Agility in information systems development

Kieran Conboy, Denis Dennehy, Lorraine Morgan, Roger Sweetman
Published online on: 21 Aug 2017

Accessed on: 04 Aug 2023
The chapter is divided into three parts.

- The first part provides a brief overview of the evolution of agile methods, how the application of agile methods from small to large co-located teams, to a variety of other contexts has emerged over time, and the predominant schools of thought on how agility is and should be evaluated.
- The second part provides a description of the next ‘wave’ of agile and how concepts such as flow are changing how we think about and apply agile in information systems development (ISD).
- The third part describes areas of ISD where agile methods have not been effectively applied, despite an ever-increasing need to do so (see Figure 10.1). These are project portfolio management and open innovation environments. It also considers how the next ‘wave’ of agile could help address existing issues.

Evolution of agility and agile evaluation in ISD

The concept of agility was formally introduced to ISD through the publication of a manifesto (Fowler and Highsmith, 2001) and embodied through approaches such as XP and Scrum (cf. Boehm, 2002; Highsmith and Cockburn, 2001; Highsmith, 2002). These approaches rejected the plan-driven bureaucratic thinking of the time in favour of faster, user-centric, and more dynamic methodologies that enabled continuous delivery, requirements change and reflection. These were employed with success across diverse contexts and domains (Boehm, 2002; Lindvall et al., 2002; Turk et al., 2002) and even extending to heavily regulated environments such as biomedical (e.g., Kane et al., 2006) and healthcare (e.g., Aronsson et al., 2011). There are two dominant perspectives in the agility literature, in terms of the enablement and evaluation of agility.

First, and more prevalent, is an adherence-based perspective, whereby agility is measured by implementation of commercially labelled ‘agile’ practices. In particular, examples can be found of measuring adherence or the application of metrics to XP (Layman, 2004; Williams et al., 2004; Mangalaraj et al., 2009) and Scrum (Scharff et al., 2012; Scharff, 2011). However, an extensive review and synthesis of discussion around the concept of agility identified
a number of significant failings of this adherence-based view (Conboy, 2009). For instance, the methodologies that form the basis of adherence suffer from shortcomings such as a lack of clarity, a lack of theoretical glue, a lack of parsimony, limited applicability, and naivety regarding the evolution of the concept of agility in other fields such as manufacturing. As a result, adherence-based assessments of agility make it difficult to compare methodologies, assess in-house methodologies, apply staged adoption of agility, and consider context. In addition, valuing adherence also contradicts the agile principle of using working software as the primary measure of success.

A second approach to agility assessment is a value-based enquiry. Failings of the adherence-based view are bridged with a focus upon goals and values (Agerfalk and Fitzgerald, 2005; Lindstrom and Jeffries, 2004). Rather than assessing the number of practices implemented from a predefined menu of commercial methodologies, one would assess the level of agility (the value) afforded by a practice or set of practices. Conboy (2009) developed a definition and formative taxonomy of agility for this exact purpose (depicted in Figure 10.2). This taxonomy was based on a structured literature review of agility across a number of disciplines, such as manufacturing and management where the concept originated, matured, and has been applied and tested thoroughly over time. The taxonomy addresses some of the shortcomings of an adherence-based view of agility in that it (1) considers the inherent agility-adding value of a set of practices rather than simply measuring whether or not commercially labelled agile practices are used; (2) allows agility to be assessed ‘in context,’ thus acknowledging the differences between design environments; and (3) allows a staged adoption of agility for those environments where more extreme levels of agility are not feasible or indeed necessary.

**Agility and flow**

Organisations are progressively seeking new approaches to improve their ISD process in order to increase agility and scale in software project management. Flow is part of the next generation of agile methods and is proving to be a catalyst for increasing agility and scale, especially in knowledge-intensive work activities such as software project management (Anderson, 2010; Petersen and Wohlin, 2011; Reinertsen, 2009; Power and Conboy, 2015). Although utilisation of the flow method is gaining momentum in the IS community, it is important to establish how flow brings agile methods to the next level.

