When someone hears it, they'll immediately think about "use cases", "usage scenarios", or something like that. But judging from these pages, they are nothing of the sort. Where does the name come from?

1. User story examples here do not look like anything I'd call a "story"
2. They are not always written from users perspective
3. Also nonfunctional requirements are handled as user stories
4. Documentation required by the customer is handled as user stories
5. Basically everything required by the customer is handled as user stories

So, "user stories" are actually "customer requirements"?

more like "tasks"

I'm working with my development team to come up with an Agile Development methodology that works for our group. Right now, I'm brainstorming with our UI expert on how to integrate a coherent UI design into the process. Another aspect of our particular set of needs (which I suspect is very common) is that the user writing the story represents business needs, but might not dovetail exactly with the wants and needs of the actual users (the users are third party clients). For this reason, we're looking at some up front UI design and some research with the actual users to determine how well the stories fit with what they do.

Anyway, one suggestion our UI designer came up with is adding another aspect to our stories. We're looking at adding a "why?" to each story card. It seems to me that this is a great idea and the benefits are threefold. First, in the case of a publicly facing page, our UI designer has some background to possibly suggest modifications of our stories to better resolve the needs of our clients with our business needs. Second, in all cases it allows both the UI designer and the developers to suggest modified stories which might better fit the "why?" aspect of the story. Third, it aids in prioritization of (and possibly even elimination of) stories. Has anyone tried anything like this? Does it work?

About *I'm working with my development team to come up with an Agile Development methodology that works for our group. Right now, I'm brainstorming with our UI expert on how to integrate a coherent UI design into the process.*: I am interested in your work. Who are you? What are your results so far?

I've created a new page called UsabilityAndAgileMethodologies to go into a bit more detail. -- CaseyCady

---

This user story doesn't specify anything about HOW to implement what is needed, but is very specific about what is needed. -- DonaldMcLean

In the event log example we usually have very specific requirements that must be met. Just saying we need an event log isn't terribly useful. We need to know things like what is the mutex behaviour, is it ISR safe, what is the max amount of memory that can be used, what's the on disk limit, etc. Is each of these a separate story or part of the same story?

That's an interesting paradigm, it's a practical method to developing. But my questions are about the maintenance of that system when the team work has been dissolved. I mean, are the UserStories enough to understand the code, and the programming philosophy?; could other work team take the administration or maintenance of that system? HOW? -- LucasZallio

---

What happened to user stories? They seem to be morphing into requirements and use cases. I read now where they have turn into an acceptance test which really requires an immense amount of detail. For example, i can't just have a short story like "user can select shipping options" and then talk about what that means later. For a test to be generated this would have to be fully designed into a form that is much larger than any index card and would be more complete than the heaviest use case.

I thought I had this all figured out; now you all have me confused.

I've been using UserStories as concrete examples of UseCases being invoked. Most of the requirements negotiation tends to fall not in the general characteristics of a use case but in the detail revealed in a user story (The who, what, why of a use case usually is pretty much determined by the business process).

What I find disturbing in this discussion is the suggestion that there is some deep gulf between them. I learned a long time ago that the most useful use case is:

- a unit of interaction; you can imagine the user coming up to the machine, invoking a use case, and walking away happy.
- a unit of work; it can be delivered in one iteration and cost-estimated with some confidence.
- a unit of test.
- a unit of documentation.
- one section in the user guide.
- one help frame.

This sounds pretty close to the specs for a user story. -- MarcThibault

---

By Mohammed Qattan:

As a matter of fact, user stories are something that I cannot really describe how they made my life easier. Here is what I did. My company is PMI standards oriented, so they still believe in plans, schedules and dead lines. And in order to get the contract, I had to submit a full detailed plan. This

cannot happen, this is why XP is for. Not to give deadlines that you really don't know about well, in the plans I used to give about 2-3 weeks of an item called "User Stories and Interface" it was a good sell for them (both my CEO and the Customer). What I did is that I used to meet with the customer, and I did write the user stories. And I changed the Metaphor concept a bit for each user story I had a page as a prototype and when user stories are complete into a scene I rushed back to the customer to get their feedback on the prototype and from the user stories I had tasks. I am a project manager, and I am not away from the code, so I meet with the developers, and we estimate on the user stories, and I make sure to down size the tasks to a day for each now I have a plan according to the PMI standards and in fact I am doing the iteration according to the XP and I have a milestone (a release) and I had the developers estimate and each developer can come to my office to get his/her user story, and with the user story there is an attached document representing the page from the prototype. All what I know is the pile of user stories is vanishing in the speed of light. I really love XP and I really owe a big thank you for the great people who made my life happy - the people who invented this XP.

Thank you all

I just read an Interesting Blog that describes the InvestModelForUserStories. -- DanPupek

---

Does this sound right?

- UserStory
  - A feature **(requirement)** or an example **(task)** of a feature to be implemented.
  - There may be several related stories that identify the purpose(s) or feature(s) needed.

-- WyattMatthews

"i can't just have a short story like "user can select shipping options" and then talk about what that means later. For a test to be generated this would have to be fully designed into a form that is much larger than any index card and would be more complete than the heaviest use case."

In my experience, that's too vague. You could just as easily say "user can buy things" and call that a user story that represents all possible features and convolutions of e-commerce. But it wouldn't be helpful to either the customer or the developers.

How about this: "When reviewing their order, user must select a shipping option. Their order total will immediately be updated to reflect the cost of shipping. Options include a flat rate set by the administrator, or a rate looked up via one of two external services based upon total product weight and destination."

That's still small, yet...

- It gives enough info for the planner to have some idea of when it can be implemented (must be able to review an order).
- It gives enough info for the developer to get some idea of the interface (most likely a dropdown box or radio buttons on order review and a set of admin CRUD forms for the flat rate, API information forms, and product weight and order destination fields in the appropriate places) and the fact that remote service calls will be needed; all of which are necessary information to estimate.
- It indicates that shipping cost will be based upon the total order rather than a per- product basis (something that an experienced developer should recognize needs to be confirmed with the customer to determine whether or not the customer's real needs are being met or if more complexity is needed).
- It shows testability (option must be selected, total must be immediately updated).
- It brings to mind details that need to be discussed or researched (What happens if the external service doesn't respond? What info other than weight and destination needed by the external services?).

So, if I understand correctly, the user story doesn't have to specify everything, but it needs to be able to serve as a basis for planning, estimation, and discussion, triggering the right questions, rather than just a bullet point on a marketing brochure that could mean anything.

Does that sound right?

See UsersAreSmarterThanProgrammers

CategoryRequirements

Last edit January 8, 2014, See github about remodeling.