Reflective Agile Iterative Design

Clint Heyer
ABB Strategic R&D for Oil, Gas and Petrochemicals
Oslo, Norway.
clint.heyer@no.abb.com

Margot Brereton
Queensland University of Technology
Brisbane, Australia.
m.brereton@qut.edu.au

ABSTRACT

In Sacks' account of new technology adoption, new technologies do not transform the world, but rather they are "made at home in the world that has whatever organization it already has". In this process of being made at home, use emerges and design understandings arise. A Reflective Agile Iterative Design (RAID) framework was employed to iteratively design a mobile social group communication technology which harnessed existing mundane technology such as instant messaging and SMS, and study its emergent use. The technology was "made at home" in peoples' lives over time and led to small shifts in their communication and coordination patterns. RAID-style approaches are often glimpsed behind Web 2.0 sites and academia, although the style itself has received little direct attention in the academic literature. The approach is summed up as 'learning about technology through living with it' and is well suited to understanding and designing mundane technology with its fusion of the action research and agile development philosophies. The characteristics and challenges of designing using this approach are described.

1. INTRODUCTION

Ethnographic fieldwork has been widely adopted for developing insights about people and cultures that can inform the design of technologies. However the separation of ethnography from design has been identified as problematic because it limits the way the designer understands field studies to finding problems to be solved or gathering requirements for new designs. This restricts the ways in which practice and technology can evolve together [5].

The design of everyday mobile ubiquitous computing and communication technologies poses particular challenges for incorporating studies of culture and use into the design process. There are challenges of understanding what value a new technology might bring to people; how it will fit into various peoples' everyday living; how to design it so that it is easy to learn (either from interface cues or friends); and figuring out how to make it work well across multiple technology platforms etc.

Communication mediated by technology such as email or text messaging changes the nature of communication itself. So an ethnomethodological analytic stance that examines the ways in which people make sense of their world, display this understanding to others, and produce the mutually shared social order in which they live could be particularly instructive for understanding communication with new technologies. Still, ethnographic fieldwork and related analytic auspices rely upon data gathered directly from observation and therefore go only so far towards understanding the predicting how and whether people will embrace, adopt and adapt new designs that do not yet exist.

Use is not an inherent property of a system or artefact, rather something that is developed and maintained over time, by a number of actors, as the artefact is adopted or assimilated. As such, time is an important, and in some cases, critical, component of study. Studying how people use a system in its first week of introduction will indeed be illuminating however is unlikely to be indicative of longer-term everyday use. Some systems in particular are inextricably 'social' in that their utility only manifests when used for, with or between multiple actors.

In the design of such social systems, there is often a desire to be led by exploratory inquiry; not only to learn about the artefact, but to allow the design and study itself to unfold through use. On the surface level, this may appear to pose a 'chicken or the egg' paradox, after all how can you examine use of a system without it being built, and how can you build it before you know the form it should take. We suggest this can be resolved by using prototypes as a boot-strapping mechanism within a process that couples iterative design and emergent use. In the sections that follow the challenges of using such an iterative approach are discussed.

2. RAID

Reflective Agile Iterative Design consists of three stages: development, use and reflection, which revolve around a continuously usable exploratory prototype. The form and functionality of the design is shaped through consideration of a problem or opportunity in a particular use context. After an initial deployment of a rough prototype, use is passively observed and actively probed. Analysis and reflection on data can then take place, with the designer/researcher considering appropriate design and methodological responses. The responses are then carried out, and the process continues.

We do not seek to make any claims as to the novelty of this method, it is indeed a blend of several well-established practices and for ease of discourse, given a title. RAID-like design approaches are commonly glimpsed in CHI and CSCW work that use a long-term prototype; however, there does not appear to be a good description of the process as a whole and the key challenges that arise in execution.

