A review of the Getting Real software development approach

Jari Sarja

Raahe Unit,
University of Oulu,
Oulu, FI-92100, Finland
Fax: +358-8-221-406
E-mail: jari.sarja@oulu.fi

Abstract: The small US software developing company called 37signals is a significant phenomenon in many ways. The company has launched successful commercial web applications. The company has its own unique way of business thinking, and the people of the company do not have any growth target.

The people of the company also have their own way of thinking about the software developing process. They have named it Getting Real. Getting Real is not described as a developing method; rather, it is a philosophy or approach behind the development activities. It has confluence with agile methods, but unlike agile methods, in Getting Real the nature of the final product is also strongly emphasised.

The success of the company has proven that there is some effectiveness in the Getting Real approach. Therefore, the aim of this study is to find out whether the Getting Real approach can be supported by previous professional literature and scientific research.

Keywords: software development; agile methods; small business; the Getting Real approach.

Reference to this paper should be made as follows: Sarja, J. (2012) 'A review of the Getting Real software development approach', *Int. J. Agile and Extreme Software Development*, Vol. 1, No. 1, pp.78–94.

Biographical notes: Jari Sarja received his MSc degree (Information Processing Science) from the University of Oulu, Finland. Currently, he is a Project Researcher and a PhD student at the University of Oulu, in a side unit of Raahe. His research is mainly focused on the area of software development: agile methodologies, user experience and renewable energy. Before his research career, he has worked a long period in the electronics and component industry.

1 Introduction

This study identifies the professional and scientific evidence for the software development philosophy or approach called *Getting Real*. The Getting Real approach was created by an US company called 37signals. The key persons[1] of the 37signals have

created the Getting Real approach based on their own experience about developing software applications and commercialising them successfully.

The key persons of the 37signals have written two books about their way of thinking about software development (*Getting Real*)[2] and creating a profitable business in the internet environment (*Rework*). These two books, *Getting Real* (Fried and Heinemeier Hansson, 2006) and *Rework* (Fried and Heinemeier Hansson, 2010), are important empirical sources in this study. In *Rework*, the authors write the following about the book: "This book isn't based on academic theories. It's based on our experience. We've been in business for more than ten years. Along the way, we've seen two recessions, one burst bubble, business-model shifts, and doom-and-gloom predictions come and go – and we've remained profitable through it all" [Fried and Heinemeier Hansson, (2010), p.3]. The same principle applies to the *Getting Real* book.

We call Getting Real an approach rather than a method, technique or procedure. Getting Real is not a described or documented software development method. It is a range of principles based on the good practices of the 37signals company. The Getting Real approach has confluence with agile methods.

What is noteworthy is that 37signals is a small bootstrapped company without any expanding purpose. Regardless of that, they have gathered an active audience that follows the company. Widespread newspapers and magazines, such as *The New York Times*, *Time*, and *The Wall Street Journal*, have written stories about the key persons of the company, and they have been crowd-pleasing speakers in different events. It can be said that the outcome of the company is much wider than just the products they have developed; it consist of a different way of processing development and business thinking, and by-products. The company seems to enjoy a strong charisma of lonely riders, or even a little bit rebellious pioneers.

Still two more issues exist, which make Getting Real an interesting research subject: according to databases, the key persons of the company and their books have been cited a few dozen times in academic researches and articles. Moreover, the Getting Real approach has not been the subject of a research before in a comprehensive way.

From a practical point of view, this study may establish if the studied development and business models – or part of them – are useful benchmarking targets to other business owners. From a theoretical point of view, the purpose of this study is to find out if there exists any support from professional literature and previous research for the Getting Real software development approach. The main contribution is to connect the proven successful small business development and business ways to a professional and scientific context. The research question is: *Can the Getting Real approach be supported by professional literature and previous scientific research?*

Since the Getting Real approach has not been researched before in a comprehensive way and it has confluence with agile methods, the studied previous research material mainly concerns agile methodologies. Abrahamsson et al. (2002, 2003) have observed that agile methodologies have evoked a substantial amount of literature. The Agile Manifesto (2011) is an important source in this research because it has been the starting point for agile definitions, rules, principles, and it is cited in most literature concerning agile methodologies. The influential persons behind the Agile Manifesto are also notable book and article writers.

The research method in this study is conceptual analysis. Järvinen (2004) describes the nature of conceptual analysis by setting the question: "What is a part of reality according to a certain theory, model, or framework?" This study applies conceptual

analysis in the contrary manner; it asks what is a part of theory according to certain practical activities? The data has been collected from many different sources: books, magazines, the internet, professional literature, scientific papers, and video clips. The collected data has been analysed using the systematic review and transcription methods. Systematic review can be undertaken to examine the extent to which empirical evidence supports theoretical hypotheses (Kitchenham, 2004). This is a very important aspect in this study, since the main research activity is to compare the empirical material to professional literature and previous researches. Transcription means converting the source text to another format, for instance from spoken language to a written form.

The second section presents the previous studies on agile methodologies. An important part of this section is the description of the Agile Manifesto. Since there exists numerous agile methods, we have chosen four of them for general inspection: extreme programming, scrum, crystal methods, and feature-driven development. The company behind the Getting Real approach is also presented in the second section. Without knowing the company behind the approach, it would be difficult to gain an understanding of the approach.

