Adopting 'Agile' and 'Scrum' Practices as 'Organisational Becoming': Cases from the UK Video Games Industry

Juan Mateos-Garcia and Jonathan Sapsed

CENTRIM - Centre for Research
in Innovation Management,
University of Brighton,
The Freeman Centre,
(University of Sussex campus)
Falmer, Brighton
BN1 9QE, United Kingdom

Tel. +44 (0)1273 877942
Fax. +44 (0)1273 877977

Email: j.d.sapsed@bton.ac.uk

Abstract

Established 'rational' methodologies in the field of project management are being increasingly challenged by scholars who argue that the emphasis that they place on idealised top-down processes neglect 'soft' human dimensions of projects. This has led to negative outcomes such as delays in delivery, low quality products and overshot budgets. In this paper, set in the empirical context of video game development, we present and analyse the Agile Programming Paradigm, an approach to the organisation of software projects that has recently emerged as an alternative to traditional, formal project management and organisation methodologies. The proponents of Agile advocate a bottom-up approach to management with an emphasis on constant product iterations and interaction with customers. They argue that a shift in attention from processes, documentation and measurement to 'softer' variables supports a development system better able to 'embrace change', which is understood as the key limitation of formal, rational methods. We use emerging analyses of 'organisational becoming' first advanced by Tsoukas and Chia (2002) in order to frame our discussion theoretically, suggesting that the Agile Paradigm constitutes a potential answer to a key question formulated during their research, 'what must organisation(s) be like if change is constitutive of reality'.

In the empirical part of the paper we present three case studies of organisations that have implemented Agile techniques. This analysis informs a subsequent discussion where we assess the advantages and limitations of the Agile Paradigm. We conclude with an interpretation of our findings within the 'Organisational Becoming' philosophical framework.

Introduction

Established 'rational' methodologies in the field of project management are being increasingly challenged by scholars who argue that the emphasis that they place on formal, idealised processes implemented from the top-down has led to a neglect of 'soft' human dimensions of project management, with negative outcomes epitomised by delays in delivery, low quality outcomes and overshot budgets (Hobday and Brady 2000). In this paper, set in the empirical context of video game development, we present and analyse the Agile Programming Paradigm, an approach to the organisation of software projects that has recently emerged as an alternative to formal methodologies such as 'the Waterfall model'. The proponents of Agile put forward practices with a strong focus on the nature of work carried out by software teams, a bottom-up approach to management and an emphasis on the dyad of interaction and iteration. They argue that a shift in attention from processes, documentation and measurement to these other 'softer' variables supports a development system better able to 'embrace change', which is understood as the key limitation of formal, rational methods. We use emerging analyses of 'organisational becoming' first advanced by (Tsoukas and Chia 2002) to frame this debate.

After a brief discussion of some aspects of video game development that hinder the success of rational management approaches, we describe the key elements of Agile Programming and the ways in which they purport to address them. In the empirical part of the paper we present three case studies of organisations that have implemented Agile techniques in different ways and with diverging degrees of success. This analysis informs a discussion where we examine the advantages and limitations of Agile techniques when compared to rational approaches to software management, and identify those contexts where their application would seem most suitable.

2- Theoretical Discussion

a) Embracing change as a way of organisational becoming

(Tsoukas and Chia 2002) argue, in their discussion on traditional approaches to organisational change in management science, that the dominant epistemological framework in the discipline has placed an excessive emphasis on the role that institutional structures (such as for example routines) play in promoting stability in organisations. In this context, change is seen as an exceptional occurrence that managers initiate from the top-down, rather than as the pervasive state of affairs where organisations operate, act upon and are influenced by. They contend that the prevalence of the former view has resulted in a fragmented understanding of organisational change and to problems in its implementation, and defend the adoption of a new philosophical approach where change is understood as the norm in an organisation, rather than as the exception.

They assert that in order to accomplish this, it is necessary to focus on the ways in which the habits and beliefs of networks of actors at all levels of an organisation (and outside it) are interwoven creating pervasive patterns of transformation with beginnings and ends that can only be arbitrarily established. These processes are what they define 'organisational becoming'.