The framework has been developed from our experiences of designing, deploying and studying 'Rhub', a tool for mobile social communication (described elsewhere in more detail [7]). Briefly, Rhub allows groups of friends to communicate, collaborate and share using simple technologies such as instant messaging, text messaging and the web. It provides a cohesive experience across existing mundane technologies. It was developed as a design response to the problems of mobile group communication observed in a university club that consisted of an evolving and changing membership and the continued need to organise ad-hoc socialising. Rhub was in use for over 1.5 years by over 170 participants who used the system on an everyday basis for everyday socialising needs. Mostly, this was to coordinate ad-hoc social events and activities, such as going out for dinner or drinks. This type of activity is difficult to accomplish with usual group text messages as messages must be manually relayed around the group as coordination evolves. We seeded the social network with our own friends and colleagues, who in turn invited further people to the network.

2.1 Exploratory Prototype

We use the term 'exploratory prototype' for a usable and useful system which is deployed on a long-term basis in real, natural settings. The exploratory prototype is not primarily used to investigate how this *particular* artefact is used, with its specific form, shape and texture. Rather, the prototype is used to explore emergent aspects of use. For example, how the artefact is appropriated within the group, how and what kind of norms develop, what new or changed meaning develops amongst its users and the impact of the prototype in the social environment. It is through participants and designers dwelling [2] with these systems and incorporating them into the fabric of everyday, mundane life, that new insight can be gained.

2.2 Development

Ideally, the exploratory prototype is continuously usable and useful during the study period. To enable this, some form of an agile development method is required. Using an appropriate development method is critical to enable feedback to be integrated back into the design, thus iteratively improving the prototype and exploring the design space further [1].

The development stage is where planning and implementation takes place. Existing methods for gathering early design requirements and conducting agile development are used to sketch out the first implementation. Congruent with agile development methodologies, the design and its implementation should be a 'bare minimum' of what's required. We suggest establishing a motto, or one-line manifesto for the system to keep the evolving design true to a unified spirit. A useful motto can enforce design discipline, simplicity and focus over multitudinous features.

2.3 Use

For many technology prototypes, user-created artefacts (such as messages, or uploaded photos), interactions and failed usage attempts are available to the researcher (as per privacy terms). This direct data gives the designer direct feedback about the nature of usage, as well as which features they are using and which they are not. Feedback data, by which we also mean Schön's design "back-talk", needs to be hunted for and gathered. Feedback might be direct, such as a user leaving a comment for the design team, or indirect, such as someone blogging about the system. For both types, feedback may or may not be technologically-mediated. With Rhub, a number of users were friends or acquaintances, and thus we had opportunity to observe first hand a number of spontaneous discussions that arose about Rhub and how it should be used. If we didn't happen to be there at the time, we would have missed much of this valuable data. We also had users come to us directly and give informal bug reports or comments, which many found easier to do than writing an email or message.

With an exploratory prototype, ethnographic approaches are also possible, for example observation-based fieldwork. Because people usefully use the exploratory prototype in an everyday fashion, such fieldwork has genuineness unattainable by technology probe based approaches, which, as originally formulated [8], are not meant for functional use.

2.4 Reflection

Reflection digests the available information from the design and use to produce considered design responses. It is through this reflective design step whereby the researcher opens up new areas of inquiry. Schön describes this as a "move-testing" experiment, in which theory, developed through reflection can be confirmed or negated in the course of action [10]. Understanding what to respond to and how is critical for the next iteration to progress instead of regress.

We utilised a reflective journal to keep observations, thoughts and ideas during the design process, as well as notes on the prevailing social conditions and events. The journal and its associated artefacts together permit a dialogue to take place between the design and designer as well as the research programme itself.

3. RELATED WORK

The idea of subjecting a prototype to everyday 'authentic' use is nothing new, and is part of an evolving human-centred perspective on interaction design. The Designers' Notepad [12] is an early application of this approach; iteratively developed over a period of two years, used by subjects for real-world tasks and approached in an exploratory fashion. In this case however, users had no opportunity to appropriate the prototype as it was used infrequently, sporadically, and not for very long at a time.

RAID inherits much from action research [9], an iterative social research method of four stages: planning, action, observation and evaluation. Our framework is slightly simplified, and related more closely to technology development.