In the third section, we describe six separate Getting Real software development principles, and the conclusions of this study are summarised in the fourth section.

2 Literature review

Software business is a much newer business line compared to manufacturing business lines. That is why traditional software development methods are based on the *generic development process* (or *new product development* process, NPD), which is widely used in various manufacturing business lines. Ulrich and Eppinger (2008, pp.13–15) describe the generic development process as a six-step process, which consists of the planning, concept development, system-level design, detail design, testing and refinement and production ramp-up phases. The traditional software development processes (e.g., Waterfall, Stage-Gate) are described from a quality-related point of view. For improving the quality of output from the process the focus has to be laid on the process itself by removing the variances of the process. There are a quality checkpoint between every working phase and the quality criteria must pass before moving to next working phase.

The most important reason for criticism of traditional software development processes is the inflexibility for changes. Avison and Fitzgerald (1991), MacCormack et al. (2001), Nandhakumar and Avison (1999), and Parnas and Clements (1986) share the idea that traditional development methods are control-oriented, too mechanistic to use in detail, too ideal and hypothetical, and not working in dynamic environment. This provides a background for the emergence of agile software development methods.

The major idea behind agile methods is to speed up the development time and to allow later changes to requirements. The number of different agile methods is existing and therefore it is difficult to find a common definition for an agile method. Strode (2006) has made a common definition for an agile method:

> "An agile method is a software development methodology designed for the management and support of iterative and incremental development of business systems in environments where change is constant. Agile methods use software development techniques that enhance teamwork in small empowered teams and support active customer involvement. An agile method is designed to produce

> working software early, using communication, feedback, learning and frequent meetings in preference to than modelling and documentation. Agile methods adapt existing software development techniques to achieve these goals".

The Agile Manifesto includes general rules and principles for agile methods. It was signed by 17 persons influential in the agile field in 2001 [Agile Manifesto, 2011; Cockburn, (2002), p.213; Lindstrom and Jeffries, 2004]. The people behind the Agile Manifesto were individuals who had published separate software development methods with similar characteristics. All these methods are based on best practice experiences and evolutionary development practices focusing on early delivery and quality of software (Strode, 2006).

The common rules in the Agile 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."

The Agile Manifesto further includes 12 explicit principles. These 12 principles of agile software are:

- "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity – the art of maximizing the amount of work not done – is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly." (Agile Manifesto, 2011)

The Agile Manifesto is an important source in this study because it has been a starting point for agile definitions, and it is cited in most literature concerning agile

methodologies. Because there exist numerous of different agile methods, we have selected four agile methods for closer inspection; *extreme programming*, *scrum*, *crystal methods*, and *feature driven development*, which belong to agile method family.

2.1 The company behind the Getting Real approach

The background information of the company, 37signals, helps to deepen understanding of the Getting Real approach, which is the research target. Without knowing the company behind the approach, it would be difficult to gain an understanding of the approach. The presented issues are those that appear time and again when the company is spoken about. These issues are the big audience, the nature of the products and the unique business models of the company.

37signals is followed increasingly, and it has a loyal audience. The company has built the audience on purpose, as a kind of affordable marketing strategy. The company launched a weblog titled *Signal vs. Noise* in 1999. According to Fried and Heinemeier Hansson (2010, p.170) it had more than 100,000 daily readers in 2010. The fans and audience mean a lot to the company. Having an audience means an affordable way to reach a great number of people and potential customers, and to get direct feedback without any information barriers. The big group of followers of the small company, the audience, makes the company even more interesting from a research point of view.

The common factor of 37signals' products is that they have all been planned to be easy to use, opinionated, and relatively light and simple overall. Considering how successful these products have been, it can be said that at least a part of customers like simple products that do not require a lot of training before they can be used. After decades' evolution with increasing features and complexity, simplicity and minimalism might be the forthcoming trend in the software business, perhaps in other business lines as well.

The 37signals people summarise the idea of simplicity as the modus operandi of the company as follows:

> Our modus operandi:
>
> "We believe software is too complex. Too many features, too many buttons, too much to learn. Our products do less than the competition – intentionally. We build products that work smarter, feel better, allow you to do things your way, and are easier to use." (Fried and Heinemeier Hansson, 2006)

37signals has a unique way of thinking about the running of a business. They have own ideas for example about company funding, product pricing, as well as about every day working methods.

3 The Getting Real approach

Getting Real is a kind of way of lateral business thinking. It is a set of principles that lead the activities of a company. It is relatively difficult to define Getting Real, nor is it clearly defined by the company. It is not a software development method since the development process is not determined in it, and it also involves working and business methods. Getting Real is based on the company's experience about developing software applications and commercialising them.

After reviewing the source material, Getting Real is defined in this research as an approach that has an existence identical with the philosophies behind software development methods. It is presumable that the company people use 'ultra-light agile methods' without any formal documentation as their development method.

