Accepted version of:
Twidale, M.B. & Nichols, D.M. (2013) Agile methods for agile universities, in Besley, T.A.C. and Peters, M.A. (eds.), *Re-imagining the Creative University for the 21st Century*. Sense Publishers. 27-48.

MICHAEL B. TWIDALE AND DAVID M. NICHOLS

AGILE METHODS FOR AGILE UNIVERSITIES

Abstract: We explore a term, *Agile*, that is being used in various workplace settings, including the management of universities. The term may have several related but slightly different meanings. Agile is often used in the context of facilitating more creative problem-solving and advocating for the adoption, design, tailoring and continual updating of more innovative organizational processes. We consider a particular set of meanings of the term from the world of software development. Agile methods were created to address certain problems with the software development process. Many of those problems have interesting analogues in the context of universities, so a reflection on agile methods may be a useful heuristic for generating ideas for enabling universities to be more creative.

INTRODUCTION

Universities are strange organizations. They are charged with multiple, perhaps contradictory, and certainly mutually complexifying missions. These include teaching, research, and service to local, national and international communities, economic regeneration and urban revitalization. They are expected to be memory institutions, preserving and passing on ancient truths, of telling people who we are and where we came from. But at the same time, universities are expected to be places of discovery, innovation and creativity. Scientific research is all about discovering and doing new things, but so too are the social sciences, the arts and the humanities. Innovation is disruptive and unsettling, challenging the old ways.

Universities have progressed for many years dealing with the balancing act of being both preservers of continuity and disruptive innovators. But every so often we should think about how these issues affect, or should affect, our own processes. A variety of current challenges relating to technological development, funding and international competition create a threat to traditional practices in universities--and a need to do things better, faster and cheaper.

In this thought piece we want to explore a term, *Agile*, that is being used in a number of workplace settings, including the management of universities. We explore a particular set of meanings of the term agile from the world of software development. Agile methods were created to address certain problems with the software development process. Many of these problems have analogues in the

context of universities and so may serve as inspiration for the development of analogous solutions. We do not have a magic bullet (Brooks, 1995) to offer as a solution. But we do believe that a reflection on agile methods can be a powerful heuristic for generating possible solutions. The guiding principle throughout this essay is the perhaps troublesome idea that an innovative research university should really be doing more research on itself to innovate new ways of operating.

A NEED FOR FASTER AND FLEXIBLE ORGANIZATIONAL PROCESSES

A number of books and articles have appeared recently noting the various challenges that universities have to confront (e.g. Christensen & Eyring, 2011). For many western universities this change includes greater global competition in research, expectations of higher standards from governments, increased comparative evaluation through both national and global university rankings, changes in funding sources (typically declines in government funding), limits to the possible growth of fees, and the potential of technological disruptions from growth of computer hardware, software and networking. Current interest in Massively Open Online Courses (MOOCs) is an example of such a possible disruption; we return to MOOCs later in the paper, using them as a case study for a more agile university.

It is always tempting to claim that the period that we are living through is special, and different from what is gone before. It can feel to those of us currently working in universities that there are particular pressures that did not apply in the past – ever greater expectations, more metrics, declining resources, greater competition, less well-educated students, etc. Regardless of whether the current moment is actually all that special, we would claim that there is a widespread pressure for universities to be ever more innovative in exploring new solutions and seizing opportunities. Unfortunately this innovation can at times feel to be impeded by a rather bureaucratic set of processes. We need to be more innovative in not just what universities do, but how they do it. Compared with small companies, and especially high technology start-ups, a university can seem rather slow in its ability to innovate. If start-ups are said to work on internet time then universities seem to work on ivy time - whereby their own organizational structures seem to stretch and warp how long things take to do.

It is in this context that the word ‘agile’ is often used as a desirable attribute (Elementa Leadership, 2012). It is more likely to be used as an aspiration rather than a description of what currently happens. That is, people may note that it would be very desirable if a university could be more agile in how it operates and reacts to changes in the environment. The irony is clear - universities are demonstrably successful in generating ideas and undertaking successful research to change the way we see the world, and to change what we can produce. Why is it so hard for universities to innovate their own processes?

Physician heal thyself: university research thyself

It would be desirable if a university could try things out really quickly. What if a university could do little experiments to see if a new way of doing things was better than the existing way? If only we had any skills in doing research. The irony is glaring although pointed out less than it might. How is it that an organization committed to world class research, which is held out as a beacon of innovation, can be so reluctant to do research on itself and especially to experiment with its own managerial practices? The idea might seem utopian, but the experience with adaptable agile methods in corporate settings makes us suspect that it might be possible in some form.

COLLOQUIAL AGILITY: FAST AND FLEXIBLE

Prior to a consideration of agile software development as a source of inspiration for the agile university, it is important to note that sometimes 'agile' is used in a more colloquial way. Agility as applied to a person carries connotations of flexibility and speed, often with aspects of balance. An agile person is less likely to fall over and can cope rapidly with both challenging and changing situations. When they do (rarely) stumble, they are less likely to injure themselves. These are qualities that we may well want to ascribe to organizations too. In this colloquial use of the term agile, it is used to describe what an organization has managed to achieve (such as seizing an opportunity, responding to a threat, quickly changing what it does or changing its internal processes in the light of circumstances, etc.).

The term agile may well be used negatively – to complain that our organization has failed to or is incapable of responding in a fast and flexible manner. It seems hard to object to this idea of agile – of course being fast and flexible is good (though we will revisit that later). People and organizations can be called less agile, but it is very rare to call a person or an organization "too agile". In this colloquial sense agile is an attribute. But little is said about what could or should be done in order to become more agile. Agile describes the outcome, but rarely does it tell us how we might get there. For people, it may involve various kinds of stretching exercises, but what should an organization do in order to become more agile? An email from university management saying "be more agile, right now!" is not enough.