Technology probes [8] take a similar approach to exploratory prototypes. Probes however aim on being a form of breaching experiment: to provoke response, reinterpretation, reflection and creativity. Technology probes, like their cultural probe predecessors, often take an abstract or ambiguous form, with their purpose established through use by participants themselves. Over time, both types of probes have seen their definition and use broaden well beyond their original definitions. 'Exploratory prototype' would be a suitable term for the many technology probes discussed in the literature which do not follow Hutchinson et al.'s formulation.

Iteration is widely recognised as an inherent part of the design process. It takes place at many levels of granularity, from Schön's reflective conversation with the materials [10] to Bellotti et al.'s iterative rounds of field work [1]. Iterative approaches to software engineering, such as agile methods, are well established. Our framework however takes a more dynamic, encompassing 'situated' view by harnessing reflections of use in the design. In our later discussion, we make some similar observations as Taylor et al. [11], particularly with regard to expectation management, the importance of reliability and the benefit of looking for - in our terms – usage 'drought'.

4. DISCUSSION

We believe RAID, sketched loosely as it is, is applicable and useful to a host of contexts. Not all systems are designed or desired to be used on an everyday basis; for these, the framework may yield little benefit. For others, particularly those in the fields of ubiquitous, mobile or social computing, learning through long-term use might be a useful endeavour. Over a period of time, the prototype is better exposed to the rigours of normal use, use that becomes more and more 'normal' and natural as participants understand and adapt the system for their own needs. The wider social backdrop of the participant and her co-participants also changes through the use of the prototype, and in turn alters how the prototype is used and perceived.

Because RAID advocates rapid and frequent design evolution, software-based systems are ideally suited, particularly if they can be updated with zero or minimal user intervention. A minimal implementation means that effort can be better spent on areas which need more 'fleshing out', rather than wasting it on a full and complete implementation of something that is not desired by users. When specific features don't work, lightweight designs are often easily superseded by an alternative approach. It is through analysis and reflection on use that the shape and function of the design unfolds. In the case of Rhub, some of our grander ambitions for the system - particularly location-based interaction - was scaled back after it became apparent that it was not something that our participants saw value in.

In our deployment, we had the advantage of being within the social context of the system's use (although not entirely, there were also complete strangers using the system), as well as being in frequent vicinity of a number of users. Thus we were participant researchers, in familiar social contexts but with the introduction of a disruptive technology.

4.1 Reliability

If a prototype is to be used over for a period of time, it is important that it is relatively stable. If the system is unavailable or unreliable, users may cease to use it, and it may be difficult to draw them back. In the case of Rhub, alarmingly soon after its deployment, people began to rely on it for communication and coordination. Rather than send emails or text messages, they would send a Rhub message. If the Rhub message did not reach the intended recipients due to a system fault, people would naturally be disappointed and reduce their usage, or use it only for inconsequential messages. In one case, a leader of a student group assignment established a Rhub group and invited the other members. A number of messages were sent to the group, and everything worked, however on the occasion that he tried to use Rhub to organise a group meeting there was a fault, and the message was not delivered. After this, the group effectively abandoned Rhub, resultantly four participants were lost and a new usage scenario was never fully studied.

4.2 Expectations

It is useful to manage participants' expectations of the prototype and its quality of functionality and service. It is easy and understandable for a participant to assume that a prototype is similar to final, polished systems they have also used. Software systems in particular are difficult to judge – the level of fit and finish is not as apparent as a physical artefact. Thus it is necessary to provide disclaimers that the system is only a prototype, and in particular signpost the rougher areas of the system where the user might encounter difficulty because of poor placeholder design or implementation. On the web, some sites manage user expectations by labelling the site as 'beta' with a prominent graphic or sub-title.

In our experience with a software-based exploratory prototype, we found that user's expectations were very high, even among those who knew it to be a research prototype developed by a single person. Participants wondered why the prototype lacked features they expected of similar systems, even though said features were rather peripheral to the prototype. Software is seen as a highly malleable medium, and users did often not appreciate the effort required to implement features.