Flow is about managing a continuous and smooth flow of value-creating activities throughout the entire software development process (Anderson, 2010; Reinertsen, 2009; Petersen and Wohlin, 2011; Poppendieck and Poppendieck, 2003). Flow emphasises the continuous movement of valuable work, rather than a sequence of discrete activities, performed by distinct

---

**Figure 10.1** The evolution of agile from small co-located teams to portfolios of projects spanning multiple organizations
teams or departments (Fitzgerald and Stol, 2014). Flow focuses on managing queues, rather than managing time lines and project phases or simple waste elimination, which makes it distinct from traditional project management (Power and Conboy, 2015; Anderson, 2013; Anderson et al., 2011).

Flow in product development is defined as “the progressive achievement of tasks along the value stream so that a product proceeds from design to launch, order to delivery, and raw materials into the hands of the customer with no stoppages, scrap, or backflows” (Womack and Jones, 2010, p. 306). In the context of XP, Beck (2000, p. 30) defines flow as “delivering a steady flow of valuable software by engaging in all activities of development simultaneously.”

One technique used to manage the continuous and smooth flow of work is the Kanban scheduling system, which uses a pull system rather than a push system. Kanban encourages the visualisation of workflow states as work passes through different states (e.g., Planned, In Progress, Done) in the ISD process (Anderson, 2010; Power and Conboy, 2015). Work-in-progress (WIP) limits are used to manage the quantity of WIP at any given stage in the workflow, as well as explicit policies frequently called entry and exit criteria that determine when a work item can be pulled from one state to another (Power, 2014). A system of coloured cards is used for signalling between upstream and downstream processes, as well as to enable team members to observe WIP.

Kanban has significant differences when compared to other agile methods in software development practices: (1) WIP limits are explicit rather than implicit; (2) it does not artificially time box sprints; (3) work items can vary in size, as there is no explicit rule that it must fit a specific time box; (4) release methodology is continuous delivery as opposed to the end of a sprint, and (5) change can occur at any time (Sjøberg et al., 2012; Cawley et al., 2013; Kniberg and Skarin, 2010). Research has also indicated that using Kanban leads to a higher level of productivity, as it is more responsive to change when compared to other agile methods (Sjøberg et al., 2012).

Cumulative Flow Diagrams (CFDs) are another tool that is used to visualise and manage workflow states in ISD. With origins in queuing theory, CFDs show the amount of work in each of the defined work states and are useful for understanding the behaviour of queues, and for diagnosing problems that are interrupting the fluent flow of work (Power, 2014; Reinertsen, 2009).

Complementary to the Kanban technique and CFDs is a number of key metrics used for understanding and managing the flow of work in software development practices (see Table 10.1).

Figure 10.2  Value-based measurement of agility
Source: Adapted from Conboy (2009).
In flow-based product development, metrics are used to understand the inputs, processes, outputs, and outcomes related to the flow of work and impediments (Power and Conboy, 2015). Impediments in software development are defined as anything that obstructs the smooth flow of work through the system and/or interferes with the system achieving its goals (Power and Conboy, 2014, p. 2). Impediments to the flow of work are broadly categorised: extra features, delays, handovers, failure demand, (too much) work-in-progress, context switching, and unmet human potential (Power and Conboy, 2014, p. 2).

While there is strong anecdotal evidence to suggest that awareness and indeed use of flow techniques is gaining popularity across the software development community (cf. Anderson, 2013; Nord et al., 2012; Petersen and Wohlin, 2011; Poppendieck and Cusumano, 2012; Power and Conboy, 2015; Reinertsen, 2009), the current body of knowledge on flow in ISD is nascent. Moreover, there are challenges that inhibit both its utility and theoretical development. These challenges are highlighted in the next section.

### Challenges of reapplying flow in agile

Despite its growth in popularity, the utility of flow is driven by practice-led research, which is not unusual for novel and emergent ISD methods (Conboy, 2009; Dybå and Dingsøyr, 2008). It does, however, present a corresponding set of challenges due to the complexities, scales, and contexts associated with contemporary ISD. First, the effectiveness of flow has largely been supported by anecdotal evidence (Sjøberg et al., 2012; Ebert et al., 2012). Second, the assumption that flow is suited to complex ISD environments has been subject to harsh criticism (Ebert et al., 2012). Third, as flow is a metric-driven process, it is critical that such metrics are interpreted in context because failure to understand the context will limit the understanding and the usefulness of the results (Kitchenham, 2010). Fourth, the absence of value-based research on flow-method use will result in a repeat of the mistakes witnessed with the adherence-based approach.

