Agile Record

The Magazine for Agile Developers and Agile Testers

Requirements in Agile Projects

August 2012

www.agilerecord.com free digital version made in Germany ISSN 2191-1320 issue 11

User Requirements in the 21st Century

by William Hudson

The Role of IT Has Changed – Requirements Gathering Hasn't

In the early 1990's, few could have predicted the explosive growth of the World Wide Web and the invasion of information technology into almost every area of daily life. The pace of change has accelerated even further in the past few years with the increasing popularity of mobile devices, such as tablets and smartphones, swelling the ranks of internet users to a projected 2.7 billion by the year 2015, about one third of the world population.

This means that information technology is being used by a vastly more diverse audience than it was 20 years ago, yet the methods we use to specify and build interactive systems are firmly rooted in the 20th century. For example, although the Agile Manifesto – used as the underlying philosophy for many current development methods – was ratified in 2001, it was based on the industrial design and development methods of the Lockheed Skunk Works™ from the 1940s [1]. In fact there are few, if any, mainstream software development methods for gathering requirements or building solutions designed around users, usability or user experience.

The absence of focus on users becomes particularly embarrassing when considering the place of technology in the new millennium. Twenty years ago, interactive systems were used primarily by people who were employed for the purpose. While personal computers were making important inroads into the commercial computing world, they were still relatively expensive and the province of a privileged few. If interactive systems were hard to use, staff would be trained or struggle on as best they could. The role of IT has changed substantially in that time, with the vast majority of users now expecting (and demanding) self-explanatory systems that are both effective and enjoyable to use.

To bridge the gap between traditional software development (including Agile methods) and the needs of present-day users, we have introduced a corrective role that we call a usability or user-experience specialist. Regrettably, this role is usually peripheral to the software development process itself and is often limited to usability evaluation. The net result is that we have added another quality hurdle, but have made no real changes to the development process itself. Both of these approaches were expressly admonished by the legendary quality consultant J. Edwards Deming (the man credited with teaching the Japanese how to build quality cars – quoted here from his *14 Points for Management [2]*):

3. Cease dependence on inspection to achieve quality. Eliminate the need for inspection on a mass basis by building quality into the product in the first place.

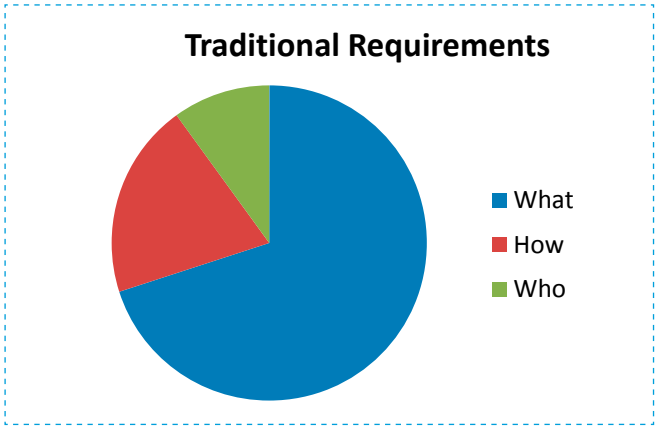
9. Break down barriers between departments. People in research, design, sales, and production must work as a team, to foresee problems of production and in use that may be encountered with the product or service.

Although Deming was addressing quality in an industrial setting, his Points for Management apply equally well to software development. This is particularly true of Agile with its narrow focus on the creation of working code rather than the overall effectiveness or suitability of the result.

Industrial Focus

As mentioned above, Agile itself has an industrial heritage, as do most of the other tools and techniques in use in mainstream software development. Ivar Jacobson's development of use cases, for example, stems from his experience of switching systems in telecommunications. One of his earliest papers on the subject refers directly to this focus – *Object Oriented Development in an Industrial Environment [3].* Throughout the 20th century, most authors in the field of software engineering wrote as though interactive systems did not exist. The design and development of effective interactive systems was left as an exercise for the reader or dismissed as a separate area of study – poorly served until the phenomenal interest in the World Wide Web in the mid-1990s. Even relatively recent texts on software development make only passing reference to usability.

The industrial heritage of software development is reflected in our approach to requirements. Jacobson, in his discussion of use cases, is not alone in treating the system under consideration as a "black box" – something whose inner workings are invisible to an external observer. This is why, given the large number of questions that we could reasonably ask about a new development project, requirements are traditionally seen as specifying **what** is required; **how** is to be avoided at all costs. There are many good reasons for this – even when developing interactive systems, we do not want to indulge in premature design. But this approach only works if we do not much care about what actually happens inside the black box, as long as it meets all of the requirements. This is not practical with interactive systems: we cannot specify the interaction requirements in enough detail to be certain that all conforming implementations will be equally effective.