4.3 Communicating change

If the design is undergoing continuous change, consideration must be paid to how users are notified of new features or options, as well as inconvenience caused by change in the interface or processes. If change is not properly communicated, there is reduced potential for active use, thus limiting feedback and impoverishing the entire process. Commercial systems sometimes employ a mailing list, development blog or change log to keep users informed. For the many disinterested users who would not read such material, designers often use visual highlighting to draw attention to changes. With Rhub we employed a mixture of these techniques: a low-volume mailing list, a high-volume blog as well as visual indicators on the website. Because we were often in physical contact with our users, we had the advantage of being able to introduce new features on a personal basis to those we thought would benefit most. Usage of some features is observable by others, for example starting an opt-out group discussion using Rhub. For these features, it was useful to introduce them through use, either by ourselves or by encouraging early adopters to try them out. Other users who were perhaps not quite as adventurous could then see how a feature was used, how it would be seen by others and perhaps learn more and use it themselves.

4.4 Feedback through use

As Crabtree et al. identify [4], there are challenges in studying the use of some forms of technology, such as ubiquitous computing. We would also explicitly add mobile and social computing as two other forms of computing which pose new difficulties for interaction design researchers. The need to "reconcile the fragments" [4] of data from disparate sources can be accomplished a number of ways. With Rhub, our date-stamped reflection journal could be synchronised with usage logs, change logs and source code repository commits. Combining this data allowed us to form a rich picture of activity; both the users' and our own meddling design changes.

Data is not always available in a form for ready analysis, and some work is often required to pre-process it [3]. For example, to aid in analysis of conversations, we had to devise an algorithm to segment message streams based on interval between messages. In the case of Rhub, analysis of messages was moderated by the level of our entanglement in the social context. When the social context is alien to the researchers, participatory design sessions with people experienced in the context may be a useful complement to usage analysis.

We observed six major usage patterns often useful for design: deluge, accretion, drought, erosion, missteps and discovery.

4.4.1 Deluge

A rapid influx of usage can expose scalability issues with usability and technical implementation. For example, one group of Rhub users treated it like an instant messaging system, sending a large number of short rapid messages, which caused messages to be delivered out of order and some sub-systems to crash. This incident occurred after many months of stable usage of Rhub by a much larger group, who used it more like text messaging, sending longer, but less rapid messages.

4.4.2 Accretion

In systems that allow people to create artefacts or other by-products of use, gradual accumulation may also reveal deficiencies. Rhub allowed users to upload photos, create groups,

tags and so on. Over time, as the number of these artefacts rose, more sophisticated techniques were required to support users in browsing, searching and managing these artefacts.

4.4.3 Drought

For some designs, under-use or non-existent use of features can lead to barrenness in the visual presentation, or may hamper the viability of other dependant features. Potential responses to drought might be to look for underlying technical faults, considering removing the feature, or redesigning it entirely. Rhub had several location-based services whose full potential was never realised because not enough users were setting their location. To address this, we successively made it easier and more rewarding to set your location which in turn led to greater use of dependant features.

4.4.4 Wearing in

Erosion or wear can reveal repeated use in physical artefacts, for example the patterns of worn paint on a mobile phone might reveal how it is usually held. A car engine wears in before it runs smoothly. Software-based systems do not physically wear, however user activity can be logged and then later analysed for trends and established usage patterns. For example usability can be improved by examining, and then shortening, common navigation paths.

4.4.5 Missteps

Users can easily make mistakes, either causing an error, or causing some effect they did not intend. The former can be found easily through logging user input and action as well as the system response. The later is more difficult, and often relies either on users reporting something went wrong, or more likely, through observation of activity logs. Interaction with Rhub's SMS interface was significantly improved based on examining user input that was rejected by the parser. For example, a user might try a command they thought should work, but parser would reject it. We would see this error, consider if the command was worth supporting and then implement it.