These challenges highlight a need for rigorous research on the assimilation of flow techniques in the real-world context for which its uses were intended and to understand the challenges of assimilating flow techniques in software development practices. The need for a deeper understanding of adaptability and extension of software development methods in

### Table 10.1 Metrics used to manage flow

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput rate</td>
<td>Reveals the rate of work through the system over time, and when combined with demand analysis shows how much work is value demand (e.g., customer requesting something new such as a new product feature) versus failure demand (e.g., when a product or product feature does not meet the customer’s needs and generates additional work).</td>
</tr>
<tr>
<td>Cycle time</td>
<td>Shows how long individual work items spend in specific workflow states and are used to understand how work flows through individual or combinations of work states.</td>
</tr>
<tr>
<td>Lead time</td>
<td>Shows how long it takes for individual work items to move through a system – from initial request to delivery to end user.</td>
</tr>
<tr>
<td>Queue size</td>
<td>Queues are the underlying cause of various forms of economic waste: longer cycle time, increased risk, increased overhead, and lower quality.</td>
</tr>
</tbody>
</table>

Source: Power and Conboy (2015); Reinertsen (2009).
practice and in a rigorous research approach were identified as significant shortcomings in the current body of knowledge (Conboy, 2009; Dybå and Dingsøyr, 2008). These challenges provide opportunities for the future development of agile and flow, which are presented in the following section.

**Future development of agile and flow**

The future development of agile and flow is contingent on the following four interrelated research activities. First, there is an opportunity to evaluate the adoption and use of flow techniques in practice. This would ensure that the effectiveness of the flow method is based on data that has been accumulated objectively rather than relying on anecdotal evidence. Second, given the complexity and socially embedded nature of the flow method, there is an opportunity to develop a flow maturity model. This would provide a roadmap for managers who are responsible for the implementation of flow techniques across distributed ISD teams, while also creating awareness that adoption of ISD methods is not a binary activity. Third, applying a suit of appropriate metrics to establish a baseline of the current state of flow and to monitor the improvements and transitions across the various adoption stages would demonstrate the value-add of flow in the context of ISD. Finally, having a deep understanding of agility and flow at a project level will provide the ideal platform for researchers and practitioners to effectively scale flow techniques to a portfolio level.

**Agility and project portfolio management**

Having considered the evolution of agile at the project level, it is now appropriate to consider its deployment in a multi-project environment and more specifically, its implications for a portfolio approach to information systems development. Traditionally, agile methods had been considered as constrained to small co-located projects carried out by individual teams (Hoda et al., 2010; Abrahamsson et al., 2009; Dybå and Dingsøyr, 2008). However, portfolio management has been identified as one of the most important topics for research in agile software development (Dingsøyr and Moe, 2013). This section introduces information systems project portfolio management (IS PPM), discusses the problems associated with it in agile portfolios, and describes how research can help address the shortcomings in both agile and IS PPM.

IS PPM is the ongoing identification, selection, prioritisation, and management of the complete set of an organisation’s information systems projects that share common resources in order to communicate portfolio success, maximise returns to the organisation, and achieve strategic business objectives (Meskendahl, 2010; Cooper et al., 1999; Blichfeldt and Eskerod, 2008; PMI, 2013). A review of the literature reveals that IS PPM consists of four components: (1) the identification, selection, and prioritisation of projects; (2) resource management; (3) strategic alignment; and (4) portfolio performance management.