One aspect of colloquial agility that does give a clue to one way to achieve agility is around size and age. Small organizations and new organizations are often able to be fast and flexible. Decisions can be made quickly because there are fewer people you need to ask, to tell, to persuade or to lobby. New organizations are typically small, and so gain this advantage solely by virtue of size. But new organizations have an additional advantage with respect to speed and flexibility - they lack precedent. In a new organization, the way you do things are new, and change may be less disruptive. At its simplest, it is less likely that someone will be able to say "but we've always done it that way". A new organization competing in a market may need to be deliberately different in order to stand out from more established competitors, creating a bias towards novelty and experimentation. Do these issues apply to organizations that happen to be universities? Are smaller

universities often faster and more flexible? What about newer universities? We are not sure, but it would be interesting to investigate.

AGILE SOFTWARE DEVELOPMENT

Software development is a complex, fraught activity. Many things can go wrong, leading to projects that are delivered late and over budget, fail to do what the customers want or need, or fail to be delivered at all. The research area of software engineering was developed to try and understand why this happened so often and to develop approaches for mitigation. Within software development, a variety of different practices were developed. One group (that included the methods of Extreme Programming, Adaptive Software Development, Crystal, and Scrum) had a certain set of characteristics that led to the development of a shared vision to articulate what they had in common - and indeed how they were different from other practices. The Agile Manifesto (Figure 1) was written in February 2001.

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:
– Individuals and interactions over processes and tools
– Working software over comprehensive documentation
– Customer collaboration over contract negotiation
– Responding to change over following a plan
That is, while there is value in the items on the right, we value the items on the left more.

Figure 1. The Agile Manifesto (http://agilemanifesto.org/)

The manifesto can be seen as a set of philosophical values. It is clearly not a recipe for exactly how one should do software development. But the various methods that were determined as being agile had those characteristics in common, as did their subsequent refinements. These methods are often contrasted with other methods that emphasize the items on the right of the manifesto to a far greater extent. For agile advocates, those elements may be carried to extremes resulting in overly bureaucratic plan-driven (and non-agile) development.

In addition to the manifesto, twelve principles underlying it were also articulated (Figure 2).

We follow these principles:

- 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.

Figure 2. Principles behind the Agile Manifesto (http://agilemanifesto.org/principles.html)

Since 2001 agile methods have been adopted by many software development teams around the world. There has been an accompanying research interest in agile, trying to understand whether it works, if it is in fact more efficient than other methods, and if so, why. It has inspired a substantial literature of books describing detailed processes derived from it, case studies, how-to advice, empirical evaluations, training materials, applications in other contexts, and reflections on how to introduce the ideas into organizations that have pre-established processes and may have individuals and whole groups who are very skeptical about the idea. In a recent review of the literature, (Dingsøyr *et al*., 2012) found 1551 research papers from 63 countries on agile software development in Web of Science published between 2001 and 2010 (inclusive).

Although not the dominant form of software development, agile methods are now a well-established niche with strong empirical evidence of success. The different methods can however seem somewhat cult-like to outsiders with partisan claims around efficiency and effectiveness.

Agile methods emerged as a reaction to sets of processes developed to try and address the difficulties of software development by very careful precise specification, planning and documentation of what to do in advance. Then the code is written, tested and then deployed in the customer's setting. This very logical, rational process is sometimes called the Waterfall method (Royce, 1970). Such an approach can seem eminently reasonable. It is in part inspired by processes that have proved to be highly effective in mass production, and in construction. But the development of something as logical as software seems to be strangely resistant to excessively logical and rational methods that try to plan everything in advance and then to systematically execute each element in a logical order. This might be because our understanding of software development is underdeveloped. Or, as agile advocates often claim, it could be that there is something fundamentally different about software development.

Many of the problems arise around the issue of requirements capture - determining what it is exactly that the client wants: "Traditional approaches assumed that if we just tried hard enough, we could anticipate the complete set of requirements early and reduce cost by eliminating change" (Highsmith & Cockburn, 2001). Various document-centric methods were developed in reaction to unsuccessful software projects that resulted in dissatisfied clients, breakdowns in trust and communication, and indeed lawsuits. If the developers can show that they have delivered exactly what the clients asked for, by referring to a voluminous requirements document, then surely the client has no reason to complain, or indeed to sue. The problem is that the client may not know exactly what they want, or what they want may change before the product is delivered. That is not because the client is confused or naive, just that interactive software products are immensely difficult to think about - even for skilled software developers.

It should be noted that many software engineers regard the waterfall method as something of a straw man. Different software development methods are often explained and justified in contrast to this hypothetical waterfall method - including methods that are not agile. However, in the literature on agile methods 'waterfall' is often used as a catch-all term for all non-agile methods that although not as linear and rigid as pure waterfall, are far less flexible and adaptive than agile. We will use the terms traditional software development and plan-driven development to refer to these non-agile methods.

Agile methods seem to work by acknowledging human fallibilities - the difficulties that clients have in knowing what they want and articulating it, the difficulties that developers have in completely understanding those wants and needs, the errors that inevitably arise in software development, and our inability to predict future needs. The manifesto proposes that the way to address to all these problems is to focus on tight iteration loops and different kinds of rapid testing and evaluation. Particular methods vary in exactly how they achieve this, but they all focus on building and testing minimal versions of the desired product very quickly and then progressively adding more features over time. This is a more organic kind of growth process (like a tree that starts off as a seedling) than say a typical construction project (where we do not begin with a tiny shed and grow it into a

mansion). The result is that at all stages the client has a product that at least does something even if it does not do everything desired. Rather than trying to plan everything correctly in advance, the methods allow for much more rapid adjustment on the fly in the light of inevitable human error and externally changing circumstances.

Agile methods seem to be especially effective in novel design settings, where developers and clients may not be exactly sure what the best software solution is, or indeed what is really needed from the software in order to do the job. The focus on early delivery of working software (versions that successfully execute just a few of the features of the envisaged final product) allows for different kinds of testing and revision of the requirements, allowing for fast and flexible response to a rapidly changing world - or indeed participants' rapidly changing understanding of the world.

