

---

# Delivering Software Sustainably

*John S. Nolan and Paul Dyson*

## Introduction

When working with clients we have found it useful to broadly classify their approach to software development and delivery in order to better understand the characteristics of the approach, as well as its benefits and risks. We felt the need to introduce this classification as we were reviewing and consulting to a number of projects that were described as 'agile' but that each had different benefits and risks to different stakeholders. Better understanding of the characteristics allows us to recommend or implement improvements to the development process more effectively<sup>1</sup>.

In this article we outline our classification and explain why it is important to understand a project's development style. We will also argue that even those projects that can truly be described as 'agile' need to augment their approach in order to deliver software in a sustainable manner.

## About sustainability

We have worked on a number of long-lived (more than 18 months) agile development projects. On each of these projects the primary aim of the development approach – actually delivering some valuable software in a timely and flexible manner – was achieved with great success. And yet there were many issues on these projects that the agile development literature had little to say about. Whenever these issues arose we had to draw on other areas of software development experience, particularly project and programme management, and figure out how to map the practices and experiences we found there into the development processes.

Looking back at many of those additional practices we realised that they were necessary in order to improve the ability to sustain the agile development approach over a long period of time. Sustainability is about delivering value in the short term without compromising the ability to continue delivering value in the long term: deliver today and keep on delivering tomorrow. This long term success is crucial to the survival and prosperity of the vast majority of businesses and, whilst agile development offers a great deal in terms of delivering today, it says little about continuing to deliver tomorrow.

## The classifications

In order to classify projects we use two categories of information: practices and characteristics. 'Practices' are standardised approaches to development and management activities that the whole team has adopted. Practices are the most easily assessed in terms of their presence, team proficiency and repeatability.

---

<sup>1</sup>We won't deal with 'Phased' or 'Waterfall' processes in this project as most of our experience of the past decade has been in delivering software with teams using some form of iterative or agile approach to development

---

Equally, practices are the most manifest thing to address with respect to process improvement.

'Characteristics' on the other hand are much harder to assess and more like outcomes or indicators of a team being within a given class of project. An alternative way of saying this is: practices are what the team consciously does, characteristics are the things that emerge as a consequence.

When we apply these classifications we are not literal; a project fits a classification if it broadly achieves the characteristics described by employing practices that are similar to those listed. We use these classifications to understand what can be realistically expected from a team and to determine how they can meet greater expectations, not to hand out a certificate with 'Agile' or 'Iterative' on it just because they employ practices from a prescribed list.

## **Ad-Hoc**

An Ad-Hoc project is one where there are few of the identifiable practices and characteristics that distinguish Iterative and Agile approaches. Ad-Hoc projects can be effective in limited circumstances but contain significant hidden risks associated with reliance upon individuals.

### **Practices**

By definition, an Ad-Hoc project has no practices in the sense that we mean them: standardised approaches to activities practised by the whole team.

Some of the activities associated with the development and management practices described for Iterative and Agile projects may be present at various levels of proficiency but they will not be used regularly in a disciplined manner by the whole team throughout the project time-frame. For example, intermittent bouts of planning and estimation when confidence ebbs in the "project plan" is not undertaking planning as a practice.

### **Characteristics**

The major characteristic of the Ad-Hoc project is the unpredictable nature of delivery. There is often a distinct lack of confidence in the plan (if there is an identifiable plan) and the promises it contains, both in time and functionality. This does not mean the project is not delivering; indeed they can be very successful within limited contexts, however, the ability to predict and plan is absent.

Successful Ad-Hoc projects tend to rely upon the particular individuals and a small team. This is often characterised by a fear in team managers about losing specific individuals, or the inability to grow the team beyond the "core group". Furthermore, there is often a pathological behaviour amongst the members of the team to over-promise and then making heroic efforts to deliver. Alternatively, and just as commonly, there is a collusion between the developers and their managers to overestimate everything in an attempt to introduce enough 'padding' to make deadline achievable.

Unsuccessful Ad-Hoc projects tend to have no tangible deliverables, stressed project team members and angry customers. A lot of the stress within the team will come from a tendency for team members to blame each other for the inability to

deliver: the managers will feel the developers never make promises they can keep and the developers will feel the managers keeping asking them to achieve the impossible.

## Iterative

An Iterative project is one that regularly delivers software, in a series of relatively short iterations, to a plan and set of requirements that is relatively static for the duration of the project.