A portfolio approach considers the total return on a complete set of components as opposed to the return on each of the individual components. This approach has been applied successfully in a number of disciplines such as research and development (Stummer and Heidenberger, 2003; Mikkola, 2001) and new product portfolio development (Cooper et al., 2001). However, despite being first applied over 30 years ago (McFarlan, 1981), the application of portfolio management in IS has had limited success. Studies have shown that up to 75% of IS portfolios fail to meet budget, time, and performance expectations (Whittaker, 1999; Keil et al., 2000; Bartis and Mitev, 2008; Gartner, 2014; Singh et al., 2009; Pervan, 1998). Moreover, IS portfolio failure is not restricted to certain industry sectors or project types; rather it occurs with some regularity in
organisations of all types and sizes such as health (Greenhalgh and Keen, 2012; Mark, 2007), finance (Drummond, 1999; Charette, 2005), telecommunications (Boonstra and van Offenbeek, 2010) and consulting (Conboy, 2010). Overall, there is little evidence to suggest the trend of portfolio failure is improving.

The continued high rates of IS portfolio failure highlight the problems with scaling agile. While agile methods have been shown to reduce the incidence of project failure (Conboy 2010), paradoxically, the difficulties with PPM are greater in organisations practicing agile (Stettina and Hörz, 2015). PPM has traditionally assumed that success is achieved by selecting the ‘right’ projects and executing each one effectively. Indeed, most PPM research focuses solely on the initial prioritisation and selection of the project mix (Cooper and Edgett, 1997; Meskendahl, 2010, de Reyck et al., 2005) and there is little research looking at the day-to-day management of portfolios (Frey and Buxmann, 2012). The idealistic perspective where all projects proceed in a linear fashion, with no need to consider change or interdependencies, is illustrated in Figure 10.3 (Stettina and Hörz, 2015).

While this approach may work in traditional industries (e.g., construction) or even in IS where a waterfall approach is appropriate and there is little change to projects, it is totally at odds with an agile approach that embraces change, even late in projects. The combination of constant change, interdependencies, and social and technical issues results in a complex dynamic portfolio that can prove difficult to manage (Orlikowski and Scott, 2008; Daniel et al., 2014) (see Figure 10.4).

While one may argue that the management of a portfolio is no different from the management of a single project, the scaling of management to the portfolio level is difficult for a number of reasons. First, individual project success does not ensure portfolio success (Billows,

![Figure 10.3](image_url)
2001; Conboy, 2010), and sometimes the entire portfolio is technically sound but offers little business value (Weill and Vitale, 1999). Consequently, the view that a good project management framework applied to individual projects in a portfolio inevitably leads to a successful portfolio does not always hold true. Second, in a portfolio there is a mix of short- and long-term projects with disparate and dyssynchronous time horizons. This makes the simultaneous management of different project time lines within the overall portfolio difficult (Collyer and Warren, 2009, Jeffery and Leliveld, 2004). Third, a portfolio requires management of projects and programs with multiple, conflicting goals and the management of multiple managers by a manager or management group. This can be problematic when project and portfolio managers have diverse preferences or nuances toward certain management types (Hansen and Kræmmergaard, 2013).

All these problems are exacerbated in an agile environment. However, while agile project management literature has increased in recent years (e.g., Fernandez and Fernandez, 2008), there is only a handful of empirical studies that has addressed the issues of scaling agile to the portfolio level (e.g., Stettina and Hörz, 2015; Rautiainen et al., 2015) and the relationship between them remains poorly understood. That said, a number of approaches to agile portfolio management have come to prominence in recent years. These approaches generally originate in the consulting literature and have been subject to little empirical validation (Stettina and Hörz, 2015). Two prominent approaches are now discussed.

Leffingwell (2007, 2010) developed the Scaled Agile Framework (SAFe) to implement agile practices at the enterprise level. The framework has three levels (portfolio, program, and team) and four values (alignment, code quality, transparency, and program execution). The portfolio management team prioritises the portfolio backlog and allocates resources. At the program level, product managers participate in program prioritisation, and at the team

![Figure 10.4 The reality of project portfolio management](image-url)
Agility in information systems development

level, five to ten agile teams deliver projects. However, the Scaled Agile Framework has been criticised by agile pioneers as being overly complicated and rigid (Schwaber, 2013) as well as being “relentlessly top down,” with no requirement for organisational agility (Jeffries, 2014). Furthermore, when compared with the definition of agile as illustrated in Figure 10.2, a number of issues arise. First, the focus on execution of programs as prioritised at the portfolio level can impair the enterprise’s capacity to either react to change or learn from it. Second, agile must not detract from perceived simplicity, whereas the SAFe adds to organisational complexity.