Nevertheless, agile can seem a very alien way of working, and switching to agile is not a trivial matter. It feels good to have a clearly worked out plan to follow. It feels like good management to begin by working on such a detailed plan. Agile is not about an anarchic free for all. But it emphasizes that plans will inevitably have to change as circumstances dictate (Suchman, 1987), and so detailed upfront planning may not be the most efficient way of working. Rather what is needed are ways to dynamically re-plan, but in a systematic manner.

> Planning is one of the most difficult concepts for engineers and managers to re-examine. For those raised on the science of reductionism (reducing everything to its component parts) and the near-religious belief that careful planning followed by rigorous engineering execution produces the desired results (we are in control), the idea that there is no way to "do it right the first time" remains foreign. (Highsmith, 2002)

AGILE AS A METAMETHOD

The substantial literature on agile methods can be rather challenging to read. It can seem slippery in what it actually advocates. This is in part because although it talks a lot about methods, it is really much more focussed on how to design methods, and indeed how to create a setting where methods themselves are continually being redesigned and improved to meet the demand of local circumstances. Highsmith & Cockburn (2001) describe agile as using generative rules: "a minimum set of things you must do under all situations to generate appropriate practices for special situations."

This abstraction is why we believe it can be applied to university settings. It operates through a process of first articulating values that lead to principles and thence to the development of particular practices (Beck, 2005, p. 15). Testing and review does not just apply to the outputs (the software produced), but also to these practices. These practices are systematically reviewed and refined as a team learns more about what it does, and how it can change its practices in order to do things better.

As an illustration of values informing method design, consider the first value in the manifesto:

– Individuals and interactions over processes and tools

Cockburn and Highsmith (2001) note: "it's not that organizations that employ rigorous processes value people less that agile ones, it's that they view people, and how to improve their performance differently. Rigorous processes are designed to standardize people to the organization, while agile processes are designed to capitalize on each individual and each team's unique strengths." This value in concert with the other three and the twelve principles leads to practices such as pair programming (two developers sitting side by side at a computer working on a single task) and an emphasis on informal communication and consensus-building; but also techniques to ensure that conversations and meetings do not go on forever, and decisions are made quickly. It also leads to approaches to how teams should be managed: "However, "politics trump people." Even good people can be kept from accomplishing the job by inadequate support" (Cockburn & Highsmith, 2001). A substantial part of an agile team-leader's role is identifying and removing barriers to a team being able to do its job.

Although the agile approach criticizes an excessive focus on documentation, the processes developed do allow teams to track their progress and indeed their rate of progress (often termed 'velocity'). Public displays, known as 'information radiators', enable a team to see how they are progressing in producing working software that accomplishes an increasing number of desired features. The aim is to work towards a constant sustainable velocity as teams learn to more accurately estimate the costs of developing each component of a project and can thereby reliably deliver working products while also being able to dynamically adjust requirements by re-prioritising the task list. This information on processes is obtained as a by-product of actually doing the work, rather than additionally documenting what is to be and what has been done. The process information allows teams to periodically reflect on their processes and to revise them to further increase productivity and minimize errors.

APPLYING AGILE SOFTWARE DEVELOPMENT APPROACHES TO PROCESSES IN A UNIVERSITY

Universities are not (centrally) about developing software, so it is unlikely that we can just apply a set of methods from one setting into this wholly other setting. However, universities do face analogous kinds of complex problems and so some of the underlying philosophies may be useful as a means to develop analogous processes. In particular, universities of course have to deal with a rapidly changing world. For many western universities this change includes greater global competition in research, expectations of higher standards from governments, changes in funding sources (typically declines in central funding), limits to the possible growth of fees, etc. (e.g. Christensen & Eyring, 2011).

Coupled with these external challenges, many academics feel that their internal administrative processes tend to hinder rather than to enhance progress. Bureaucratization of processes typically looks much more like a waterfall method than an agile method. New initiatives have to ripple down through many layers of approval, and documentation can appear greater than that the real work that the documentation is intended to support. Indeed it is easy to begin seeing the documentation as the actual real work. Is it possible to make some of a university's processes agile? Is it desirable, and is it effective, assuming we can agree on what counts as effective? Universities often address ideas in a very careful, analytic and systematic way. That has many virtues. We do not want to waste money, and in particular we want to be careful about creating a series of costly ongoing commitments or precedents. However the deliberation process can also be perceived to be extremely slow, perhaps unnecessarily so. We believe that it is worth investigating if there are ways to develop alternative processes that are faster and more flexible, and yet can still deliver useful results while avoiding waste.

Nevertheless, all those university rules, processes, procedures, approval levels etc. were created for a reason. They are there because of real concerns. The same applies in software development. Agile is not anarchism, it does not claim that just because these rules, documents, etc. can slow things down that we can and should abolish them, and then everyone will be able to get their real jobs done much faster and more flexibly. Rather it acknowledges the problems these structures were developed to mitigate and proposes different ways to mitigate these same problems that also allow greater speed and flexibility, acknowledging human fallibility (Highsmith, 2002).

We have noted that the reaction to the software development crisis was an understandable inclination to try and systematize software development by greater documentation and oversight. There are remarkably similar pressures in universities for documented accountability, both internal and external. Some of these are very difficult to ignore or to change - they may have the force of law or contracts behind them. The agile manifesto does not reject clear plans, contracts, documentation and processes (those items on the right of Figure 1). Rather it claims the greater importance of other aspects (the items on the left). For example, a military software development contract may have documentary requirements that seem onerous and inefficient to an advocate of agile methods, but there may be no opportunity to ignore or change them. The same applies with certain processes at a university that may be mandated by laws or by a contract with a funding body. Nevertheless, there may be some room for creative manoeuvre even with parts of these, and certainly with those processes, rules and documentary requirements that originate within the university itself.

