User Stories Don't Help Users: Introducing Persona Stories

William Hudson
Syntagm | whudson@acm.org

User stories are one of the most popular alternatives to traditional user requirement specifications. But despite their promising name, user stories are not about—and don't necessarily help—users at all. In most cases, user stories are written about *roles* that users adopt and take no account of the needs and behaviors of real users. Were that not indictment enough, user stories suffer from demonstrable flaws in structure and are often written by the wrong people at the wrong time.

Here, I examine the background of user stories in their current form, highlight their failings, and propose a more appropriate alternative for the development of interactive systems: *persona stories.*

User stories—as brief scenarios of use written on small cards—were used in the first extreme programming project, C3 at Chrysler, in 1996 and described by Kent Beck in his book *eXtreme Programming Explained* in 1999. The idea of describing requirements as stories of use has a long history, much of it in human-computer interaction. In *Scenario-Based Design,* Jack Carroll traces the thinking about scenarios in design back to a 1959 article by C.W. Mills. But it was Ivar Jacobson who, in a 1987 paper on use cases, proposed describing the requirements of industrial systems from this scenario perspective.

Like use cases, early user stories did not have a specific form. The lack of form and consistent content was seen as problematic by some in the Agile community and was addressed by a team at Connextra in 2001. They proposed that user stories should take the form:

As a <role>, I want <goal/desire> [so that <benefit>]

Elements in angled brackets (such as role) are to be supplied and elements in square brackets are optional. This style of user story was popularized by Mike Cohn in *User Stories Applied* in 2004.

User Stories

While there is frequent debate on the value of user stories in general and of the Connextra form in particular, both give rise to substantial issues when viewed from an HCI perspective, which I will address in some detail. In particular, the focus on roles is inappropriate for many systems, and the narrative structure as outlined here is unsuitable for a number of reasons.

The role of roles. A role is a systemizing concept that has widespread use in business and industry. Its primary function is to describe an individual's activities and responsibilities. In software development, Jacobson made use of roles in use cases (here, quoting Kristen Nygaard from a 1986 lecture): "A role is defined through a specified task or a group of closely related tasks, which are performed by persons during the development and/or operation of a system."

But roles are not as simple as they might first appear. In *Understanding Organizations,* Charles Handy devotes an entire chapter to roles and interactions, describing issues such as role ambiguity, role incompatibility, role conflict, role overload, role underload, role strain, and many others. Role ambiguity is particularly relevant to systems design since it describes the lack of clarity that an individual, his or her colleagues, and the organization itself may have about roles. Those of us conducting user research have usually had first-hand experience of this.

Daniel Ilgen and John Hollenbeck describe the issue explicitly: "The simplicity of the role definition as a set of expected behaviors masks the complexity and ambiguity that is discovered as one probes more deeply into the underlying assumptions behind the definition."

Roles are only a tiny part of the picture when it comes to the needs and behaviors of users. They tell us, approximately, what *kind* of activities a user may undertake, but they say nothing about *how* and *when* tasks are performed. So, for example, we may identify an accounting or bookkeeping role, but we would need to do user research to discover that some tasks are performed many times a day while others are relatively rare. The implications of these differences are significant for interactive systems design.

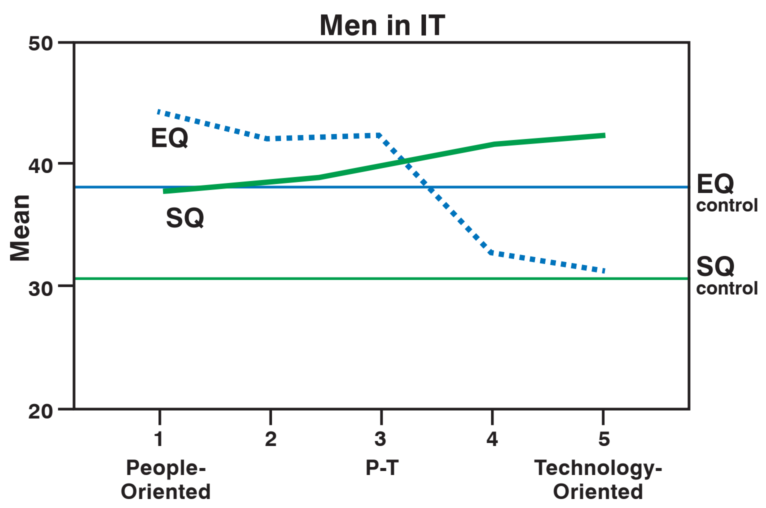
Larry Constantine and Lucy Lockwood have acknowledged the need for research and role modeling in their *usage-centered design* process. Constantine and Lockwood's user roles are more descriptive, including characteristic detail that results in role names such as SingleTicketPurchaser. However, since this role is somewhat tautologous—anyone purchasing a single ticket is by definition a SingleTicketPurchaser—it is hard to see where the role ends and the use case or user story begins. In this instance, applying the Connextra/Cohn form of user story yields: "As a customer I want to purchase a single ticket so that I can travel."