Krebs (2008) proposes “Agile Portfolio Management” to deliver a dynamically managed portfolio based on agile principles. Portfolio management is divided between projects, resources, and assets with a dashboard recommended to monitor the whole portfolio. Progress, quality, and team morale are proposed as project metrics. Resource transparency and the establishment of a Project Management Office (PMO) are considered key to agile portfolio management (Stettina and Hörz, 2015). Again there is little experimental validation of this framework. Like SAFe, it can be criticised as overly prescriptive. It assumes that what works in one organisation will work in all organisations, and there is little scope for “method tailoring” (Conboy and Fitzgerald, 2010) that has helped to make agile successful at the project level.

Future development of agile PPM

While agile promotes a bottom-up approach, its original development was aimed at single teams of around eight people. It is therefore unsurprising that agile has proven difficult to scale, and it should be noted that many of the criticisms aimed at agile approaches to portfolio management are merely extensions of criticisms that have been directed at agile software development (e.g., practitioner led, hard to scale). However, while it is not unusual for research in ISD to be practice led, it is necessary for academic research to address the shortcomings arising from this approach, develop effective theory to underpin practice, provide empirical data to support (or refute) claims of effectiveness, and illuminate limitations of such methods.

Indeed, academic research is seeking to both validate IS PPM and to provide a theoretical underpinning to existing frameworks as well as develop new frameworks (Stettina and Hörz, 2015; Vähänittty, 2012). For example, principles for scaling agile have been developed that seek to address the additional challenges associated between managing multiple teams across organisational levels (Dingsøyr and Moe, 2014). Furthermore, by articulating the challenges associated with agile at the portfolio level, academics may influence the next wave of agile methodologies in order to make them more scalable. Concepts such as flow (discussed earlier), which are currently being incorporated into agile, have the potential to help manage a dynamic portfolio of interdependent IS projects. For example, the focus on value in flow (Wang et al., 2012) can help with performance management at the portfolio level. Furthermore, flow’s use of visualisation techniques (Petersen et al., 2014) provides the necessary transparency to ensure resources can be moved rapidly around the portfolio as priorities change.

Agility and openness

While the previous sections focused on the evolution of agile at the project and portfolio level, this section reflects on how open collaborative practices can augment agile software development. A particular strength of agile approaches is that they move away from reclusive development, where the team building the system is disconnected from the customer. Instead,
Agile approaches continually involve the customer in the development process, often resulting in the development of a more innovative and valuable software product (Conboy and Morgan, 2014, 2011). Nonetheless, it is useful to contemplate how agile software development could benefit from more openness and transparency between agile teams and other stakeholders across an organisation. A previous study by Conboy and Morgan (2014) highlighted a number of challenges (see Table 10.2) that organisations need to consider when combining agile and open approaches.

Despite these challenges, however, there is great potential for agile and open development practices to be combined to enhance the effectiveness of software development inside organisations. While companies might think they have to make a clear choice between the two, this choice has more to do with the range of participants and the openness of the process to other stakeholders than it does with the philosophy of design and development (Goldman and Gabriel, 2005). Open collaborative development via communities, for example, open source software, is widely understood and accepted, and organisations are now realising that these characteristics can be applied to improve their own internal software development process as well, with many looking to apply them to enhance their own internal software development methods, typically in conjunction with agile or lean methodologies (Yeaton, 2012).

### Future development of agile and open

Given the notable success of open source communities and projects (Linux, Apache, etc.), many organisations are now applying open source development practices within their own internal software development environments, a phenomenon that is termed innersource (Stol et al., 2011). Comparable to open source development, innersource applies an open, concurrent model of collaboration and has been referred to as a good example of intra-organisational