To all intents and purposes, our black box is translucent. Users cannot see the implementation itself, but timing, terminology, organization and demands on users' skills and judgement are fully exposed – resulting in something more akin to a "grey box". And the solution to building more effective interactive systems does not lie with superficial improvements to the user interface. It demands significant (but not earth-shattering) changes to the way we collect and evolve user requirements. The overall approach can be thought of as user-centered Agile development, although the terms Agile UCD and Agile UX are equally applicable.

User-Centered Agile Development

For interactive systems in the 21st century, we must do more to understand how our users behave and what they need from our interactive solutions. Furthermore, these behaviors and needs must inform the design process without violating agile constraints (for example, no big design up front). To do this requires a few adjustments to a typical development process:

1. Integration of usability and user experience expertise with the development team
2. An appropriate amount of user research up front
3. The distillation of the user research to a suitable form for design (personas)
4. Parallel streams for interaction design and development

The following sections address each of these adjustments in turn.

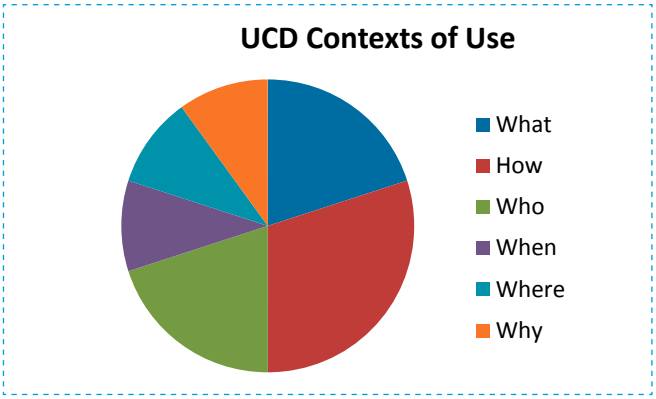
1. Integration of usability and user experience expertise

In his book *Agile Software Requirements [4]* Dean Leffingwell shows an "ideal" agile team with a UI (user interface) resource as external and includes user experience designers in his discussion of "other supporting roles". This is not an unusual arrangement for many organizations, but it does explain why many interactive systems suffer from poor usability and user experience. The key expertise for researching, designing, advising and assessing the user experience is removed from the process of building the solution. It many ways this is simply a reflection of the industrial focus that dominated software development in the 20th century – the user interface and user experience in general are separate topics outside a team's core activities. Yet Leffingwell discusses at length the pitfalls of "functional silos" (the separation of resources by function); he doesn't appear to consider the user interface and user experience important enough to include them in the "ideal" agile team. This again reinforces the industrial background of current methods and Agile's focus on working code over and above solutions that are effective for our users.

Let's be clear about the Skunk Work's (and Agile's) working philosophy: close cooperation should take the place of extensive documentation. If we are building systems that are predominately interactive, we need to build the required expertise into our teams – not call on it occasionally as an external resource. The usability and user experience expertise:

- drives the user research process,
- develops personas and other user-requirement-focussed artifacts,
- evaluates the usability of product increments (or prototypes), and
- advises and supports developers in usability and user experience issues.

This approach gives the team a real chance of understanding their users and addressing usability issues, while there is time and resource available to make adjustments.



2. User research

Detailed user requirements cannot be elicited in the same way as business requirements, rules and constraints. Instead we must investigate and understand what the international standard on human-centered design [5] refers to as the contexts of use. This

need not be a long or onerous process, particularly in a well-established problem domain, such as e-commerce. However, it does require knowledge of research methods and some training in human-computer interaction.

One of the main reasons for this is that just talking to users about their requirements is known to be an ineffective way of establishing how an interactive solution should be designed. This is particularly true in novel problem domains – users are not familiar with the possibilities and are often not aware that their own needs may differ significantly from other users' or what the design team has in store for them. Instead, the preferred methods of user research usually involve a mixture of observation (to understand working practices in some detail) and interviews – see, for example, Karen Holtzblatt and colleagues' work on contextual design [6, 7].

The user research process can also be used to familiarize the development team with real users and their needs. This is an important part of collaborative development since otherwise team members can have real difficulty in understanding that users are often not as comfortable with technology as they are and may not appreciate the need for a user-centered approach [8]. In addition, affinity diagramming workshops (discussed in [6, 7]) can be used to immerse the team and other stakeholders in the domain of real users.

3. Distillation of user research

A large body of data often results from even a small amount of user research. Since the functional requirements of the solution are still specified in use cases or user stories, user research needs to be reduced to a more compact and immediate form. For these purposes, personas are ideal although quite widely misunderstood. The focus of personas should be on user behaviors and needs, not demographics.

Most interactive solutions will have a relatively small number of personas with some defined as primary and others as secondary. Each primary persona requires a separate user interface because of their particular contexts of use. The persona itself is a description of a "typical" user with their needs, behaviors and motivations as well as problem areas and challenges. We present a persona as if they were a real person – this often makes the development team uncomfortable at first, but there are good psychological reasons for doing so, since this gives us a better chance of understanding and treating users as real people [9, 10].