## Practices

These are the practices we identify with Iterative projects:

Short Iterations	<p>Iterations typically last 1-2 weeks at the development level. They have clear goals and hard deadlines. Key to this is that deadlines are not missed and every effort goes into ensuring delivery as promised.</p> <p>The iterations have a repeating and regular internal structure, roughly; initiation, planning, developing/testing and delivering. At the end of every iteration is a software release that can be QA'd, reviewed and, potentially, taken live.</p>
All Development Activities Occur During Iteration	<p>The activities that the development team undertake during an iteration time-box include all the activities required to develop and deploy a software system (at least to a test environment). This means designing, coding, testing, deploying. This distinguishes Iterative development from Phased development approaches which may have time-boxed activities in sequence (i.e. phase 1=Analysis, phase 2=Design, etc).</p>
Regular Management Stand-Ups	<p>At regular times during the iteration team leaders and managers meeting for a short session to communicate with each other about the ongoing progress of an Iteration. The form is usually a summary of what has happened and what is about to happen plus exposure of any new risks. This is also usually the time for agreeing who is going to resolve issues and address new risks amongst the people present (but they are not resolved in this meeting).</p> <p>Management stand-ups usually happen daily or at least twice a week.</p>
Continuous Integration and Build	<p>In order to ensure that there will be a working release at the end of every iteration the development team will have to integrate their code and build the complete system on a 'continuous' basis. On many projects this will mean integration and build many times a day. On almost all successful projects with a short iteration length there will be at least a daily integration and build point.</p>

## Characteristics

The key characteristic of an Iterative project is reliable, regular delivery every few weeks. Development efforts are focused at hitting the iteration deadline and there is noticeable collaboration around how to achieve this.

Iterative projects tend to have few interruptions for meetings to sort out conflicts and issues. The daily management stand-up allows a specific time for the major portion of this activity, leaving the resolution collaboration itself, which is often then done as part of the normal flow of work by the development team.

Iterative development is effective in exposing project and technical risks. Structuring of the plan into iterations means that there are clear and measurable

goals for each release of the software delivered. Carrying out all the development and QA activities on an iterative basis means that assumptions can be challenged and technical problems exposed much earlier in the project than if a phased or waterfall approach was taken. An Iterative approach can be very successful in guiding a large team to continuously deliver in an environment where there is detailed specification of requirements and where changes to the requirements or the solution tend to be small and infrequent.

The major shortcoming of Iterative projects is their inability to deal with significant changes. These changes may come about due to changing circumstances around the project but may equally come from early visibility of the system; the exposure of technical problems or the reaction of the users to the system under development leading to new or modified requirements. But the practices of an Iterative project are focussed on reliable delivery, not on the ability to respond reliably to change.

## Agile

Agile development projects regularly deliver software, in a series of relatively short iterations, to a plan and set of requirements that are elaborated and changed throughout the lifetime of the project.

### Practices

The practices associated with an Agile projects encompass those of an Iterative project and also include the following additions:

Daily Scheduling Meeting	Each day the development staff meet for a short time (<30mins) to agree on who is doing what during the current day. Developers sign-up to tasks; they are not assigned to them.
Test-Driven Development	Tests are written in concert with changes to the code base by the development team. For every change there will be matching tests to confirm the behaviour. Consequently, there are a large number of tests that can be run to confirm the functional integrity of the whole system.
Collective Ownership of Artefacts	All developers 'own' all of the code, configuration or documentation of the system. They can make changes to any of the collective artefacts. Consequently, when making changes it is their responsibility to ensure all the artefacts are kept in sync with each other – no one person is solely responsible for a particular piece of code or document.  Pair Programming is a particular development practice that enshrines the concept of collective ownership of artefacts: every artefact is developed by two team members in a pair and pairs are constantly formed and reformed to ensure everyone in the team pairs with everyone else at some point.
Product Backlog with Dynamic Priorities	There exists a dynamic, prioritized list of all of the required functionality of the system. This is normally maintained by a product manager in collaboration with the customer and user base. It is the artefact which is used to feed the planning sessions.  Key to the Agility of a project is that no anticipation is made about what is being done in the current iteration based on the projection of the next iteration from the Product Backlog as it may well change.