But herein lies a significant problem. Roles are not particularly useful in consumer-oriented systems. We end up with stories with roles like "customer," "visitor," or "subscriber," which tell nothing about what we call the *contexts of use*. These contexts cover a range of issues beyond simple role descriptions. And for many systems, these are things that we should research at an early stage.

What we need is an approach to requirements that focuses on those differences in the behaviors and needs of users that will require us to provide substantially different forms of interaction. But first, some more challenges with user stories.

Wrong people. While user stories are the child of extreme programming (XP), they have since been adopted by other Agile approaches, most notably Scrum. In XP and Scrum, user stories should be written by the business, product owner, customer team, or user representative, depending on where and when you look (it changed between the first and second editions of *Extreme Programming Explained*, for example). But as I argued during the early excitement about user stories, no one person (or even small team) can dictate or represent the needs of users [1]. This is particularly true of users on the Agile team who are almost always chosen for the wrong reasons—after all, you would not select someone who wasn't particularly good at their job to act in this capacity, but they may actually be much more representative of real users.

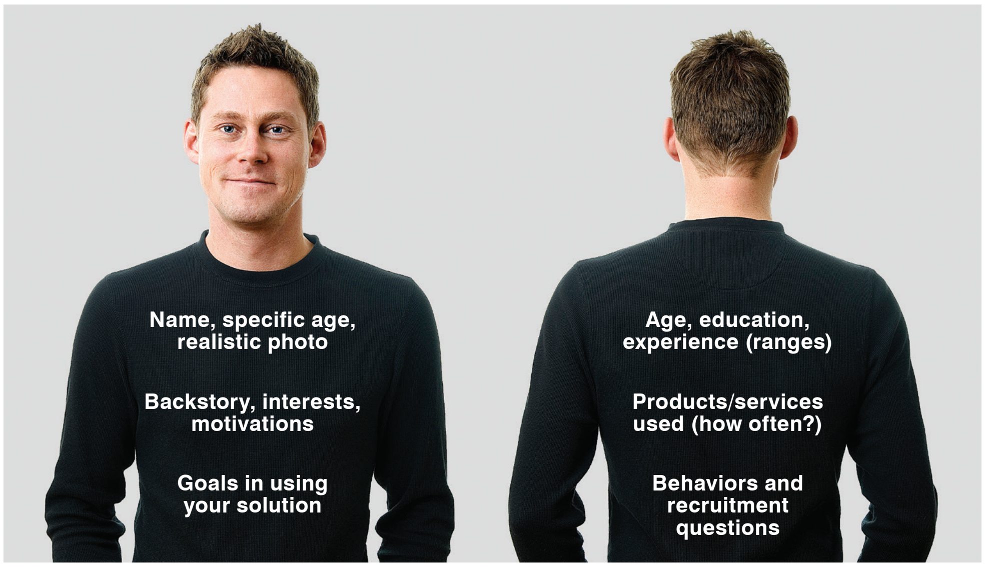
There is a further complication that needs to be mentioned. From my own research, using methods normally applied to the study of autism, I found that technology-focused men working in IT had significantly reduced empathy. Figure 1 shows the result of the study for men (EQ is empathizing quotient, SQ is systemizing quotient [2]). These results are indicative of something that had been observed decades earlier by Nathan Ensmenger in his book *The Computer Boys Take Over:* "Programmers dislike activities involving close personal interaction. They prefer to work with things rather than people."