Tsoukas and Chia are advocating a conceptual and methodological shift in the area of Management Science that, when incorporated into organisational practices, can give rise to new structures for project management, and original outcomes. In this paper we explore these issues by analysing the context of application, advantages and limitations of the Agile Paradigm, an emerging organisational approach with a philosophy and practices that seems to constitute a possible answer to the central question informing Tsoukas and Chia's work, this is, '*what must organisation(s) be like if change is constitutive of reality?*' More specifically, we look into the way in which the implementation of management structures which legitimise the bottom-up emergence of organisational and communication arrangements for development, and a multiplicity of interactions with a diversity of stakeholders *via product iterations* promote flexibility and creativity in video game organisations. In a dynamic application of Conway's Law (Conway 1968)), we should expect this organisation to develop products with architectures better able to incorporate internal (creative) and external (market) influences, key determinants of market relevance and success.

We accept, however, that 'embracing change' might give rise to undesirable trade-offs by reducing, for example, the predictability of development outcomes, or the stability of a product component at a particular moment in time, and giving rise to communication and co-ordination costs. Our analysis is pragmatic and suggests that a philosophy of management that focuses on change, as perhaps reflected on a wholehearted embrace of Agile tenets, might not be suitable for all organisations, or at least to all areas inside a project. We examine these issues in the empirical context of three video game studios who have followed different Agile implementation strategies, with a diversity of outcomes.

b) Organisational crises in video game development and the limitations of a rational response

Rationalisation of growing team sizes

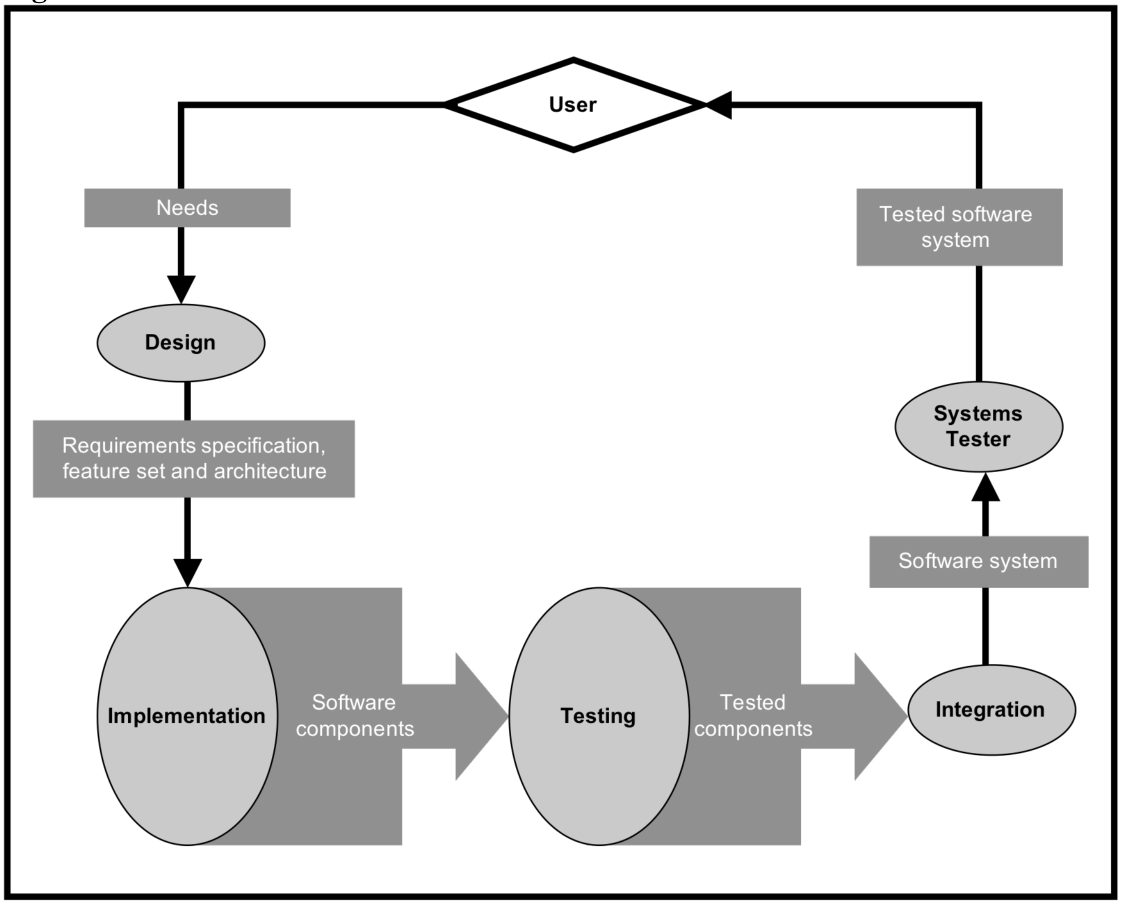
The video game sector is a software-intensive creative area of growing economic importance where organisations have for a long time struggled to establish suitable methodologies for product development. The need to integrate the outputs of a diversity of disciplines into sophisticated products, the rapid rates of technological change, intense competitive pressures and the fuzzy definition of quality (inherent to most creative sectors (Caves 2002)) have resulted in uncertainty regarding which are the right approaches to the design and creation of video games. This situation has opened up a space for organisational experimentation in which many studios are currently engaging.