Iteration Backlog with Fixed Priorities	The Iteration itself will have a fixed Iteration Backlog of functionality with associated development tasks. This Backlog has fixed priorities which do not change during the Iteration.
Collaborative Iteration Planning	Iteration Planning is undertaken as a collaborative process at the beginning of an Iteration, involving the development staff and the product management team. The Iteration Backlog is constructed from a combination of Product Backlog requirements, developer estimation and projected resource availability.
Management by Exception	With such a regular and disciplined process, the practice of managing the development of the system becomes one of “managing by exception”. Intervention is only required when something goes outside a tolerance, such as estimated time or emergency re-prioritization.

### Characteristics

The major characteristics of an Agile project are the self-organizing team, dynamic leadership, and the ability of the team to respond to changes in requirements or solution specification in a timely and predictable manner.

Within the agreed constraints of the Iteration Backlog the team organize their own work schedule and act in a stigmergic<sup>2</sup> manner upon the artefacts of the project; individuals (or pairs) modify shared artefacts for the purposes of their own tasks working with the state of the system as found without needing to co-ordinate and plan these changes.

Agile teams demonstrate a high degree of dynamic leadership<sup>3</sup>, allowing individuals to provide direction, motivation and generate interest around particular issues outside of their perceived seniority or responsibility. More than this, the teams will co-operate and follow such leadership if it is seen to contribute to the project. However, dynamic leadership can produce a negative characteristic: Agile development teams can sometimes be seen as arrogant or aggressive in their approaches to other teams and the rest of the organisation. This undermines the benefits an Agile approach may offer to the greater organisation and its business and requires effective management.

Agile teams plan with confidence and enthusiasm, with little or no reference to “changes in requirements”. This confidence is not misplaced; the foundations of practices such as Collective Ownership of Artefacts, Test-Driven Development and planning based on feedback metrics such as Load Factor or Velocity, mean that Agile teams tend to hit their estimates repeatedly without the need for padding or heroics. Change is accepted as a fact of development projects and the practices are seen as a disciplined mechanism for producing a plan and delivering in an ever changing environment. In fact all aspects of the project are open to change: it is not uncommon for priorities and plans to co-evolve in the Planning session, as the

<sup>2</sup> Stigmergic is a term from the study of self-organizing systems, first coined by Pierre-Paul Grasse in 1959. It refers to a method of indirect communication where individuals communicate through modifying the artefacts of their local shared environment. See <http://en.wikipedia.org/wiki/Stigmergy>

<sup>3</sup> We refer to management and leadership in the John P. Kotter sense (“A Force for Change: How Leadership Differs From Management”, John P. Kotter, 1990, The Free Press, ISBN 0-02-918465-7) where management is about producing predictability and order, and leadership is about setting direction and inspiring change. Kotter claims a project needs to have a good balance of management and leadership in order to produce long term (sustainable) success.

---

understanding of the overall impacts of development estimates and tactics for achieving the prioritised goals are discussed.

## **Agile and Iterative: What's the big difference?**

The major noticeable difference between Iterative and Agile projects is how they deal with change. On a project we consulted to that was 'just' Iterative, the plan for delivering the system was split into a number of iterations. Within each iteration there was limited scope for change; trivial changes that could be absorbed into the work planned for that iteration were okay. Despite the use of iterations, however, the total plan for the project was seen as a single whole and the scheduling of work to an iteration was based on the needs of the development team rather than the customer. This meant that large changes, either due to changes in circumstance or based on visibility of each iteration's release of the system, could not be incorporated as the knock on effect was to displace work later in the plan, which could be more vital to the system than the work carried out to date.

Whereas Iterative projects tend to resist change, Agile projects “embrace change” through regular replanning and re-prioritisation. The additional Agile practices help create a mechanism that continually delivers the highest priority functionality in an ongoing manner. We identify that by adopting this iterative and collaborative approach to managing prioritization Agile projects are more sustainable than Iterative projects. Indeed, we believe by extending this concept and making all activities associated with projects iterative leads to greater project sustainability.

## **Applying a classification**

From the practices and characteristics listed above it should be fairly plain that the each classification represents a higher degree of discipline and ability to deliver over the previous one. That is not to say that each classification represents an approach that is better than the previous; if you're using an Ad-Hoc process and it is working for you then why change to an Iterative approach just because it is more 'disciplined'? However, most of our clients engage with us because some aspect of their development and delivery approach isn't working and they either want to improve it to deal with those limitations or want to try a totally different approach.

In each of these cases understanding roughly where the development and delivery team is in the spectrum from Ad-Hoc to Agile has been useful in helping us understand what we can realistically expect from that team. Often the expectations the project stakeholders have of the team are far greater than what they realistically should expect. Sometimes what can realistically be expected falls far short of what is required in order to successfully deliver today and keep on delivering tomorrow. Understanding where you are in relation to where you need to be is the first step to making concrete improvements to the development and delivery approach.