The radical, disruptive and innovative approach of agile is to question if those rules are really strictly necessary, or indeed desirable, and to design and propose alternative processes that can be tried - and tested - to see if they are actually better. Some of the activities that universities do are large, complex and have difficult to understand interrelationships. Existing structures have been developed to provide checks for effectiveness and unintended consequences. We see many similarities with large complex software development projects that may contain bugs and unanticipated interactions that need to be addressed. Given that an agile approach

We are uncovering better ways of developing **students** by doing it and helping others do it.
Through this work we have come to value:
- Individuals and interactions over processes and tools
- Demonstrable **student achievements** over comprehensive documentation
- **Dynamic learning discussions with students, (as well as parents, government employers and other stakeholders)** over **documents, metrics and policies**
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Figure 3. A first draft at an agile manifesto for a university's teaching mission

has been found to be an effective way of tackling software complexity, maybe an analogously agile approach can be developed to tackle some of the problems that universities face in their development of teaching, research, outreach and other activities.

CONSIDERING RESEARCH

In many respects, at certain levels of analysis, the way that universities handle the management of research is actually reasonably agile. Researchers are typically empowered to try and pursue external and internal funding by writing grant proposals. Universities provide centralized shared resources such as libraries, central purchasing and account administration. Great efforts are taken to support the acquisition of grants and to not hinder this process. Well-run universities treat this active support of, and non-interference in, the research processes themselves as a critical managerial function. Small, fast and flexible research teams are able to seize opportunities of new discoveries by themselves or others as well as explore funding opportunities. There is often strong encouragement and explicit support for writing grant proposals and for the creation of spin-off companies based on the findings of research.

However, this agility may not apply at all levels of analysis, and so there are opportunities for improvement. No doubt grant holders and the managers of laboratories can identify various examples of non-agility. There may even be recurrent patterns in the ways that similar extant processes at many universities slow down the research process by imposing requirements that researchers perceive as distractions or burdensome hurdles. Seemingly petty rules about travel reimbursement immediately spring to mind. The agile challenge is to try and design processes that achieve the (perfectly legitimate) aims of the current rules, but in a more efficient manner that is in better alignment with the core values. But overall, we suspect that it is possible to find many exemplars of agility and process innovation in the supporting of research. Having examples of agile process innovation from within the same institution can be helpful in showing that these kinds of innovation are possible in that particular institutional context.

Revisiting the Agile manifesto of Figure 1, note that it is very short and written in clear language. It describes a set of aspirations, perhaps an underlying philosophy. How might this manifesto, written for the context of software development, be rewritten for the context of a university? It is not a simple matter, because there is no single clear agreed upon activity that a university does. We suggest one example of applying agile here (Figure 3), focused just on the teaching mission of a university - as a provocation. We hope that it might inspire the reader to come up with a better example, maybe tailored to her university and within that institution, to an area that could benefit from a more agile approach.

This manifesto focuses on the teaching mission of the university. Even the act of rewriting the manifesto can force a degree of reflection and raise interesting questions about what it is we actually do, and what it is that we actually want to do when we teach:

- What do we actually aim to produce as output?
- What would we want to measure, assuming that it is possible or feasible?
- Who are our "customers"? The students, or others who contribute to paying the bills: parents and the government? What about employers? The local community and its economy? Society? The country?
- Who should have an influence in what we teach?
- Who should have an influence in how we teach?
- Is something important lost when we even try to equate software development with student development?

"Developing students" is a deliberately provocative rewrite of "developing software" in the original agile manifesto. An alternative, perhaps closer to the nature of the software development task might be "developing learning experiences". Like software, these can be difficult to develop, the process can be inefficient, as can the learning that they are intended to achieve, and they can certainly be buggy or error prone in failing to achieve the desired outcomes.

The thinking behind the agile manifesto reminds software developers that just focusing on software that works, although very important, is insufficient. Yes indeed, the software needs to work, but it also needs to do what the customer actually needs. Approaches that address the challenges of software development (processes, tools, documentation and plans) can be valuable, but carry the risk of becoming the main focus of attention rather than producing working software that does what the customer needs. These development foci can also distract developers from the reality that the customer's needs may be evolving. The agile approach tries to help developers - and the practices that they create - to stay on track.

Similarly, we don't (or rather we should not be tempted to) just create courses, syllabi, lectures, assignments, learning experiences etc. as ends in themselves. What should matter as a central concern is the impact that they have on our students as they engage with them. How much learning happens? Are we able to handle evolving learning needs?

Figure 2 shows the 12 Principles behind the Agile Manifesto. This is the next stage elaboration of the agile approach. There is still little exact detail of what you might do in an agile software development process, but there are indications of the kinds of activities you might expect to see. The principles are also articulated to contrast with some of the features, or consequences, of traditional software development processes. For example, consider the second principle, to "Welcome changing requirements, even late in development." Changing requirements are traditionally rarely welcomed. They are disruptive, can render prior work wasted, cause delays and complexities and often lead to bad feelings between customers and developers because of a lack of understanding of exactly what is being requested and how difficult it is to provide.

Just as with the manifesto, it can be an interesting exercise to try and translate these principles to a university context. Immediately in principle 1 we revisit the challenge of "who is our customer?" If we decide to focus on the student, then we have a thought-provoking idea of satisfying them "through early and continuous delivery of valuable learning experiences". This may not be too controversial in the abstract, although on reflection some of us may wonder if we have ever considered course design from this perspective. There are some courses where students can feel frustrated at all the rather tedious prerequisite knowledge and skills that they have to master before they can get to the concepts that they care about. Similarly certain courses may only come together and make sense right at the end. These require trust on the part of the student that all the effort will be worthwhile. Where possible it is certainly pedagogically desirable if the student feels that they are making clearly observable progress and accumulating skills or insights that they deem valuable as they go. So we might ask ourselves what, if anything, we should do if our students do not regard the learning experiences that we deliver as "valuable" but rather as arbitrary points that must be accumulated to gain the prize of a certificate. It is certainly sobering to ponder this question.

To take another example, consider principle two:

– Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

This might be adapted to:

– Welcome changing **learning needs, or (syllabi),** even late in the **semester**. Agile processes harness change for the **student's personal development**.