There are six separate general principles in total:

> "1 Getting Real is about skipping all the stuff that represents real (charts, graphs, boxes, arrows, schematics, wireframes, etc.) and actually building the real thing.
>
> 2 Getting Real is less. Less mass, less software, less features, less paperwork, less of everything that's not essential (and most of what you think is essential actually isn't).
>
> 3 Getting Real is staying small and being agile.
>
> 4 Getting Real starts with the interface, the real screens that people are going to use. It begins with what the customer actually experiences and builds backwards from there. This lets you get the interface right before you get the software wrong.
>
> 5 Getting Real is about iterations and lowering the cost of change. Getting Real is all about launching, tweaking, and constantly improving which makes it a perfect approach for web-based software.
>
> 6 Getting Real delivers just what customers need and eliminates anything they don't." (Fried and Heinemeier Hansson, 2006)

The key persons of the company define the benefits of the Getting Real approach as follows:

> "Getting Real delivers better results because it forces you to deal with the actual problems you're trying to solve instead of your ideas about those problems. It forces you to deal with reality. Getting Real foregoes functional specs and other transitory documentation in favour of building real screens. A functional spec is make-believe, an illusion of agreement, while an actual web page is reality. That's what your customers are going to see and use. That's what matters. Getting Real gets you there faster. And that means you're making software decisions based on the real thing instead of abstract notions." (Fried and Heinemeier Hansson, 2006)

In order to clarify the analysis of the Getting Real theme, the principles are numbered (1–6). Each principle is introduced in detail below and compared with professional literature and scientific findings.

3.1 Getting Real – Principle 1

> "Getting Real is about skipping all the stuff that represents real (charts, graphs, boxes, arrows, schematics, wireframes, etc.) and actually building the real thing."

The first Getting Real principle is very similar to the common agile definitions regarding documentation. Already the set of common general rules for all agile methods – the Agile Manifesto – recommends focusing on well-functioning software instead of documentation: "Working software over comprehensive documentation". The Agile Manifesto further includes 12 explicit principles. One of them is to focus on the

development of process measurement: “Working software is the primary measure of progress”.

In her research, Strode (2006) introduces common properties of agile methods and also mentions working software as the main product of development, together with minimising documentation.

The key persons of the company criticise the need for comprehensive operational documentation – not only documentation related to software developing. Besides specifications, the key persons of the company call into question the need for roadmaps, projections (Fried, 2008), business plans, five-year plans (Heinemeier Hansson, 2009), financial plans, and strategies [Fried and Heinemeier Hansson, (2010), p.19]. Many professionals share the thinking about minimising documentation. Torvalds, the creator of Linux operating system, says about specifications: “A spec is close to useless. I have never seen a spec that was both big enough to be useful and accurate. And I have seen lots of total crap work that was based on specs. It’s the single worst way to write software, because it by definition means that the software was written to match theory, not reality” (Torvalds, 2005).

Palmer and Felsing (2002, pp.100–101) state that it is a painful process to generate documents of source code and it increases the chances for updating. Cockburn (2002, p.177) suggests to dispense with design documentation beyond whiteboard sketches. This is in line with what the key persons argue; they recommend using paper sketches and real HTML screens in the planning phase instead of documents (Fried and Heinemeier Hansson, 2006).

Principle summary

The first Getting real principle closely resembles one of the four common rules for agile methods, it is only defined in a somewhat more detailed and extensive way. We can conclude that the first Getting Real principle is supported by professional literature and previous scientific research.

3.2 Getting Real – Principle 2

> “Getting real is less. Less mass, less software, less features, less paperwork, less of everything that’s not essential (and most of what you think is essential actually isn’t).”

Fried and Heinemeier Hansson (2006; 2010, pp.62–63) describe the second principle as follows: “If you keep your mass low, you can quickly change anything: your entire business model, product, feature set, and/or marketing message. You can make mistakes and fix them quickly. You can change your priorities, product mix, or focus”.

One of the principles in the Agile Manifesto can be compared to this second principle: “Simplicity – the art of maximising the amount of work not done – is essential”. The key persons of the company emphasise to do less at every level; the code level, feature level, daily routine level as well as company strategy level.

All characteristics of the agile methods, as well as those of the Getting Real approach, aim at flexible changes during the developing process. Appleton (2005) has summarised this idea as follows: “There is no *code* that is more flexible than *no* code!” He argues the good software design is not knowing what to put into code but it is knowing what to leave out.

The key persons of the company see many different advantages in reducing codes. Less software is easier to manage, it reduces the code-base, which means less maintenance work, it lowers and speeds up the cost of change, and it causes fewer bugs and reduces the need of support. "For every feature that makes it into your app, ask yourself: Is there a way this can be added that won't require as much software? Write just the code you need and no more. Your app will be leaner and healthier as a result" (Fried and Heinemeier Hansson, 2006).

Wild (2008) recommends writing less code by justifying and prioritising every feature and minimising useful feature sets. It is a received principle that software design should be kept as simple as possible. For instance, Appleton (2005), Fernandez (2008), Lindstrom and Jeffries (2004), Müller and Tichy (2001), Nielsen and Mack (1994) and Wild (2008) share this principle with the 37signals people. According to the key persons, feature evaluation should always be done by thinking what is really needed, and leaving out the rest.