The majority of the efforts towards better management of video game development have been informed by research in the Software engineering field. This discipline, devoted to the elaboration of methodologies and processes to improve the efficiency of software development, emerged as a reaction to the 'software crisis' of the 1960s. This situation was brought forward by the need to develop increasingly complicated software systems to harness growing hardware power, a process which put intense pressure on the management capabilities of organisations faced with the need to co-ordinate the activities of mounting numbers of developers (Dijkstra 1972; Somerville 2006). The move towards 'next generation consoles' in video game development, and the ensuing need to enlarge team sizes in order to build products able to fulfil the enhanced potential of new hardware has created similar pressures towards rationalisation in video games (McGuire 2006).

The Waterfall Model is a dominant approach to the organisation of software projects often translated to video game development. This methodology, which was first presented, with a different terminology, in (Royce 1970) establishes a set of clear-cut process phases that any software project should follow (Cusumano 2006; Somerville 2006). Royce proposes a strong design effort at the onset of a project aimed at elaborating a robust and thoroughly documented software architecture that can be implemented afterwards in a systematic and predictable fashion.

The logical structure of the waterfall model is reflected in the organisational division of labour inside a video game studios: game designer, developers (this is, programmers responsible for software code and artists engaged in content production), testers etc., are located inside different functions of the organisation, carrying out their specialised activities at each of the different phases presented in Figure 1.

Figure 1- The Waterfall model



Another prevalent tool for process management in software engineering is the Capability Maturity Model (CMM). The CMM can be understood as a framework for process improvement inside a software organisation which strives to establish a productive environment where tasks are 'optimised' (Paulk, Curtis et al. 1993). The goals of CMM are increased precision and predictability in an organisation's performance as tasks become growingly well-defined, stabilised and managed. CMM requires the implementation of a particular lifecycle model for software development, and this has traditionally been the Waterfall model.

Criticisms of Waterfall

The Waterfall model remains highly popular in software development, and many video game studios trying to improve the efficiency of increasingly complicated development processes have adopted it. Although in principle, Waterfall appears like a logical way of organising software development, with step-by-step sequence including the determination of user needs (or project goals), strategies to address them, a plan for their implementation and built-in Quality Assurance procedures, it has nevertheless been subject to strong criticism from both scholars and practitioners engaged in software (and video game) development.

They contend that the model's assumptions about the stability and predictability of the environment where development activities are carried out are unrealistic to an extent that renders this method inapplicable, or even harmful, for projects above a certain threshold of size, or inside dynamic sectors. These critics argue that uncertainty and change render highly detailed project plans irrelevant. Henceforth, the ones adopted in practice are either vaguely defined, 'simulacra' of reality that can be understood as useful maps guiding the development effort (Parnas and Clements 1986), or as strategic justification for budgets and milestones in the context of a contract with a client (Brooks 1995). The survival of the Waterfall model in software production environments is often presented as a consequence of the way in which it creates an illusion of predictability and offers software clients a legal basis to hold their contractors accountable for the delivery of project milestones within an agreed schedule and budget.