This principle could be a mixture of the obvious and the incendiary. It hardly needs an article on the agile university to note that it can be a good idea for university professors to introduce topical issues into their lectures in order to illustrate the power and applicability of abstractions and theories to give analytic purchase on contemporary problems. Good teachers have always seized teachable moments. Great teachers may discover (to their surprise and disappointment) part way through a semester that a substantial proportion of the class lack a certain set

of prerequisite knowledge or skills, or harbour a major misconception. That changes the requirements of the learning and the teacher re-plans accordingly. Again, such flexibility already occurs in many (but by no means all) classes. This re-planning can look very problematic to a standards-based approach which believes that good teaching only comes from careful lesson plans and sticking to the syllabus. Changing the syllabus as you go seems foolhardy as well as impossibly expensive in time and effort. Do we really want to welcome changing the syllabus as you go?

Again, we have to ask ourselves who our customer actually is. The example above assumed it is (just) the student. But maybe the real customers are an agglomeration of disagreeing stakeholders including students, the government, society, parents, future employers, and accreditation bodies. If so, we may realize that we have multiple kinds of customers all at once. Suddenly it becomes a little clearer why universities have major problems with focus and prioritization, let alone speed and flexibility.

Taking the problem back to agile software development, developers may successfully interact with a single customer representative and be able to welcome changing requirements from her. However, if the team has to deal with an array of customer stakeholders with very different views of what the software is for, then there may be rather too many changing requirements to welcome.

Agile values working software over comprehensive documentation; such as a syllabus, lesson plans, learning outcomes, rubrics etc. As the manifesto states, this documentation is not worthless; its value is recognized. It is just that something else is valued more, namely working software - which all this documentation is created to facilitate. Similar issues arise with documentation around teaching. However, in addition to any disagreements about what exactly is being taught, we also have the challenge of multiple stakeholders, including those who only or mainly focus on the documentation, not the real 'product' - our 'developed students'. Imagine how a non-agile accreditation body would react to a professor who said: "Yes I know it says I would teach that in the syllabus, but by being more customer-centric with my students I decided very late in the semester to teach them something else that they wanted and needed to learn more".

Finally, we wish to note the emphasis in the principles on sustainable development. This is a reaction to software projects that hit deadlines and then expect developers to work excessive overtime in order to meet the deadline. The claim is that agile methods allow for a much more sustained and sustainable process where people are less likely to be exhausted (and thereby make mistakes) and also to be subject to burnout, quitting the job and thereby losing substantial personal and organizational investments in skill development. Certain agile methods include the bizarre notion of the 40 hour work week (Glass, 2001). This is another dangerously radical and controversial health-promoting idea that we would like to bring to the university.

COUNTEREXAMPLE: SPECIAL TOPICS SEMINARS ARE ALREADY (SOMETIMES) AGILE

Although university processes may veer more to the "comprehensive documentation" side of the agile description, there are counter examples and precedents. We claim that an agile university is a radical notion, but innovative universities always contain pockets of agility that can serve as precedent and reinforce that the very idea is by no means alien. In the context of course design, many universities have a special case for a faster and more flexible approach, one that may more easily evolve as it progresses, often through necessity or the seizing of opportunity – classic drivers of agility.

One example is the "special topics" lecture course. This is a course that acts as a placeholder for a variety of different courses. It is widespread at the graduate level, but there may be undergraduate versions as well. They allow for one-off teaching of a particular topic, typically an advanced one. This might be by a visiting researcher, or by a faculty member trying out something new, or who perhaps is in the process of writing a book and wanting to test it out chapter by chapter on a student audience. Such a special topics course has many of the attributes of agility. It may be next to impossible to fully specify in advance – because the requirements are still in flux. However, these may be rather un-agile instructor-centric requirements considered in terms of what the instructor wants to cover. The experimental nature of the course is flagged by its official name. The instructor may further note its experimental nature and solicit feedback to enable greater interaction and responsiveness. Students will need to expect that things may well go wrong pedagogically, but that is the price they pay for encountering something exciting and novel. The course is usually optional, creating a greater acceptability for the inevitable uncertainty of outcomes.

After a series of (often rather informal and impressionistic) iterative testing of the components over the semester, the course may be abandoned, turned into a book, or revised into a more traditional, more fully documented course. As such it has another component of agile software development that is sometimes missing in university settings - a defined end point. This is significant because one of the powers and problems with universities is the issue of continuity. A software project is delivered and done. Yes it may be revised, but this is treated as a new project. By contrast, traditional courses are taught many times over many years and other aspects of universities like departments, libraries, research institutes and centers are typically expected to persist for years, decades, even centuries. This very persistence can be one reason why university structures have been set up to be rather slow, un-agile and rather risk averse. Unlike software developers, they are not mostly dealing with a set of complex one-off projects that must be delivered before moving to the next one. Rather they have to deal with structures that are assumed to persist for many years and that can well persist longer than is desired. Creating something new can imply a continued commitment to maintaining it and so creation is a matter to be treated with extreme caution.

Even conventional courses have a small aspect of agile development in how they operate. In traditional software development, the requirements are fixed in advance, and the time and resources to complete the project are estimated.

Unfortunately these estimates are very frequently wrong and overwhelmingly are under- rather than over-estimates. Of course it would be nice to fix everything in advance, but we are fallible human beings. Agile recognizes that fallibility but suggests fixing the time and resources available in advance, and estimating the features that will be delivered. Then when, sadly, problems arise, a product will be delivered, on time and on budget, but not necessarily doing everything the client may want. The features it does successfully deliver have been the result of a process of reprioritization negotiations to maximize the utility of what can be done with fixed resources. This approach is called timeboxing (Highsmith, 2002). Note that this is actually how we teach. Teaching is timeboxed at the level of the semester and the lecture. Despite frequent problems of students lacking prerequisite knowledge, struggling to understand what the instructor thought obvious, 'bugs' in pedagogy etc., it is extremely rare for a course to overrun the semester or to exceed its budget. Instead, time and budget are kept fixed and instructors change what they planned to cover, in what depth and in what form. Whether this is an optimal or negotiated reprioritization as occurs in agile methods is quite another matter, but again we note that several agile elements can already be found in universities.

