User Centred Design in Agile Application Development

Marc McNeill PhD.
ThoughtWorks Ltd.
9th Floor Berkshire House
168-173 High Holborn
London, WC1V 7AA

Agile methods are becoming increasingly common in application design, with their collaborative customer focus and iterative, test driven approach. They share many common principles, yet it is rare for Agile methods to incorporate user centred design. This paper argues that by incorporating user-centred design (and in particular using low fidelity prototyping as an iterative model for the application rather than time consuming code) better applications can be developed, delivering business benefit with a focus upon the end user and their experience.

Keywords: Agile, Methodology, Usability, User-Centred Design, Storyboards, Lo-fi Prototyping, Business Value.

1 Introduction

A common theme of many IT projects is the friction between the different stakeholders involved in the project. They talk different languages; the developers think in terms of code, the business thinks in terms of business value and interface designers think in terms of customer experience. Reconciling these different languages can be a challenge (Figure 1). A criticism often levelled at user interface designers from the technical community is that they do not sufficiently consider the technical implications when creating interfaces. A usable and creatively well designed solution to a user centred problem may be a costly nightmare to code. Conversely, the interface design community often argue that leaving the user interface to the developers may result in a technically elegant solution that is difficult and unappealing to use. Both these positions however ignore the most critical stakeholder in any product design; the end user.

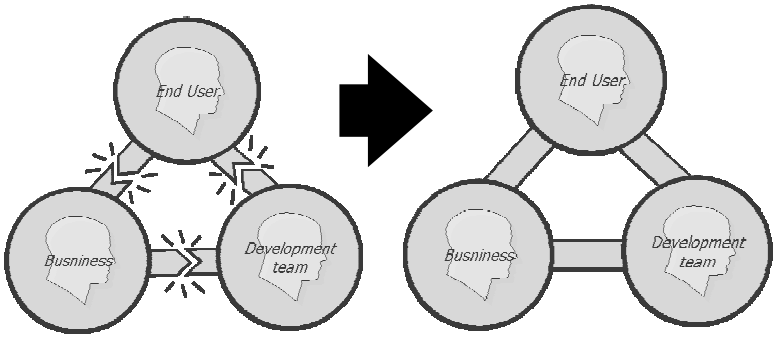

Figure 1 In traditional IT projects there is breakdown in communication, with stakeholders talking different languages. Using visual models provides a common language, ensuring a comprehensible dialogue between the stakeholders.

An inherently usable and technically elegant application cannot be considered a success if it does not satisfy the end users' needs. End users are often left out of the development process. Rather than real end users who will actually use the product, "empowered business users" or "representative" customers may be engaged through workshops of focus groups. Many projects assume that "the customer" consists solely of those stakeholders who sign the cheques. While these people are certainly important to please, and rightly define project scope and feature sets, they are often least representative of the system's end users, and hence not necessarily qualified to design a system that meets those users' needs.

This paper considers the role of the end user as the customer in application development, specifically within an Agile approach delivering real business benefit and mitigating both technical and business risk.

2 Agile methods

Martin (2002) identifies common fears that are present on many projects; the project will produce the wrong product, the product will be of inferior quality, the project will be late, the team will work excessive hours, commitments will be broken and ultimately the project will be a world of pain for all involved. Processes, constraints and deliverables are added to projects to help mitigate these fears; however they often become an end to themselves, making projects even more cumbersome and likely to fail. In an effort to overcome this project overload, a group of industry experts came together as the Agile Alliance and drafted a manifesto for a new way of developing software (Beck et al 2001). Key to the manifesto are;

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Agile methods are lightweight software development processes that employ short iterative cycles, involve users to establish, prioritise and verify requirements and rely on knowledge within a team rather than documentation (Boeham and Turner 2004). They have developed in reaction to traditional software engineering that is seen as overly bureaucratic and slow. Rather than investing a large amount of time in up-front design, rigidly capturing and documenting requirements, Agile methods are adaptive and people orientated. Change is an inevitable and often painful aspect of IT projects; requirements and scope change, development issues impact the process and managing against the plan becomes the goal rather than delivering a solution that everyone will be happy with. Agile however welcomes change allowing the project to adapt to changes as and when they happen.