The 'less mass' philosophy seems to present even a competitive edge to the company. "It's all part of how we differentiate ourselves from competitors; instead of trying to build products that do more, we build products that do less" (Fried and Heinemeier Hansson, 2006). The key persons of the company highlight the minimalist character in every source when they speak about the products (e.g., Fried and Heinemeier Hansson, 2006, 2010; Fried, 2008; Heinemeier Hansson, 2009; Park, 2008, etc).

It is difficult to prove scientifically whether there is any correlation between the number of features and the success of the product since there is not very much previous research available on this subject. There are also many variables, e.g., different researched customer segments [the customer segments according to Moore (1991): innovators, visionaries, pragmatists, conservatives, sceptics]. But it can be concluded that the fewer features a product contains, the simpler it is to use. Simple-to-use products always have loyal users. The company has proved it with more than three million users [Fried and Heinemeier Hansson, (2010), p.3]. As mentioned, minimalism might be a growing trend also in other business lines, such as consumer electronics or catering business. The key persons refer to Gordon Ramsey – a three-Michelin-star chef – who recommends to have only around ten dishes on a menu and to focus on them [Fried and Heinemeier Hansson, (2010), p.83].

Principle summary

The second Getting Real principle is very general and extensive. It recommends doing things in a simple and light way at various levels. The phrase 'less mass' covers practically all functions of a software company, including the daily-level job, planning, documentation, and product design activities. The phrases 'less paperwork' are more descriptive and belong under the 'less mass' umbrella. The general less mass thinking – including all sub-thoughts – is in line with the Agile Manifesto principle: "Simplicity – the art of maximising the amount of work not done – is essential".

The idea of 'less software' and keeping the code as simple as possible is clearly in accordance with the agile approaches and is shared for instance by Appleton (2005), Fernandez (2008), Lindstrom and Jeffries (2004), Müller and Tichy (2001), Nielsen and Mack (1994) and Wild (2008).

The 'less feature' thinking is very close to the 'less software' thinking since in most cases less software is the result of reducing the number of features. It is not possible to

reach a conclusion concerning the correlation between the number of features and the success of the product because there are so many different kinds of customer segments. It is certain that innovators and visionaries want more features than mainstream customers. On the other hand, it can be seen that many customers want products that are easy to use, which in many cases means fewer features. The 'less paperwork' thinking is clearly in line with the agile methodologies and is defined in the Agile Manifesto as one of the four common rules: "Working software over comprehensive documentation".

We can conclude that the second Getting Real principle is supported by professional literature and scientific research.

3.3 Getting Real – Principle 3

> "Getting Real is staying small and being agile."

This is the first instance in which the key persons mention the concept of agility. However, they do not seem to refer directly to agile methodologies; rather, agility is a consequence of the small size of the company. "All the cash, all the marketing, all the people in the world can't buy the agility you get from being small" (Fried and Heinemeier Hansson, 2006).

This is also the first Getting Real principle which does not have a straightforward connection with the Agile Manifesto rules or principles; in the Agile Manifesto itself, agile methodologies are not linked only with small teams. However, some key persons behind the Agile Manifesto (e.g., Beck, 1999; Lindstrom and Jeffries, 2004) argue that particular agile methods are meant for small teams. Some researchers (e.g., Müller and Tichy, 2001; Rising and Janoff 2000; Strode, 2006) also share the idea of the small team size.

The key persons of the company do not speak about the size of the developing team but the size of the whole company. They point out that a small business can be profitable and that growth itself should not be a main target of a company [Fried and Heinemeier Hansson, (2010), pp.22–23]. They emphasise four main reasons why it is favourable to keep a company small:

- the possibility of cheap and fast changes
- resource limitations force one to do things faster and cheaper
- fewer formalities, less bureaucracy, and more freedom
- nimbler organisation.

Power and Reid (2005) have researched the flexibility and performance of small firms, and they have identified the main factors that influence the performance of long-lived small businesses positively. Two out of the four factors support this principle. A small firm must be aware of the drivers of change, and it must be ready for quick changes.

Principle summary

The Agile Manifesto does not commit itself to the size of a company, but one of its main ideas is to react fast to requirement changes even in a late phase of the developing process. The fourth common rule of the Agile Manifesto is "Responding to change over following a plan", and one of the 12 explicit principles is: "Welcome changing

requirements, even late in development. Agile processes harness change for the customer's competitive advantage". Agile behaviour and the possibility of reacting fast are a consequence of the small size of a team or company. It is an established fact that small companies are more agile and faster than bigger ones (e.g., Power and Reid, 2005).

We can conclude that the third Getting Real principle is supported by previous scientific research.

3.4 Getting Real – Principle 4

> "Getting Real starts with the interface, the real screens that people are going to use. It begins with what the customer actually experiences and builds backwards from there. This lets you get the interface right before you get the software wrong."

The order of building software is not addressed in the Agile Manifesto. One rule is loosely similar and closest to this principle; it is the one that was already introduced with the first principle – "Working software over comprehensive documentation". The key persons of the company emphasise in many sources that it is important to start the building of software directly with real things without formal planning and documentation (e.g., Fried and Heinemeier Hansson, 2006; Fried, 2008), and the fourth principle states that the starting point should be the user interface.