THE PROBLEMS TO BE TACKLED: INERTIAL DAMPENERS OF INNOVATION

There can be a number of reasons, some very laudable, why a university may not be very agile. We consider these to be the inertial dampeners of innovation. Understanding what these might be is useful in appreciating what a more agile approach will need to address, and indeed the likely opposition to agile adoption. We list a few inertial dampeners here, but do not claim to have the complete set. As a simple first example, a notable feature of agile groups is the use of fluid role definitions (Beck, 2005). This is something that universities can be rather inflexible about.

A university is a memory institution that may consider itself as a preserver of tradition. Large size and the age of an institution can have associated features that can slow down innovation. Much of the activity is about managing flows rather than products. Structures that are created may create commitments to continued preservation and may be very difficult to close down, creating a disincentive to risk creating new ones. Many universities have a consensus-based, collegial management structure that means that a lot of people have to be consulted before a decision can be made, slowing the process considerably. This inertial dampener is especially odd in the context of agile, because at the micro level of the agile software development team, consensus-based decision making is actually a core component of agile and one that is contrasted with other more managerially top-down development methods.

There are also issues around risk management, and cultures of risk taking and risk aversion. Universities may be culturally rather risk averse at the level of management and institutional structures. Although exhorted to be more entrepreneurial by many politicians and commentators, these same groups would not doubt be equally condemnatory if the university gambled and lost a substantial fraction of its assets in high risk venture capital deals. Entrepreneurial risk taking

sounds exciting, but it rather depends upon what the consequences are of failure. Public universities may be required to provide greater access to information about everything that they do, successful and unsuccessful, and this can interact with a public feeling a sense of ownership of and interest in everything that occurs. This is inevitable as universities try hard to make themselves seem relevant and part of a larger community activity.

The fear of being in the spotlight or becoming a political football may reinforce risk aversion at the higher levels of a university. Scandals typically involve something that was done and that the university failed to prevent. Unfortunately there is less outrage about university structures that render an innovation infeasible or make it too slow to be effective.

This risk aversion, although both unfortunate and understandable is truly odd because at a lower level of analysis universities are collectively renowned for being hotbeds of controversial ideas. Professors (and often students) are constantly challenging the status quo and saying things that annoy powerful interest groups. On the whole, university managements are commendably aggressive at protecting this freedom of inquiry and expression. The tenure system at US universities was set up precisely to protect the undertaking of controversial scholarship (Amacher, 2004). So certain risks of controversy are embraced by universities even as others are feared. This needs further examination to truly understand and to consider how we might design structures to move along parts of the risk curve.

Agile software development also deals with risk. A poorly designed project delivered late and over budget seriously damages the reputation of the supplier. Bugs can be not only annoying and frequently expensive in their consequences, but in the case of safety critical systems positively dangerous. The agile approach deals with this risk by many iterations and an almost obsessive focus on testing. For example in some techniques, the automated test suite is built before the software it is going to test, so it immediately initially fails. Agile methods aim to lower the consequences of risks by failing fast in order to discover problems (bugs and changed requirements) early. Clear methods to identify and recover from problems as part of the design process replace all the heavy duty upfront checking and validation processes of less agile methods. This does however create a barrier to adoption - the need to convince all interested parties that you are replacing one kind of oversight with a different kind, and the agile one is actually at least as effective in identifying and fixing problems in order to minimize overall risk. There is much talk in the literature of the challenges of making the case for agile, and processes for incrementally introducing the techniques into an organization. Agile advocates also note the costs and lost productivity of keeping traditional checking methods alongside the new agile methods that should render them obsolete.

TIME AND TEMPORAL SCALING

Universities operate on many different timescales all at once. Together, these may not fit well with the iterative build cycles of software development, and so create certain barriers to flexibility, speed and agility in the colloquial sense. This means that we will need to think how to adapt agile software development insights to the

constraints imposed by timescales. Examples of cycles include: the 50 minute lecture, the weekly teaching cycle, the semester or term, the academic year, and the 3-4 year undergraduate degree. Additionally universities have long term perspectives of several years for a given course or degree option and many years (decades, even centuries) for departments, schools, institutes, centres, etc. Finally, like most organizations, universities have to handle external shocks and opportunities such as changes in government policies affecting them, funding opportunities, the economic cycle and changes in the economy's demand for certain kinds of skills, professions and accreditations.

A company practising agile software development also operates on multiple temporal scales. Indeed the very short (sometimes 1-2 weeks) build cycles or scrums of certain methods are a distinctive feature of agile. But there is a sense of working through sequences of projects, and within a project, pulling an item off the backlog, working on it, delivering it and moving to the next item. This creates a linear feel, whereas by contrast, much that a university does can look much more cyclical than linear. This is in part just a matter of the level of analysis one chooses - for an individual student we may (hope to) see a linear progression of increasing knowledge, understanding, skills, personal development etc. Whereas for the institution as a whole, each year a new set of 18 year olds arrive and we start all over again. Dealing with the cyclical and linear aspects will be a challenge.

This agile approach can seem rather short-termist to a memory institution such as a university. University leaders have to worry about legacy and the financial sustainability of activities - in particular whether they entail ongoing commitments. Those very legitimate concerns lead to multiple levels of review and the creation of checks and balances. As a result activities such as creating a new research center or a new degree can understandably be rather slow. The challenge that agility raises is to ask whether it has to be as slow as it currently is, and what is possible to change to increase speed and flexibility without re-introducing major problems. One possibility is to have something like a special fast-track (agile-track) for activities with defined time-limits, unable to create ongoing commitments. These are more in the linear than the cyclical category outlined above. Precedents already exist - special topics courses and research projects are treated as one-offs. But we must recognize the tension inherent in a university proclaiming its commitment both to legacy and long-termism and also to innovation. The challenge seems to be about making it easier to discard in order to grow elsewhere:

> agile enterprises pursue a series of temporary competitive advantages—capitalizing for a time on the strength of an idea, product, or service then readily discarding it when no longer tenable (Stacey, 2006)

One may hope that new activities will ensue, but they do not need the careful review that creating say a new department or centre needs. That model might be extended to create other kinds of time-limited (linear) activities.