Similarly rather than shoe-horning the project into a proscribed process, producing documentation and deliverables for process sake, Agile methods assert that no process will ever increase the skill of the development team, so the role of a process is to support the development team in their work (Fowler 2003). One manifestation of this lightweight approach is in analysis; rather than using formal analysis documentation, Agile use stories.

2.1 Anatomy of a story

Stories are short descriptions of what the application should do from the perspective of the user. They are small pieces of functionality that relate to business value. A high level story is very simple; a title and a sentence or two of plain English to describe it. In its most elementary form a story identifies who wants the story, what it needs to do and why it is valuable to have: "As an X, I want to Y, so that Z".

2.1.1 As an…

This defines who the end user will be. The parallel with user centred design (UCD) approaches is obvious here; an understanding and appreciation of the user. Too often the business, developers, and designers try to second guess the user, this often results in complex functionality that may address a perceived need for the business 'super user' but ignore the majority of novice users.

2.1.2 I want to…

Again this is a common step with UCD approaches; an understanding of the user goal and tasks to achieve that goal. Addressing a requirement in terms of the user and their goal focuses development upon what is needed. Consideration is made of the 'how' as well as the 'what'; the requirement to view a customer's financial product holdings will be the same for the branch and the call centre, however the access to, and delivery of, that information will differ.

2.1.3 So that…

The story is expressed solely in business language and should not specify implementation detail. For example state the need to store information, but not the mechanism for achieving it (e.g. database, xml file etc.). The story must have business value. In the crudest analysis, the business value will often be monetary, for example either driving an increase in revenue or a reduction in costs. This business value is then used to prioritise the story. Those with the highest business value coming first, those with the least, last. The development team, by estimating the effort required to complete each story, set a cut-off point for the number of stories that can be addressed in that iteration. The customer may re-prioritise until they are happy that their highest priority stories will make the cut. This mechanism of ascribing business value can be especially challenging for the UCD practitioner where non-tangible factors such as subjective satisfaction are often justification for their inclusion. For example in an on-line banking application, the UCD practitioner may champion the ability for customers to give their individual accounts nicknames.

3 6 Agile and User-centred design

Interaction design shares many of the characteristics of Agile (Table 1), however they are rarely combined and there can be conflict between the two practices (e.g.

Nelson 2002). The greatest source of contention is whether user centred design processes constitute "big up-front design", an anathema to Agile. Agile practitioners argue that traditionally an inordinate effort goes into the design which will undoubtedly change as the project develops. From a user centred design point of view there is an inherent risk in this approach. Focussing upon building discrete functional components to be stitched together as they evolve (rather than considering the application a holistic user experience from the outset) risks delivering a product that is inconsistent and confusing. This almost inevitably results in an inefficient, error prone and ultimately unfulfilling user experience. The underlying code structure may be technically elegant; however the end user cares little for this unless it makes the application more responsive. It may be argued that for the user, the interaction design *is* the application.

Table 1 Similarities between features of Agile methods and UCD.

Principle	**Agile**	**User-centred design**
Customer Focus	All activities are focused on providing tangible business value.	All activities are focused on providing (business) value through ensuring a useful, usable and engaging product. The customer is not defined as the project stakeholders, but the end users as well.
Iterative Development	Early and frequent delivery of working software contributes to project visibility, reduces project risk via regular feedback, fosters continuous improvement and enables early realisation of business benefits. Iterations are typically one or two week cycles.	Develop, test and refine the user interface via regular feedback to the end users. The focus is upon the business risk as well as the technical risk.. Iterations are typically one to two day cycles.
Test-Driven Development	Testing plays an integral role in every phase of the project life cycle.	User testing plays an integral role in the development of the interaction design.
Collaboration	Collaboration between customers, product managers, business analysts, developers, and QA maximizes overall team efficiency.	Collaboration between customers, product managers, business analysts, developers, and QA maximizes overall team efficiency.
Visibility	All stakeholders are provided with maximum visibility into project progress.	All stakeholders are provided with maximum visibility into project progress – the interaction design is the premier communication tool – it is the outward appearance of what the product will do.