The key persons also equate the user interface to a product. They state the user interface is a product from user point of view (Fried and Heinemeier Hansson, 2006; Singer, 2008). The key persons of the company argue for starting the interface design first because it is relatively light and easy to change before the programming has started. They also argue that the user interface gives an impression of the application to the designers.

Constantine and Lockwood (2002) note that the web page itself is a user interface. They also claim that the success of the user interface design determines the success of web applications. Nielsen and Mack (1994) have estimated that billions of dollars have been lost in internet sales because of usability problems. It is hard to find support for starting the design from the user interface, or for the opposite viewpoint. In all likelihood, the issue has not been researched extensively. However, Parnas (1969) notes in his paper that the user interface should be designed first, and thus share the viewpoint of the key persons.

The 37signals people have focused strongly on user interface design, also at the more detailed level. They mention the same principles as with whole products; the user interface must be easy to use, opinionated and aesthetic. They emphasise the meaning of blank slate design, and the form of context and language, e.g., buttons, links, search functions, words and sentences, etc. [Fried and Heinemeier Hansson, 2006; Singer, 2008; see also Nielsen and Mack, (1994), pp.279–293]. The language of the user interface has also been studied before. De Souza (1993) presents the semiotic engineering approach for user interface designers, which has similarities with this Getting Real principle.

User interfaces have been researched already before the internet became a common phenomenon. Grudin (1989) had already concluded that context is more important than consistency, against previous studies. He also states that knowing the users and their tasks can be a cutting edge for the designers. Grudin and Gentner (1990) also emphasise the difference between the engineer's and user perspectives when designing the user interface.

Principle summary

The fourth Getting Real principle is ambiguous. The first point suggests to start the design of an application from the user interface and to design the real screens first. This seems to be based on the best practices of the company, and it is certainly a good observation. It is maybe stating the obvious for software designers, or it has not been researched a lot. In any case, it is hard to find support from professional literature and research for this point.

The second point – the user experience – is loosely connected with this principle. However, the company has focused on and described user interface design from the user's point of view, in other words the user's experience, so deeply that it is valuable to summarise the point. All aspects the 37signals people present, the form of context and design including the language of user interface, are supported to some extent by previous research. A noteworthy matter is that most of the relevant studies are relatively old, from the time before the internet became common. The newer user interface research mostly focuses on more complicated user interfaces. However, it can be concluded that the same principles are valid with simple and minimalist web applications.

We can conclude that the fourth Getting Real principle is partly supported by professional literature and scientific research.

3.5 Getting Real – Principle 5

> "Getting Real is about iterations and lowering the cost of change. Getting Real is all about launching, tweaking, and constantly improving which makes it a perfect approach for web-based software."

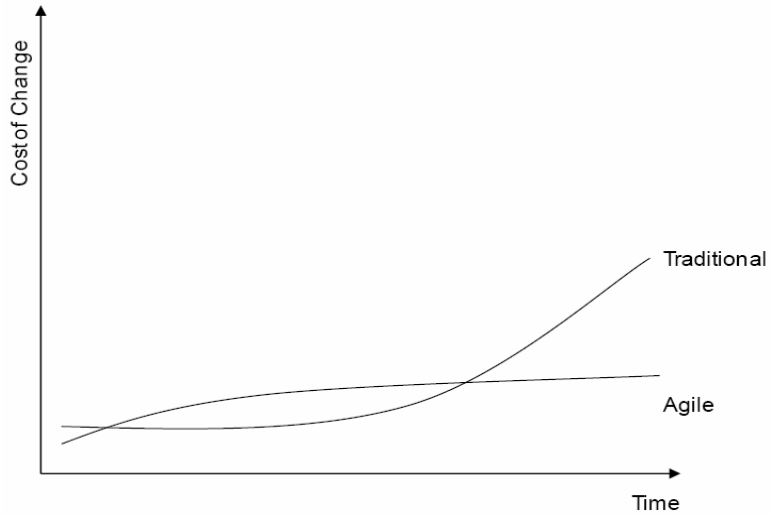
The fifth Getting Real principle is not directly supported by the Agile Manifesto rules or principles. The Agile Manifesto addresses the possibility of change, but from a different point of view. The Agile Manifesto rule and principle "Responding to change over following a plan" and "Welcome changing requirements, even late in development", refer to one of the main characteristics of agile methods – flexibility for changes, but not for the cost point of view. However, it is self-evident that lowering the cost of change is one important motive behind the establishment of agile methodologies.

The key persons of the company emphasise the change possibility from the cost point of view. The cost thinking is in line with their experience of limited development resources and the minimalist design principles.

Kunz et al. (2008) summarise the relationship between the cost of change and agile methodologies saying agile software development methods try to decrease the cost of change and therewith reduce the overall development costs. The different cost of change in agile software development in comparison with traditional software development according to the project progress as suggested by Beck (1999) is shown in Figure 1.

There is a strong connection between iterations and agile methodologies. Miller (2001) has defined nine characteristics which make a software development process agile. One of those characteristics is iteration; a short cycle which is repeated many times for refining the deliverables and completing activities. Kunz et al. (2008) emphasise the same aspect of extreme programming.