---

**Table 10.2 Challenges in combining agile and open approaches**

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile methods open development to the customer only.</td>
<td>The customer plays an essential and integrated role in agile development. However, this practice does not extend to include other stakeholders in an open ecosystem.</td>
</tr>
<tr>
<td>Open development practices require significant time and resource investment from multiple stakeholders, as well as from the development team.</td>
<td>Often there are conflicts of interest that need to be addressed on a continuous basis. Thus, there has to be a high level of commitment from various participants. Furthermore, due to the commitment required, staff may miss or not prioritise their commitment due to time constraints or required work on multiple initiatives.</td>
</tr>
<tr>
<td>Open innovation can be difficult to implement when there is a level of competition or diverse interests between teams and business units.</td>
<td>Some developers are happy to discuss the ideas of others while being protective of their own.</td>
</tr>
<tr>
<td>Agile can damage visibility in open environments</td>
<td>Agile methods attempt to optimise the use of documentation and eliminate waste paperwork. This may be a significant concern if trying to create an open network of development teams.</td>
</tr>
</tbody>
</table>

*Source: Conboy and Morgan (2014).*
open innovation (Morgan et al., 2011). Innersource development implies benefits analogous to distributed ownership and control of code, early and frequent releasing, developer rotation, many continuous feedback channels, and improved innovation. Indeed, innersource can be a great tool to help break down silos, encourage internal collaboration, accelerate new engineer on-boarding, and improve overall productivity (Oram, 2015).

On the surface, agile methods and innersource might appear to be drastically different. However, the two share many of the same principles and values. For example, agile methods such as Scrum are highly prevalent in software development practices and often praised for being flexible and lightweight, thus enabling creativity and innovation given their emphasis on communication and collaboration and valuing people over process (Conboy and Morgan, 2014; Beck, 2000). Moreover, innersource is considered a development philosophy that focuses more on principles of egalitarianism, self-organisation, developer rotation, transparency, and meritocracy (Riehle et al., 2009; Agerfalk et al., 2015). Indeed, there is significant overlap between agile and innersource methods, as both approaches foster teamwork and collaboration, certainly essentials that assist in addressing some of the challenges outlined in Table 10.2. Nonetheless, to effectively take advantage of implementing innersource, there are a number of prerequisites that need to be put in place.

First, top management support is vital, particularly when developers may be contributing to another team’s project. Second, open development practices like innersource require a mindset of collaboration and cooperation. Thus, there need to be incentives that recognise contributions and provide the motivation for collaboration and willingness to share with other stakeholders.

Third, developing an innersource reference model that can feed into an organisation-wide common development process, namely establishing a rich taxonomy, or pattern language, consisting of patterns (each defining a context, recurring problem, and common solution) is important. The reference model can help with the dissemination of expertise, coaching, and training activities of agile teams.

Fourth, project managers need to plan appropriately for cross-team collaboration. Standards need to be clearly defined and followed, therefore, organisations should develop an effective governance framework and communications plan with supporting strategies and tactics for successful and sustainable agile development and innersourcing. This aspect should focus on the integration between organisational structures necessary to serve the business needs of the organisation (e.g., timely delivery of product release deadlines) as well as internal processes and relational mechanisms that serve to engage all stakeholders. Finally, developing metrics will help in quantifying and monitoring agile and open initiatives. Metrics may include the number of active projects and contributors, the distributions of contributions across projects, and so on.

**Conclusions**

Agile methods were designed with contemporary ISD environments in mind. However, the emergent and complex nature of ISD has increased, even since the original agile manifesto was conceived.

This chapter provided a brief overview of the evolution of agile methods since the early 2000s, and compared the ways agility is and should be evaluated today. The chapter also introduced the emerging concept of flow and discussed the ways it is changing how we think about and apply agile in ISD. The chapter then described project portfolio management and open innovation environments – areas of ISD where agile methods have not been effectively applied despite an ever-increasing need to do so.
When first drafting the sections on flow, project portfolio management, and open innovation, we considered each individually. However, it is interesting to note that across each of these areas we see similar issues — a lack of knowledge regarding how these concepts are applied in practice, a lack of research as to how they link and accommodate agility, and concern regarding the clarity and applicability of the concepts in practice. This chapter therefore raises some interesting issues regarding these concepts going forward.

References


Anderson, D. 2013. Lean Software Development. Lean Kanban University (LKU), Seattle.


Agility in information systems development


Agility in information systems development