Given the similarities between Agile principles and user centred design principles, it is argued that user-centred design need not be seen as big up-front design rather a "fast track" to galvanising project success. The overall agile process that of rapid iterations around a tangible model is in fact very compatible with the user-centred design approach. Traditionally the model used has been a coded prototype. That prototype being reviewed by a business customer at the end of each iteration with a business customer. The suggestion is that even more value can be extracted from this highly iterative process by using a variety of models, not just code alone. For example low fidelity prototyping or storyboarding. The

storyboards become a lightweight, flexible model to help visually articulate the stories that have been written. They are illustrations of how the application might support the user. Rather than waiting a week, and possibly two for an iteration to be completed and showcased to the customer, the customer can review it almost immediately, helping to identify potential problems early on and reducing the need for rework at the end of the code iteration. Thus, combined with Agile, user-centred design has the following advantages:

- Better understanding of the problem.
- Allows rapid testing and validation of story concepts before time consuming coding.
- Provides a clear, sociable visual representation of the project vision.
- Provides usability by stealth.
- Engaging the end user as a customer.
- Improves basis for estimation.
- Mitigates project risk.

3.1 6.1 Better understanding of the problem

Product success can be measured not by the number or complexity of functions and features, rather its ability enable users to accomplish their goals. In placing the goal at the centre of the design it is important to understand the goals from the user's perspective. This is achieved by research and modelling (Cooper and Reimann 2003). Within Agile the objective is not to produce designs, rather a picture of what the users want so as to create effective stories.

Story writing is a collaborative process involving stakeholders from the business, analysis and IT. Agile refers to the business as the customer, however from a UCD perspective this is not the customer as end user. Where end user representation is included it is often by proxy rather than a cross sample of end users themselves. The application will be used by the end users rather than the customers. The business often makes assumptions about end users without understanding their genuine needs. This can be overcome by exposing the team to the end users who will use the product and working environment that it will be used in. Rich qualitative data can be captured using ethnographic and participatory techniques to inform the story writing process. The reality on the ground is often very different from the head office view; grounding solutions in that reality, engaging staff in the process is beneficial in the process of change management.

3.2 Allows rapid testing and validation of story concepts before time consuming coding

Applications are developed in Agile through small, regular incremental iterations, continually testing both the form (i.e. is it delivering on business requirements?) and function (i.e. does the code work). Storyboarding allows the application form to be tested quickly and cheaply, ensuring that the development iteration focuses upon delivering quality code with minimal need for re-work because it does not meet the customer or client expectations.

Using the storyboard as a model of the solution will only work successfully if there is continual involvement from a developer. By engaging the developer in creating the model it will be grounded in technical reality. Indeed this shared involvement is already ingrained in the Agile philosophy with paired

programming. The process here involves the user-centred designer working with the business to identify the stories and commence creating storyboards. The storyboards are refined and developed with the user-centred designer and the developer working alongside each other, cross pollinating ideas and testing assumptions as they proceed.

3.3 Provides a clear, sociable visual representation of the project vision

A shortcoming of many IT projects is a lack of common understanding or vision of the ultimate project goal. This results in a mismatch of expectations and scope changing as the project matures. Different stakeholders have different motivations and expectations of what will be delivered, these different views often result in an underlying mistrust between the different stakeholders. Agile attempts to bring together the user and developer; the developers will be co-located with the client; they are closely involved in the writing of stories and review sessions at the end of every story that is developed. There is still a shortcoming with written stories in that it is difficult to visualise what will actually be delivered. At the end of the iteration working software will be delivered with the developers able to show how it will work, however this often lacks context; working functionality sitting within a placeholder rather than within the broader application framework. This may be acceptable to showcase to the immediate stakeholders, but it is limited in its validity when demonstrating developed features to the broader user community.

Documentation can be ambiguous. Typically in large projects senior stakeholders will sign-off requirements documents having had little involvement in the development of those documents. They are presented with a large number of complex technical documents with a limited time frame to digest and sign them off. There are a number of shortcomings with this approach. Rarely will the document author be available to verbalise the document content. This results in either sign-off being ill-informed with the stakeholder not entirely sure of or misunderstanding the requirements or a painful process with the document generating further documentation addressing and answering the stakeholders concerns. Stories and storyboards help overcome the limitations of documentation.