Figure 1 The cost of change compared to the development method



Source: Beck (1999)

The key persons of the company have described the iterative process as follows:

> "Instead of banking on getting everything right up front, the iterative process lets you continue to make informed decisions as you go along. Plus, you'll get an active app up and running quicker since you're not striving for perfection right out the gate. The result is real feedback and real guidance on what requires your attention." (Fried and Heinemeier Hansson, 2006)

They do not mention agile methodologies but speak about the exactly same iteration characteristic as the researchers and co-founders of agile methodologies.

Principle summary

The Agile Manifesto strongly supports flexibility for changes. It does not mention the cost of change viewpoint. However, lowering the cost of change has been recognised as one important motive behind the establishment of agile methodologies, and it is an important characteristic of agile methodologies in general. In the fifth Getting Real principle, iteration is regarded as a method of implementing the lowering of the cost of change. It is totally in line with the characteristics of agile methodologies (see e.g., Miller, 2001; Strode, 2006). A noteworthy matter is that also the second Getting Real principle introduces many other methods of implementing the lowering of the cost of change. It can be said that to lower the cost of change is a consequence of iteration, but also of other methods of implementation introduced with the second principle.

We can conclude that the fifth Getting Real principle is supported by professional literature and scientific research.

3.6 Getting Real – Principle 6

> "Getting Real delivers just what customers need and eliminates anything they don't."

The 37signals people have emphasised the importance of eliminating extra features in many sources, as has been discussed before. They have turned it into a competitive advantage, calling it 'underdoing the competition'. The content of the sixth Getting Real principle resembles the previously introduced 'less software-thinking'. Less software means less features, less code, and less waste. All these methods and mindsets have been introduced previously in this study. Therefore, we examine one small new viewpoint in this chapter, the waste eliminating.

Wild (2008) has defined principles of lean thinking. One of the seven principles is *eliminate waste*. Wild defines waste as follows:

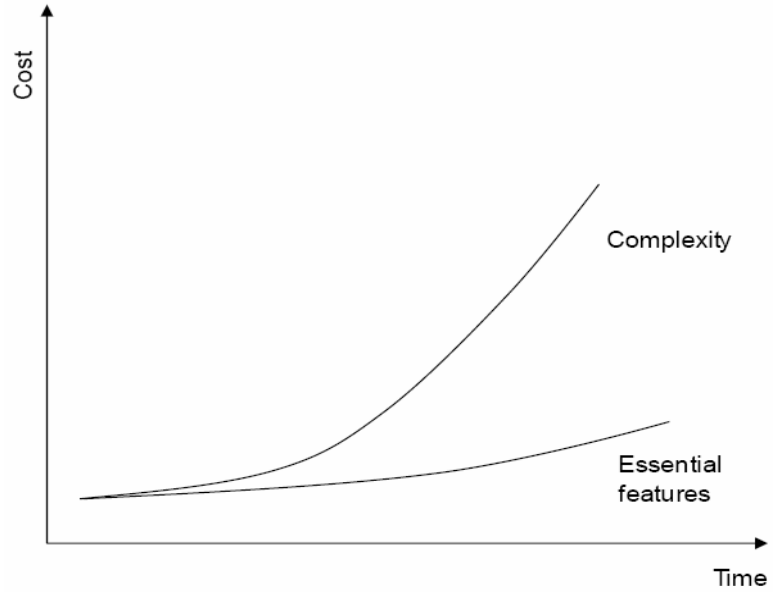
- anything that does not create value for the customer
- the customer would be equally happy with the software without it.

He explains the prime directive of lean thinking:

- create *value* for the customer
- improve the *value stream* by removing non-value-adding activities.

The cost of complexity as suggested by Wild is illustrated in Figure 2. The curve titled complexity simply means that more features cause more waste. According to Wild complexity is the biggest source of waste.

Figure 2 The cost of complexity

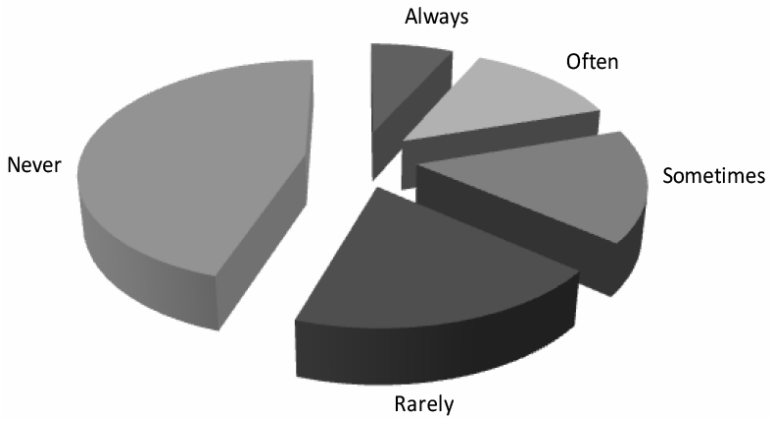


Source: Wild (2008)

Wild has found that only 7% of features and functions are always used in typical systems. 13% of them are used often, 16% sometimes, 19% rarely, and 45% never. The percentages are illustrated in Figure 3. It means that only 20% of features and functions are used always or at least often. This means that, roughly speaking, 80% of features and functions are against lean thinking and are waste. Avoiding these 80% could make a lot

of savings in the development and maintaining phases. This theory is totally in line with the sixth Getting Real principle and supports it.