▸ Figure 1. Technology-oriented men showed a marked reduction in empathy (EQ and SQ controls are for the average population). (*n*=156 men, *n*=285 women)

Reduced empathy, however, doesn't just explain the lack of enthusiasm for personal interaction. It also means that technologists, who are very good at building and understanding systems, find it hard to see a problem from a perspective other than their own. And low empathy isn't confined to technologists; in the normal population, empathizing and systemizing skills are unrelated. So we need to take concrete action to ensure that any artifacts describing users' interactions with a system are written by someone who has empathizing skills and a good understanding of user behavior, and in such a way as to promote empathy within the team. (Women, on average, have higher empathizing scores than men. Both men and women who were people-oriented in their job roles scored above population averages for empathy in the study.)

Wrong time. In XP, user stories are meant to feed into the planning game (adapted from "The New New Product Development Game,"


Name, specific age, realistic photo
Backstory, interests, motivations
Goals in using your solution
Age, education, experience (ranges)
Products/services used (how often?)
Behaviors and recruitment questions

▸ Figure 2. Suggested content for a minimal persona (front) and recruiting brief (back).

a frequently cited 1986 article in the *Harvard Business Review* by H. Takeuchi and J. Nonaka). But if a team is going to make estimates and plans based on user stories, they need to at least know what needs to be done and approximately how. This is particularly true of novel systems, where there are no established current practices to be described. That means we need to give some thought to the purpose, shape, and size of the system *before* writing user stories. Unfortunately, many Agile adopters believe that design is a bad thing, even though this is not actually what the *Agile Manifesto* and its "Twelve Principles of Agile Software" say ("big design" up front is bad, but rough design is not). So, not surprisingly, estimates and plans made with premature user stories may not be very reliable.

Structural flaws. I mentioned at the outset that user stories, certainly in their Connextra/Cohn form, are structurally flawed:

• Roles are not a suitable focus of attention for many systems.

• The "As a <role> I want…" form is unnecessarily wordy and repetitive.

• The use of the first person is counterproductive.

Since I have already outlined the case against roles, let me address the two remaining points. Even in small systems there are likely to be scores, if not hundreds, of user stories. To read and write "As a <role> I want…" each and every time is both monotonous and time-consuming. When we look at how people scan and read material, superfluous words at the beginning of sentences are particularly troublesome, since they move the more important content further into the text, thereby making it harder to process.

On the final point listed here, there are two separate reasons for declaring that the use of the first person is unhelpful. The first reason is very familiar to all who have worked in usability and user experience: Many usability issues arise because developers assume the users are similar to themselves. In the majority of cases this is simply untrue, although reduced empathy (as discussed earlier) makes this hard for developers to understand. In fact, a Google search for the phrase "you are not the user" produces more than 470,000 results.

The second reason is that thinking about yourself is not actually as effective in a design context as thinking about others unknown to you. In four studies published in 2011, researchers found that participants were more creative and better able to solve problems when doing it for distant others (people they did not know personally) rather than close others or themselves [3].

Personas and Persona Stories

Given the case against user stories, but the undeniable interest in using something so relatively immediate (compared with traditional requirements or use cases), what alternatives do we have? My belief is that user stories can be adapted to be more user-centered by changing their structure and shifting their focus from roles to *minimal collaborative personas*.

Minimal collaborative personas. Although the term *persona* and its related concepts in English are quite old (originating from the Latin for *person*), Alan Cooper is the first to have applied the term to characterize users of an interactive system during design in his 1999 book, *The Inmates Are Running the Asylum*. Cooper's goal with personas was to give designers focus and to make users seem more like real people. Personas have since become very popular in the fields of usability and user experience, but unfortunately the original purpose has become somewhat blurred. It is not uncommon to hear about UX teams spending many weeks or even months developing personas; I have seen individual personas of six pages in length. These are not likely to help give developers focus. Nor is the common practice of the UX team researching and writing personas in isolation, then providing them as a fait accompli to developers likely to make the personas seem like real people.

To be Agile we need *minimal, collaborative personas:*

Minimal. Each primary persona requires a different user interface. The persona descriptions need to explain what behaviors and needs each persona has that make a different interface necessary. There should be a small amount of backstory (character description) and motivation so that anyone presented with a scenario for our system can read the persona and come away thinking, "Ah, that's why we need to do it that way." Ideally, each persona should occupy one side of a sheet of paper. The back of the same sheet should provide practical information along the lines of, "How would I know this persona if I saw one?" used to recruit participants for research and usability evaluation. Specific personal details should be provided to the extent they help the persona seem like a real person. More general demographics, however, should always go on the back (they are not part of the persona proper; see Figure 2).