Storyboarding is a powerful tool to communicate to all stakeholders the project vision. For example, in developing a new account opening application for a bank, storyboards helped refine the proposition. The requirements capture process was significantly shortened; rather than eliciting requirements in a void, a representation of the end goal better enabled stakeholders to sign-off against, or prompt for new requirements that may otherwise have been missed until later when they would become costly change requests. Perhaps the most powerful result of the storyboards was the ability to place them in front of end users and ask them to complete simple processes in a linear fashion. This user testing rapidly demonstrated that the new process would significantly reduce the time to complete the process by more than 50%, providing greater validity to the business case.

3.4 Engaging the end user as a customer

In Agile development the role of the ‘customer’ is central. However the customer is one voice and is generally a business representative who will determine what will have business value and thus be prioritised in the development. However business value for the customer does not always translate into value for the end user. Socialising storyboards with end users as well as business sponsors draws out needs and concerns for the development that the business may not have anticipated or identified. For example presenting storyboards to end users in a retail banking environment demonstrated that whilst they represented the process as the business saw it, the reality in the branches was significantly different. To

role out the application according to the original business requirements would have involved training and behavioural change that had not been budgeted for. By working with end users a process that met both the business and user goals was successfully developed.

3.5 Provides usability by stealth

One shortcoming with Agile is that the developers work from the story cards and showcase the functionality they have developed through each iteration at the iteration showcase. Solving complex technical problems is the primary challenge with little room left for developing the GUI. Furthermore, if functionality is incrementally developed by different developers and left 'on the shelf' to be assembled according to a release schedule there is a risk that the UI will be incoherent and usability design will be reactive rather than proactive. Paying attention to usability when developing the storyboards takes the decision of how the screens should look away from the developers, providing them with a framework to build from. Usability is designed into the application from the outset.

3.6 Improves basis for estimation

Visualisation of what the application will look like and behave helps identify how functionality can best be developed for maximum performance with minimum development effort. What may appear like simple cards, once brought to life can become complex, with functionality needing to be broken down into component story cards. For example, in developing functionality for a retail organisation to search for products sold in specific stores in the first instance appeared to be a minor request. However, once the storyboards had been developed it became clear that different, complex search criteria would be required to deliver the results the business was seeking. The storyboards helped better articulate the requirement and resulted in the requirement spanning two stories (iterations) with a more accurate estimate for the effort to complete them.

3.7 Mitigates project risk

It is a painful reality that not everything we like can be incorporated in the design. In traditional 'waterfall' projects functionality is stripped out as the project progresses and, time frames and budgets get stretched.

In conventional Agile projects, once the stories have been written they are prioritised and handed to the developers to code. Developers work in pairs to ensure continual quality assurance. With user centred design the stories are augmented with storyboards to help inform the vision, test the concepts and confirm the usability.

3.8 Conclusions

Agile methods are gaining acceptance in IT organisations as an efficient and effective means to developing applications that deliver on the business's requirements. This paper has argued that user centred design fits well with Agile. Agile is usually a development-centric philosophy, espousing engagement with the business and using stories and code as the model for communication. User-centred design extends the approach, however rather than using code as the model it uses visualisation to articulate the solution. Through collaborative workshops, creating stories then translating them into storyboards and low-fidelity prototypes enables iterations to be showcased on a daily rather than fortnightly basis. Engaging all stakeholders in the process ensures that when the developers start cutting code the focus will be on ensuring code quality, mitigating the risk of business driven

changes that could not be articulated without having something tangible to evaluate.

4 References

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, G., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D., [2001] *Manifesto for Agile Software Development.* http://Agilemanifesto.org/

Boeham, B. & Turner, R. [2004], *Balancing Agility and Discipline*. Addison-Wesley.

Fowler, M. [2003], *The new methodology*. http://martinfowler.com/articles/newMethodology.html

Martin, R. C. [2002] *Agile Software Development, Principles, Patterns, and Practices*. Prentice Hall.

Nelson, N. [2002] *Extreme Programming vs. Interaction Design*, http://www.fawcette.com/interviews/beck_cooper/default.asp

Cooper, A. & Reimann, R [2003], *About Face 2.0: the essentials of interaction design*. Wiley Publishing.