4. Parallel streams for interaction design and development

Since the agile philosophy has us avoiding "big design up front" (for good reason), we need to weave detailed interaction design into the mainstream development process. We do this by involving usability and user experience expertise in the creation of (light-weight) use cases or user stories and then expanding them in the cycle before the development team is due to implement them. The process is summarized in Figure 1. The UX stream is involved in both specifying the interaction design, but also in evaluating the product increments delivered in the previous cycle.

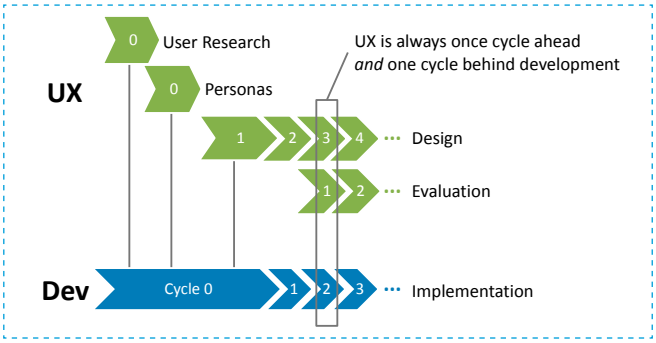


Figure 1, Parallel Streams Approach

Summary

Our development processes need to be aligned with the needs of the 21st century in light of the experience of the past decade. Interactive solutions will be used by an increasingly large and diverse population; usability and user experience cannot remain as optional or occasionally-invoked resources on the periphery of development teams. In this article, we have looked at some of the vestiges of the "industrial" approach to software development and the remedial steps needed to make agile processes more user-centered.

1. Hafen, G. *The Skunk Works: Agile Development for 60 Years.* SSTC 2005 Proceedings, 2005. Available from: http://www.sstc-online.org/proceedings/2005/PDFFiles/GLH930.pdf.

2. Deming, W.E., *Out of the Crisis*2000: The MIT Press.

3. Jacobson, I. *Object Oriented Development in an Industrial Environment.* 1987. New York, NY: ACM.

4. Leffingwell, D., *Agile Software Requirements*2011: Addison-Wesley.

5. DIS, I., *9241-210: 2008. Ergonomics of human system interaction-Part 210: Human-centred design for interactive systems (formerly known as 13407).* International Organization for Standardization (ISO). Switzerland, 2010.

6. Beyer, H. and K. Holtzblatt, *Contextual Design: Defining Customer-Centered Systems*1998, San Francisco, CA: Morgan Kaufmann.

7. Holtzblatt, K., J. Wendell, and S. Wood, *Rapid contextual design: a how-to guide to key techniques for user-centered design*2005: Morgan Kaufmann.

8. Hudson, W. *Reduced empathizing skills increase challenges for user-centered design. in CHI 2009 and BCS HCI 2009 Conferences*. 2009. Boston, MA and Cambridge, England: ACM.

9. Nordgren, L.F. and M.H.M. McDonnell, *The Scope-Severity Paradox*. Social Psychological and Personality Science, 2011. 2(1): p. 97-102.

10. Sears, D., *The Person-Positivity Bias*. Journal of Personality and Social Psychology, 1983. 44(2): p. 233–50.

> About the author

William Hudson[1] *is a User Experience Strategist who consults, writes and teaches in the fields of user-centered design, user experience and usability. He has over 40 years' experience in the development of interactive systems, initially with a background in software engineering. William was the product and user interface designer for the Emmy-award-winning "boujou"; now an indispensable tool in major film studios. He has specialized in interaction design and human-computer interaction since the late 1980's. William has written and taught courses that have been presented to hundreds of software and web developers, designers and managers in the UK, North America and Europe. He is the founder and principal consultant of Syntagm, a consultancy specializing in the design of interactive systems established in 1985. William has presented papers, talks and tutorials at international conferences including CHI, UPA and Nielsen Norman Group Usability Week. He is the author of over 30 articles and papers*[2]*, including the Interaction Design Encyclopedia entry on card sorting*[3] *and the Guerrilla UCD*[4] *series of webinars introducing project teams to usability, user experience and user-centered design. William is also a contributor to the Rational Unified Process and to Addison-Wesley's Object Modelling and User Interface Design.*

He is an Adjunct Professor at Hult International Business School, Courses Co-Chair for the CHI 2013 conference[5] *in Paris and an organizer of the UCD 2012 conference*[6] *in London.*

1 http://www.syntagm.co.uk/design/whudson.htm

2 http://www.syntagm.co.uk/design/articles

3 http://www.interaction-design.org/encyclopedia/card_sorting.html

4 http://guerrillaucd.com/

5 http://ch2013.acm.org/

6 http://ucd2012.org/