4.4.6 Discovery

Once in use, a new communication medium will surprise with unanticipated uses. With Rhub, a message sent to a co-located group causes all phones to ring at once. People tended to send informal invitations that we described as half-invitations. Other discoveries that led to shifts in communication are reported in [7]. Over time it became evident that groups appropriated Rhub into their lives and relied on it for day to day communication. Rhub enhanced the feeling of belonging to a group and led to more group invitations and events than would otherwise have occurred.

4.5 Confined design

One of the concerns rightfully levelled at the use of long-term prototypes is that once deployed, the potential for the design to evolve is limited. If the wrong paradigm was pursued to begin with, the designer risks following a dead end path. Use of lightweight prototyping can furnish an array of different designs with minimal effort; however, these prototypes do not yield the same form and depth of data. Clearly there is a place for both approaches. For example, conducting early workshops with numerous design alternatives to - in Bill Buxton's terms - get the *right design*, followed with a course of long-term use to get the *design right*. More creative mixes of these approaches is also possible. With Rhub, we were able to completely redesign some aspects of functionality whilst still within the overarching design paradigm. Data from long-term use can also feed into fresh, experimental designs which can be explored tangentially.

5. CONCLUSIONS

Designing iteratively with regard to emergent use is an effective way of creating and understanding interactive systems, particularly those with an element of social use. Prototypes deployed on a long term basis are exposed to a greater variety of use and give people a chance to appropriate the system into everyday use. Reflective Agile Iterative Design is a framework we developed during our experiences in designing a mobile social software system. Over many iterations of a continuously-usable prototype, a design is improved through reflective design responses to usage. Designs in use undergo phases of deluge, accretion, drought, wearing in, missteps and discovery. Establishing and staying true to a motto for the design is a useful way to filter and prioritise potential new features. One of the greater potentials for the framework is to better integrate the disciplines of design, social science and engineering.

6. REFERENCES

[1] Bellotti, V., Ducheneaut, N., Howard, M., Smith, I., and Neuwirth, C. 2002. Innovation in extremis: evolving an application for the critical work of email and information management. In *Proc. DIS'02*. ACM, 181-192

[2] Brown, B. and Randell, R. 2004. Building a context sensitive telephone: Some hopes and pitfalls for context sensitive computing. Journal of CSCW 13:3, 329-345.

[3] Cheverst, K., Fitton, D., Rouncefield, M. and Graham, C. 2004. 'Smart Mobs' and Technology Probes: Evaluating Texting at Work. In *Proc. ECITE 2004*. 73-80.

[4] Crabtree, A., Benford, S., Greenhalgh, C., Tennent, P., Chalmers, M., and Brown, B. 2006. Supporting ethnographic studies of ubiquitous computing in the wild. In *Proc. DIS'06*. ACM, 60-69.

[5] Dourish, P. 2006. *Implications for design*. In *Proc. CHI'06*. ACM, 541-550.

[6] Grudin, J. 1994. Groupware and social dynamics. *Commun. ACM* 37:1. ACM, 92-105.

[7] Heyer, C., Brereton, M. and Viller, S. 2008. Cross-channel mobile social software. In *Proc. CHI'08*. ACM, 1525-1534.

[8] Hutchinson, H., Mackay, W., Westerlund, B., et al. 2003. Technology probes: inspiring design for and with families. In *Proc. CHI'03*. ACM, 17-24.

[9] Lewin, K. 1946. Action research and minority problems. *Journal of Social Issues* 2:4, 34-46.

[10] Schön, D. 1990. The design process. In V Howard (ed). Varieties of Thinking. New York: Routledge, 1990

[11] Taylor, N., Cheverst, K., Fitton, D., Race, N. J., Rouncefield, M., and Graham, C. 2007. Probing communities: study of a village photo display. In *Proc. OZCHI '07*. ACM, 17-24.

[12] Twidale, M., Rodden, T., and Sommerville, I. 1993. The Designers' Notepad: Supporting and understanding cooperative design. In Proc. ECSCW. Kluwer 93-108.