## **Improving sustainability**

The progression from Ad-Hoc to Iterative to Agile doesn't just represent increasing discipline; we believe it represents an increase in sustainability. An Ad-Hoc process

works at a given time, for a given set of people, in a given circumstance. If any of these change, particularly the project team itself, the Ad-Hoc process may well break down. An Iterative process is more sustainable: its a repeatable process and the regular releases demonstrate progress and make some risks visible and, potentially, addressable. However a process that is 'just' Iterative can't comfortably deal with the significant changes in requirements or technical approach which are bound to happen in a project of any significant length. An Agile process is more sustainable still and a good process responds to changes in requirements, technology and the team.

Some of the Agile practices are clearly aimed at sustainability, such as the 40-Hour Week or Sustainable Pace practices of XP. However, a pure Agile process is not truly sustainable in the long run. Agile is focussed on delivering valuable software today and the practices have been chosen to cope with even quite rapid changes in what 'valuable software' means. But there are many measures of business value which cannot be delivered by changing the way the system works. One of the most common complaints we hear from senior managers is that Agile projects can tell them, with some confidence, what will be delivered in the next week, or two weeks, or month, but very little about what can be delivered in nine months or a year. "How does this help", they say, "when I have to plan an annual budget and have a major deadline to hit six months away". When that deadline represents the launch of a new product, the introduction of some legislation that must be complied with, or the delivery of a new system that other development projects depend on, having confidence about what functionality will be delivered in the next iteration is not sufficient.

Another interesting example of Agile development becoming unsustainable was on a highly successful, long running XP project where the lead developer expressed de-motivation as , "the frustration and monotony of being able to deliver exactly what we commit to; constantly and repeatedly". The strong emphasis on regular commitment to deliver functionality prioritized by the customer, with a very short-term horizon, leaves little scope for the more creative aspects of technical development. It is difficult to argue the prioritization of some of the non-functional aspects of the technical solution (such as changing the database vendor) when it has no short-term benefit, leading to an anxiety about the ability to keep delivering in the long term.

## Sustainability practices

Practices which improve sustainability stretch throughout the structure of a project and its context. A selection of some of the practices we use and introduce to clients are outlined in the following table by way of illustration:

Gold Cards	A mechanism for developers to use at the Daily Scheduling Meeting where they can opt-out of the day's work to undertake self-guided work. Usually each developer has 2-3 gold cards a month, which is taken into account in planning. Gold Cards give time for exploration and technical leadership activities to happen, they improve motivation and provide relief from the monotony of constant delivery.
------------	--

Retrospectives	Scheduled and structured sessions for the team to reflect on the project and its process to date and to identify areas for improvement. We advocate a combination of both Heartbeat (every 3-4 weeks) and Project (every 5-6 months) forms. Retrospectives are a forum for a mixture of management and leadership activities.
Project Board	Having a properly constituted, independent project board to which the team must report progress regularly every month. The activity of generating the report is a collaborative exercise and some of the project board members should be responsible for communicating project progress and success out to the wider organisation.
Regular Project 'Initiation'	For the project and its sub-phases undertake an iterative "initiation process" to maintain the statement of purpose, business justification and risk profile. These highlight and strengthen the awareness of management issues
Extended Team Sessions	Regular sessions with a mix of people from across the delivery and business teams for informal interaction outside of the organisational structure. These sessions are not what are commonly referred to as "team building sessions"; but more like project specific think tanks with a cross-section of the extended team that normally do not work together. Not only do they produce action plans, they also help with motivation and building the informal networks which are present in successful projects
Release Planning	Development plans are mapped out for the coming months, not just weeks. There is a necessary reduction in the detail of the requirements and the plan from the short-term (high detail) to the long term (much lower detail) so the ability to quickly respond to change is not lost. The work expected to take place in the medium-term should be planned in sufficient detail that managers and customers can identify potential risks to upcoming milestones and deadlines. The longer-term work needs to be planned in sufficient detail that budgets and resources can be planned. Of course, all these plans should be reviewed iteratively and re-prioritised.

## Characteristics of a sustainable approach

The purpose of introducing these and other sustainability practices is to sustain the short term benefits of an agile approach to software delivery through a long period of time; sustainability is itself a characteristic. However, 'sustainability' isn't easily quantified or recognised so there are three key characteristics that we believe are the major outcomes of increasing sustainability.