Collaborative. Agile is collaborative at heart. Agile teams make hundreds of detailed design and planning decisions every day. If the core team does not appreciate or understand the user-experience aspects of the project, it is likely to fail. (The popular alternative of specifying the entire user experience in advance, before any code is written or any learning has taken place, is actually a waterfall approach, with all of the inherent dangers and drawbacks that waterfall methodology had in the 1980s and 1990s.) The core team should be involved in the user research required for personas—at least as observers—and must be actively involved in the development of personas. There is substantial evidence that involving people in the decision-making process is essential if they are going to feel any sense of ownership of the resulting personas.

One of the primary goals of personas is to create empathy and motivation for the team. Personas do this by allowing us to connect emotionally with other individuals rather than abstract collections such as "users" or "SingleTicketPurchasers" [4,5].

Persona stories. Persona stories differ from user stories in several important respects. Persona stories are written:

- …about personas, not roles. Where it is useful or important to refer to roles, we simply qualify the persona name. Let's say we decide to characterize a warehouse returns operative as "Jack." In many cases we probably would not need to describe his role, since it would be obvious from the story. So a persona story would be as simple as: "Jack processes a return." Note that as a persona, it's Jack's behaviors and needs that are important. It may be there are other Jacks in our system, probably working in a cold and dusty warehouse and without particularly good typing skills (these points would be part of the Jack persona). So Jacks might also process pick lists or label packages for dispatch.
- …in the third person, that is, about the persona. So the form is: <persona[:role]> <performs a task>[so that<unobvious goal>]. The persona:role element is based on the Unified Modeling Language notation name:class. An alternative would be simply to place the role in parentheses when it is needed. The optional "so that" clause should be provided only if the goal of the task is not obvious. "Julie adds an item to the shopping basket" does not really need an explanation.
- …by user experience specialists in collaboration with business analysts and/or members of the core team. They are subsequently elaborated into scenarios and visual designs (again in collaboration with the core team) one or two project cycles ahead of their implementation [6]. This avoids the Agile/Waterfall conflict that is brought about by up-front UX design.
- …after user research and rough design. The research is required to discover the needs and behaviors of users of interest to a project. Rough design defines the scope and shape of our venture. For example, if we are selling houses, we may decide that a comparative shortlist feature is more appropriate than a shopping basket and would write our persona stories accordingly.

One of the main benefits of persona stories, when produced as outlined here, is they and their resulting scenarios and visual designs will be *descriptive* rather than *prescriptive*. That is to say they describe interactions we have good reason to believe will work (because we have done research and evaluation with real users) rather than just prescribing what users must do. It is a subtle but extremely important difference.

ENDNOTES:

1. Hudson, W. Adopting user-centered design within an Agile process: A conversation. *Cutter IT Journal 16*, 10 (2003), 5–12; http://www.syntagm.co.uk/design/articles.htm

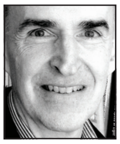
2. Hudson, W. Reduced empathizing skills increase challenges for user-centered design. *Proc. CHI 2009 and BCS HCI 2009 Conferences.* 2009; http://www.syntagm.co.uk/design/articles.htm

3. Polman, E. and Emich, K.J. Decisions for others are more creative than decisions for the self. *Personality and Social Psychology Bulletin 37,* 4 (2011), 492.

4. Nordgren, L.F. and McDonnell, M.H.M. The scope-severity paradox. *Social Psychological and Personality Science 2,* 1 (2011), 97–102.

5. Sears, D. The person-positivity bias. Journal of *Personality and Social Psychology 44*, 2 (1983), 233–250.

6. Hudson, W. User requirements in the 21st century. *Agile Record.* 2012. 12–16; http://www.syntagm.co.uk/design/articles.htm



ABOUT THE AUTHOR
William Hudson has been building interactive systems for more than 40 years, focusing on user-centered design for the past 20. He has written more than 30 publications and is the creator of the Guerrilla UCD series of webinars (www.guerrillaucd.com). He is also the founder and principal of Syntagm Ltd, a small Oxfordshire, U.K. consultancy specializing in user-centered design and training.

DOI: 10.1145/2517668