Figure 3 The features and functions used in a typical system



Source: Wild (2008)

Principle summary

It can be concluded that the sixth Getting Real principle is relatively universal by nature. It is a higher-level principle compared to the other Getting Real principles, and it summarises many other principles and mindsets of the company.

The sixth Getting Real principle is also in line with the Agile Manifesto principle: "Simplicity – the art of maximising the amount of work not done – is essential". It is also supported by Nielsen and Mack's (1994) usability heuristics (aesthetic and minimalist design), as well as by Wild's (2008) principles of lean thinking (eliminate waste).

We can conclude that the sixth Getting Real principle is supported by previous scientific research.

4 Conclusions and discussion

Considering the size of the researched company, the company and the Getting Real approach are relatively well known, especially in its home country and by special interest groups. The key persons of the company and their books have been cited a few dozen times in academic researches and articles. The Getting Real approach has not been a research subject before in a comprehensive way, so it has been interesting to connect the provenly successful small business (or small organisation) development and business ways to a scientific context.

Following the investigation of the Getting Real principles, it can be concluded that many principles include a similar message. Perhaps, the authors have meant a somewhat different viewpoint between the principles. However, the main message of the principles could be summarised as follows: *The application should have as few features as possible, and the whole development process should focus on building the real things directly instead of deep planning and documentation. That, and being small, make late changes*

possible, and also keep the cost of changes reasonable. For instance, the messages of the first and fourth principles are very similar to one another, as well as the messages of the second and sixth principles.

The Getting Real principles resemble the Agile Manifesto rules and principles, but the viewpoint is somewhat different. The Getting Real principles deal with the nature of an application, which limits its use only to light products, such as web applications. The rules and principles of agile methodologies also address other viewpoints, such as individuals, teams, customers, collaboration, and reviews. However, the agile rules and principles do not commit to the nature of the final product. It would be interesting to know if the 37signals people have thought about the agile rules when defining the Getting Real principles. We tried to find that out with a short personal e-mail interview, but Fried's answer was a polite refusal explained by the lack of time (personal e-mail, 24.5.2011).

The support from previous research for the Getting Real principles is introduced in Table 1. It shows that the Getting Real principles, which reportedly are based on the best practices of the researched company, are mostly supported by Agile Manifesto, experts and specialists, professional literature, and previous scientific research.

Table 1 Professional and scientific support for the Getting Real principles

Getting Real principle	*Support from*	
	Agile manifesto/Agile methodologies	*Professional literature and scientific papers*
1	x	x
2	x	x
3	Partly	x
4	Partly	Partly
5	x	x
6	x	x

We can conclude that four out of the six Getting Real principles are supported by the Agile Manifesto. These four principles are clearly supported by other professional literature and researches as well. One out of the six Getting Real principles is not directly supported by the Agile Manifesto, but there are other professional literature and researches which support it. Finally, one out of the six Getting Real principles is only partly supported by the Agile Manifesto and other literature and researches.

Consequently, we can conclude that there are not so many new aspects in the Getting Real approach. The ideas included in the principles have existed already before in some form. How is it possible that so much attention have given to the Getting Real approach? We think there are many reasons. The company has proved in practice the effectiveness of the Getting Real approach by developing and commercialising successful products. The success of the small company has provided the people of the company with a strong charisma. The charismatic entrepreneurs have been very open and shared their knowledge with their growing audience. The books written by the key persons of the company [*Getting Real* (Fried and Heinemeier Hansson, 2006) and *Rework* (Fried and Heinemeier Hansson, 2010)] resemble the company's products; they are very light and easy to read and use ordinary language instead of jargon. We think the manner of representation has helped people to appreciate the thoughts of the 37signals people.

Even though there are not so many new aspects in the Getting Real principles, it can be said that the small company has picked up the right principles from all possible business and development rules and tenets for strengthening their activities. From a practical point of view, we can conclude that the researched company might be a good benchmarking case for other small business owners as well. From a theoretical point of view, we can conclude that there are no conflicts between professional literature and previous research, and the Getting Real approach.

Acknowledgements

Acknowledgements to Professor Samuli Saukkonen for the support and valuable discussions during the study.

References

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. (2002) *Agile Software Development Methods. Review and Analysis*, p.478, VTT Publications, Otamedia Oy, Espoo.

Abrahamsson, P., Warsta, J., Siponen, M. and Ronkainen, J. (2003) 'New directions on agile methods: a comparative analysis', *Proceedings of the International Conference on Software Engineering*, 3–5 May, Portland, pp.244–254.

Agile Manifesto (2011) 'Manifesto for agile software development', available at http://agilemanifesto.org/ (accessed on 13 April 2011).

Appleton, B. (2005) 'Brad Appleton's ACME blog', available at http://bradapp.blogspot.com/2005/02/there-is-no-code-that-is-more-flexible.html (accessed on 11 May 2011).

Avison, D.E. and Fitzgerald, G. (1991) *Information Systems Development*, Blackwell Scientific Publications, Oxford.

Beck, K. (1999) *Extreme Programming Explained: Embrace Change*, Addison-Wesley, Reading, MA.