The first of these is that the project is highly integrated with the Programme and Business Management processes of organisation. A sustainable approach is one where the Development, Programme and Business Management processes execute in a structured and co-supporting manner; there is a noticeable dependency and influence on each other between all three constituencies. Like a well-oiled machine, the parts mesh together and run smoothly, but both driving and giving feedback to each other.

The second major characteristic is the ability to plan with confidence and measurable competence for the longer term. Whether planning software releases, resourcing levels or budgets, there is sufficient detail to ensure the plans are realistic and sufficiently frequent re-work to ensure they change as the demands on

---

the project changes. These plans are very visible and open to question, whether from the project board or the development team.

The third characteristic is a working practice of 'continuous improvement'. A sustainable approach to software delivery recognises that *everything* can change throughout the project: the software, the team delivering it, the users and stakeholders, the process and practices, and even the goals and objectives for the project. Just because a practice was the right thing to do at the start of a project doesn't mean it is still the right thing to do two years later. A sustainable approach to software delivery is one where the team continuously reflects on what it is doing, in a structured manner, and make improvements wherever it is valuable to do so.

## Conclusion

In this article we have described a model of software development that characterises different approaches as Ad-Hoc, Iterative and Agile development. The purpose of this model is to understand what can be reasonably expected from a development team operating in a given style, and if change is required, what practices need to be addressed to improve the situation. We have also explained why we think being agile is simply not enough unless you are delivering a short project with a small team into a small organisation.

To be clear: we are advocates of agile development. Sustainability is about delivering in the short term without compromising the ability to continue delivering in the long term; one of the reasons we advocate Agile is that far too many teams fail to deliver in the short term. Indeed, we advocate Ad-Hoc teams progress via Iterative development, as getting the management of delivery under control addresses many of the problems attributed to software development by customers and managers, and this will gain you permission to continue the improvement process to become Agile.

But if you're delivering a system into a corporate environment, or as part of a long-term project, delivering in the short term is just the first problem to solve. Once you start delivering effectively, there is a different set of changes you will need to make to your practices and processes if you wish to sustain that success over the long term.

## About e2x

e2x offers consultancy in project, programme and change management, as well as software development and delivery services. e2x helps its clients to improve the sustainability of their development process whether they are currently employing an Ad-Hoc approach or have a functioning Agile approach that needs to be more sustainable (or even if they wish to move away from a Phased or Waterfall process). A key aspect of our consultancy is ensuring that any new practices and processes adopted are well integrated with existing project, programme and business management mechanisms: to be truly sustainable an Agile approach has to adapt to the business, not the other way around. e2x also supports its clients through the change process; we provide hands-on assistance for every role from Developer to Project Director, from Tester to Project Sponsor.

---

e2x has the most experience of sustainable, agile project/programme management and development of any consultancy in the UK. e2x consultants are recognised as being agile pioneers and world-class experts in the field. e2x has particular experience of running large-scale, long-term agile projects in corporate environments (outside the normal agile 'sweet spot' of small teams, short projects and small organisations); all of our people are experienced practitioners as well as consultants.

John Nolan is a partner in e2x and has been an agile practitioner since 1999. Founder and CTO of Connextra, one of the world's longest-running agile (XP) projects; Connextra has been used as an influential case-study within the agile community. He has many years of experience working in a number of domains from finance and advertising through to engineering and research. John is also a certified ScrumMaster, speaker at international conferences and an ACM Distinguished Engineer

Paul Dyson is a partner in e2x and a practising advocate and early pioneer of agile development methodologies. Paul led the first XP project run in the UK in 1997 and was instrumental in the early promotion of agile through seminars and conference workshops. Paul has provided technical leadership on a wide range of agile projects, especially those that don't fit within the agile 'comfort zone' of a small team working closely with a single customer. Paul is a published author and a speaker at a number of international conferences.

## Contact us

e2x limited  
35 New Broad St.  
London EC2M 1NH

t: +44 (0)207 194 8015  
f: +44 (0)207 194 8016

[www.e2x.co.uk](http://www.e2x.co.uk)  
[john@e2x.co.uk](mailto:john@e2x.co.uk)  
[paul@e2x.co.uk](mailto:paul@e2x.co.uk)

## Acknowledgements

A number of people have read and commented on this article. Particular thanks goes to Kent Beck, John Daniels, Tim Diggins, and Bernard Horan.