There is another temporal factor that may be problematic for agility. It could be that the very attributes that the general public, students, alumni and other stakeholders admire about a given university: traditions, heritage, buildings, schools, departments, famous alumni etc. are at the same time inhibitors of certain kinds of innovation. That pride creates interest in what the university does,

deliberately encouraged by development offices to increase donations, and political support for spending public money. But that interest can mean greater visibility of both successes and failures. If the perceived cost of failure is greater than the perceived benefits of success, we end up with a risk-averse culture.

An extreme version of agility (not one we espouse) would allow for no sense of history, precedent, tradition or indeed security. One could imagine a university run in a more corporate manner where subunits (departments, institutes, degrees, etc.) are created rapidly because they can be disbanded equally rapidly when no longer essential, or simply when the opportunity cost is too high. This would move the university into a realm of Schumpeterian creative destruction. It may be very reactive, but also rather stressful for employees who may fear losing their jobs. Currently many faculty and university employees invest a lot of time, effort, care and indeed emotion in their units, such that disassembly and reassembly can seem traumatic in a way that would not apply in a factory or a software development company where people were regularly reassigned. For many, this kind of extreme agility is the apotheosis of the creeping corporatization of the university. As such it is something to be critiqued (Gillies, 2011) or even actively opposed, chiefly because it leads to the loss of many virtues seen in the traditional liberal university. This raises a problem for us as advocates of some agility (but not this extreme form). Our version of agility might be perceived as a Trojan horse for university corporatization, and so something to be opposed in that light. These concerns need to be understood and aired. We hope they are ill-founded, but they are certainly understandable. We would note that the literature on agile software development has substantial evidence of the way that effective agile teams are necessarily groups of empowered professionals, and display high levels of job satisfaction and a strong sense of autonomy. There is no guarantee that what works for software developers would also work for university faculty, but in the ethos of agile, we believe that it is well worth experimenting to find out.

MOOC DEVELOPMENT AS A CHALLENGE TO A UNIVERSITY'S AGILITY

Over the past year or so, a particular kind of teaching experiment has emerged that has been hailed by many to have the potential to be a radical disruption to the traditional operation and funding model of universities. MOOCs have generated both publicity and criticism (Daniel, 2012), with some worried they may even threaten the success of physical universities.

Although there were earlier MOOCs, the Stanford artificial intelligence class was particularly influential; a description of the interactions around this class illustrates the tension between existing organisational structures and the new challenges of the MOOC:

> A few days later the class had 10,000. That's when the Stanford administration called. Thrun had neglected to tell them about his plan—he'd had a hunch it might not go over well. The university's chief complaint: You cannot issue an official certificate of any kind. Over the next few weeks, 15 meetings were held on the matter. Thrun talked to the dean's office, the registrar, and the university's legal department. Meanwhile, enrollment in

> CS221 was ballooning: 14,000, 18,000, and—just two weeks later—58,000. In all those meetings, not one person objected to Thrun's offering his class online for free. They admired his vision. The administration simply wanted Thrun to drop the assignments and certificate. He refused. Those two components, he argued, were responsible for driving the sign-ups. Someone proposed removing Stanford's name from the course website altogether. (Leckart, 2012)

The challenge to the organisation was how to react to a new form of course that had not been through familiar procedures. Any kind of novel course design looks very like the waterfall method. Typically a course is carefully planned in advance, with substantial documentation. This course proposal then has to be reviewed by various committees, as a way of achieving quality assurance and in order to check for undesirable interactions with existing activities. Eventually the course makes its way through the approval process and it can now be taught. There is typically far less and much lighter ongoing monitoring of the course, although there may be periodic reviews. If the course is especially innovative and consequently does not fit the patterns of previous courses that have moved through the approval process, the processes themselves may not be able to cope, creating the need for new sub-processes, the creation of exceptions and fears about precedents. In this way a perfectly understandable approval process can be a barrier to innovation.

Redesigning course approval processes to make them more agile would involve looking at the agile manifesto and 12 principles for inspiration. It would involve considering whether the upfront work (the attempt to plan as much as possible in advance) could be changed to a more iterative and responsive monitoring; checking and testing. It will be challenging to design a process that is time-shifted in this way, checks for the things that actually matter, and is at least no more administratively burdensome than the current processes. In turn this means that the design of new (agile) processes should be given the care and status that software design is given. It should not be a matter of a Dean making something up on the fly.

It is interesting to observe that much of the MOOC activity is currently taking place through start-ups (e.g. Coursera and Udacity) that take much of the organisational burden away from Universities themselves. The rapid growth of Coursera is characteristic of the Internet-time approach they have taken, with rapid experimentation and at least one "failure" (Jaschik, 2013). It remains to be seen if this can best be understood as a corporate outsourcing of agility, or as a creation of a safe space for experimentation deliberately excluded from traditional administrative structures.

PROCESSES OF AGILE INNOVATION AND ADOPTION IN UNIVERSITY OPERATIONS

There is substantial evidence that agile methods improve efficiency in software development. In this article we have made the case that there are some similarities between the development of novel software and university activities. We have also noted the existence of pockets of agility within universities to emphasize that the

ideas are not completely alien to this context. So it seems worth experimenting with applying agile methods in a university. Unfortunately we can't just copy the agile techniques that have been developed because they are aimed at supporting software development. So we will have to adapt techniques and combine them with new ones that we create, inspired by the agile approach. We don't have a set of techniques ready-made and tested to offer the community. Rather we want to encourage many people to design and test different approaches so that we can discover what works best.

The agile literature has much to offer as inspiration, including various processes, how to manage agile projects, and how agile management is different. Additionally, there are case studies on how software development companies have tried to move agile methods from their software development activities to other parts of their operation and on the challenges of introducing agile methods into an organization and overcoming perfectly understandable skepticism.

On the last point, the unsurprising consensus is to start with a pilot project, treat it as an experiment and collect a lot of data to provide evidence for improvement over time. It may require a number of projects before a team learns how to operate in an agile manner, so early results many not be spectacular. A key point is to have management support. An agile project will need permission to not use existing organizational processes as it deploys its own processes instead. Using both processes will most likely mean it is very difficult to show any improvement. This 'permission to be different' can be easiest to grant in a project far outside normal operation or one that is clearly experimental. It will need some demarcation from other normal operations, perhaps by analogy with corporate 'skunkworks' or the special economic zones set up in communist China by Deng Xiaoping to explore alternate more capitalist modes of production. We think the latter analogy is rather apposite, but perhaps is not the most expedient one to use in making the case to university administrators.