Cockburn, A. (2002) *Agile Software Development*, Addison-Wesley, Boston.

Constantine, L.L. and Lockwood, L.A.D. (2002) 'Usage-centered engineering for web applications', *IEEE Distributed Systems Online*, Vol. 19, No. 2, pp.42–50.

De Souza, C.S. (1993) 'The semiotic engineering of user interface languages', *International Journal of Man Machine Studies*, Vol. 39, pp.753–773.

Fernandez, O. (2008) *The Rails Way*, Addison-Wesley, Upper Saddle River, NJ.

Fried, J. (2008) 'Jason Fried of 37signals at business of software 2008', available at http://37signals.com/speaks (accessed on 17 March 2011).

Fried, J. and Heinemeier Hansson, D. (2006) *Getting Real*, Chicago, available at http://gettingreal.37signals.com/toc.php (accessed on 10 January 2011).

Fried, J. and Heinemeier Hansson, D. (2010) *Rework*, Vermillion, London.

Grudin, J. (1989) 'The case against user interface consistency', *Communications of the ACM*, Vol. 32, No. 10, pp.1164–1173.

Grudin, J. and Gentner, D.R. (1990) 'Why good engineers (sometimes) create bad interfaces', *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Empowering People*, pp.277–282.

Heinemeier Hansson, D. (2009) 'David Heinemeier Hansson of 37signals at FOWA Dublin 2009', available at http://37signals.com/speaks (accessed on 5 May 2011).

Järvinen, P. (2004) *On Research Methods*, Opinpajan kirja, Tampere.

Kitchenham, B. (2004) 'Procedures for performing systematic reviews', Keele University Technical report.

Kunz, M., Dumke, R. and Schmietendorf, A. (2008) 'How to measure agile software development', *Lecture Notes in Computer Science*, Vol. 4895, pp.95–101.

Lindstrom, L. and Jeffries, R. (2004) 'Extreme programming and agile software development methodologies', *Information Systems Management*, Vol. 21, No. 3, pp.41–52.

MacCormack, A., Verganti, R. and Iansiti, M. (2001) 'Developing products on 'internet time': the anatomy of a flexible development process', *Management Science*, Vol. 47, No. 1, pp.133–150.

Miller, G.G. (2001) 'The characteristics of agile software processes', *Proceedings of the 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems*, pp.385–387.

Moore, G.A. (1991) *Crossing the Chasm*, HarperCollins Publishers, New York.

Müller, M. and Tichy, W. (2001) 'Case study: extreme programming in a university environment', *Proceedings of the 23rd International Conference on Software Engineering*, pp.537–544.

Nandhakumar, J. and Avison, D.E. (1999) 'The fiction of methodological development: a field study of information systems development', *Information Technology & People*, Vol. 12, No. 2, pp.176–191.

Nielsen, J. and Mack, R.L. (1994) *Usability Inspection Methods*, John Wiley & Sons, New York.

Palmer, S.R. and Felsing, J.M. (2002) *A Practical Guide to Feature-Driven Development*, Prentice-Hall, Upper Saddle River.

Park, A. (2008) 'The brash boys at 37signals will tell you: keep it simple, stupid', *Wired Magazine*, 16 March 2008, available at http://www.wired.com/techbiz/media/magazine/16-03/mf_signals?currentPage=1 (accessed on 23 May 2011).

Parnas, D.L. (1969) 'On the use of transition diagrams in the design of a user interface for an interactive computer system', *Proceedings of the 1969 24th National Conference*, pp.379–385.

Parnas, D.L. and Clements, P.C. (1986) 'A rational design process: how and why to fake it', *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 2, pp.251–257.

Power, B. and Reid, G.C. (2005) 'Flexibility, firm-specific turbulence and the performance of the long-lived small firm', *Review of Industrial Organization*, Vol. 26, No. 4, pp.415–443.

Rising, L. and Janoff, N.S. (2000) 'The Scrum software development process for small teams', *IEEE Software*, Vol. 17, No. 4, pp.26–32.

Singer, R. (2008) *Ryan Singer of 37signals at FOWD New York 2008*, available at http://37signals.com/speaks (accessed on 6 March 2011).

Strode, D. (2006) 'Agile methods: a comparative analysis', *Proceedings of the 19th Annual Conference of the National Advisory Committee on Computing Qualifications*, pp.257–264.

Torvalds, L. (2005) *Linux: Linus On Specifications*, available at http://kerneltrap.org/node/5725 (accessed on 5 May 2011).

Ulrich, T. and Eppinger, S.D. (2008) *Product Design and Development*, The McGraw-Hill Companies, Singapore.

Wild, W. (2008) *Agile Software-Development & Tools*, available at http://www.softnet2008.info/download/Wild.pdf (accessed on 12 May 2011).

Notes

1 The definition 'key person' refers to Jason Fried and/or David Heinemeier Hansson, who are the partners of 37signals and co-writers of the books *Getting Real* and *Rework*. The definition is used when it is not known which partner/co-writer to cite, or when citing both.

2 The internet version of the book *Getting Real* does not have page numbering. Citations to this book are in the form '(Fried and Heinemeier Hansson, 2006)'.