CONCLUSION

Our aim is to provoke reflection on how things are done in universities - mostly because we happen to work in them. Similar challenges apply in many other kinds of organization, both for-profit and non-profit. Software development is a very particular kind of complex collaborative activity requiring peculiar combinations of creativity, analytic rigour and deep understanding of both what people do and what they say that they want to do. This very complexity is why we believe that the methods developed around agile software development can serve as an inspiration for the development of methods to address the many challenges of an innovative research university. We believe that the first stage of this is for research universities to more explicitly apply their considerable research skills to analyzing, improving - and experimenting with - their own managerial practices.

If we want our universities to be more innovative, responsive and adaptable - to be more colloquially agile, then we need to examine the barriers to agility. It might be nice to simply abolish these barriers, but the processes, documentation, etc. that act as barriers were usually created to address real problems. Therefore we need to design new processes that are more agile. This design activity itself will require

innovation and agility. It requires analysis of what is done now and why it is done, and mixtures of creativity and engineering design pragmatics to develop new processes. Those who extol the value of greater innovation rarely include the importance of innovating and experimenting with our administrative and managerial processes. That is precisely what we are advocating. Taking an agile analytic lens allows for the questioning of what we do and why we do it. It encourages us to ask how we could redesign any single process in several different ways to make it more agile, and then how we might try out these ways, compare them and learn from them

In line with agile thinking, we also caution against hubris. This is not about one big bout of careful analytic research followed by the development and deployment of an ultimate University Administration Process Design Solution. Agile methods are deliberately designed with human frailties in mind. We get things wrong. Our best guesses are wrong, Even if we were right in our diagnosis of the problem, the world changes and we should now be working on solving a different problem. The art is not to get it right, but to fail fast, to be able to test innovations as early in the design process as possible, and be able and willing to re-prioritize and replan as more is learned. Consequently, redesigning processes to enable a university to operate in a more agile manner needs to have these same aspects of seeking early feedback, constantly iterating and developing ways of testing early and often.

This is intrinsically a process of learning and discovery. We need structures to support multiple experiments on the way that we do our work, so that we can measure what works and what does not - and understand why. We also need a way to tolerate failures in our administrative-engineering innovation experiments; otherwise risk aversion will dampen support for the whole endeavour. These are all issues that universities handle extremely well in their research, but less so in their teaching and administration. We believe it is time that they start experimenting there as well.

REFERENCES

Agile Manifesto, The (2001) http://agilemanifesto.org/

Amacher, R.C. (2004*) Faulty Towers: Tenure and the Structure of Higher Education*. Oakland: Independent Institute.

Beck, K. (2005). *Extreme Programming Explained: Embrace Change*, (2nd ed). Addison-Wesley.

Brooks, F.P. Jr (1995) *The mythical man month*. Boston, Massachusetts: Addison-Wesley.

Cockburn, A., & Highsmith, J. (2001). Agile Software Development: The People Factor. *Computer*, *34*(11), 131–133.

Christensen, C. M. & Eyring, H. J. (2011). *The Innovative University: Changing the DNA of Higher Education from the Inside Out*. Jossey-Bass.

Daniel, J. (2012) Making Sense of MOOCs: Musings in a Maze of Myth, Paradox and Possibility. *Journal of Interactive Media in Education*, http://jime.open.ac.uk/2012/18

Dingsøyr, T., Sridhar, S., Balijepally, V. & Moe, N. B. (2012). A Decade of Agile Methodologies: Towards Explaining Agile Software Development. *Journal of Systems and Software*, *85*(6), 1213-1221.

Elementa Leadership (2012) The Agile University: higher education in a changing world, http://www.elementaleadership.co.uk/media/1257/elementa_agile_university_brochure.pdf

Gillies, D. (2011) Agile bodies: a new imperative in neoliberal governance. *Journal of Education Policy*, *26*(2), 207-223

Glass, R. L. (2001). Agile versus traditional: Make love, not war! *Cutter IT Journal*, *14*(12), 12-18.
Goldman, S.L., Nagel, R.N. & Preiss, K. (1995) *Agile Competitors and Virtual Organizations*. New York: Van Nostrand Reinhold.
Highsmith, J. & Cockburn, A. (2001). Agile Software Development: The Business of Innovation. *Computer*, *34*(9), 120–122.
Highsmith, J. (2002) What is Agile software development? *Crosstalk: the journal of defense software engineering*, *15*(10), 4-9.
Highsmith, J. (2010) *Agile project management : creating innovative products*. Addison-Wesley.
Jaschik, S. (2013) MOOC Mess, Inside Higher Ed, February 4, 2013. http://www.insidehighered.com/news/2013/02/04/coursera-forced-call-mooc-amid-complaints-about-course
Leckart, S. (2012) The Stanford Education Experiment Could Change Higher Learning Forever. *Wired*, http://www.wired.com/wiredscience/2012/03/ff_aiclass/
Stacey, R. (2006). The Science of Complexity: An Alternative Perspective for Strategic Change Processes. In R. MacIntosh *et al.* (eds.). Complexity and Organization: Readings and Conversations. London: Routledge. 74–100.
Suchman, L. (1987) *Plans and Situated Actions: The Problem of Human-Machine Communication*. New York: Cambridge University Press.
Williams, L. (2003) The XP Programmer: The Few Minutes Programmer. *IEEE Software*, *20*(3), 16-20.
Williams, L., Kessler, R.R., Cunningham, W. & Jeffries, R. (2000) Strengthening the Case for Pair Programming. *IEEE Software, 17*(4), 19-25.

AFFILIATIONS

Michael B. Twidale
Graduate School of Library and Information Science
University of Illinois
USA

David M. Nichols
Department of Computer Science
University of Waikato
New Zealand