We examine below the reasons why this predictability is in many cases deceiving, with initial references to the classic sources in the software development literature, and examples from video game development.

<u>*Plans are disrupted by emergent problems and unpredicted dependencies*</u>

The Waterfall model assumes that the uncertainties of software development can be addressed through enough ex-ante planning, and that the intricacies of implementation can be codified and communicated via a complete set of documents (Parnas and Clements 1986).

Critics of the model argue that perfect planning at the front-end of development is impossible because unexpected problems and dependencies are bound to emerge during the implementation of any large project. If there is a strong degree of inter-linkage between the tasks of different team member (as it tends to be the case), it becomes very difficult to increase or reallocate a project's staff in order to address emergent problems without further disruption. Bringing new developers into the project increases communication overhead, interdependencies between tasks and potential sources of error, resulting in the outcome which Brook's summarises in his well-known Law '*adding manpower to a late software project makes it later*' (ibid p. 25).

The knock-on effects of the changes made as a reaction to emergent problems puts severe stress on projects, particularly towards their latter stages, when testers often find more, and more difficult to solve bugs that expected or allotted for in the schedule. The video game organisation faces, in this context, an undesirable disjunctive: to release a late product, or a buggy one.

An strategy often adopted by organisations with limited resources, or aware of the problems brought by a spike in project headcount late in development, is for the team to go into 'crunch mode', this is, to do large amounts of (usually unpaid) overtime in order to release a more or less ‘complete’ product on time and budget. Crunch-mode is however reported to impact negatively the productivity and morale of developers, and has become a very controversial practice the sector, where management is growingly aware of quality of life issues (Hyman 2007).

The elusive dimension of ‘fun’ (which could be understood as the quality and uniqueness of the gaming experience) constitutes a key emergent variable difficult to predict at the planning and component design stages of a project. A game integrates content, technology and user, and gauging the nature of the experiences of the latter through an analysis of its components early on development is deemed impossible by practitioners: the proof of a game is in the playing it, so until the game being produced can be played (even as a prototype) there is a large degree of uncertainty regarding the success of the development effort. It is often the case that a game’s fun factor is found wanting when its components are integrated towards the end of development, and the perennial disjunctive between releasing a late product, or a low quality one is once again faced by developers (McGuire 2006).

Plans are disrupted by changes in the environment

The uncertainty that characterises software development also presents important external aspects. For example, the needs and priorities of clients might be difficult to elicit precisely at the beginning of a project and do in occasions change halfway through its lifecycle, when new features are requested, disrupting the development effort. The latter problem is particularly severe in the case of products being developed for highly competitive markets such as video games. In this sector, the need to react to innovations from rivals or to pressures from publishers trying to steer the direction of a product as market conditions change often makes it necessary to modify a project late in its development.

There is conflict between the goals and incentives of different stakeholders

Another important limitation of the Waterfall Model is that it assumes concordance of goals inside the development organisation, and therefore does not address issues caused by the divergence of priorities and the misalignment of incentives of the disciplines or groups engaged in the development of a software project (Boehm and Ross 1989). The traditional divide between resource-aware ‘rationalising’ conservative project managers and creative, difficult to manage developers, exemplifies this problem, also manifest inside the developer ‘class’. In the latter case, tensions and communication breakdowns between programmers and artists with different priorities and professional languages are often reported as a source of problems during development. Given the high degree of inter-linkage between the activities of these actors, projects planned according to a methodology which neglects these issues face unexpected disruptions from internal conflict and misunderstandings.

The trade off between creativity and routinisation

The adoption of incomplete plans which acknowledge the uncertainty of development processes, or the rejection of complete ones when changes in the production environment renders them invalid makes it necessary for developers to exercise their creativity throughout the production process, not just at its front-end.

The existence of this space for the exercise of creativity is often presented as one of the reasons why video game developers decide to enter this industry, in spite of smaller salaries compared to other software-intensive sectors: video game development is perceived to be 'fun'. In this context, strides in the direction of creating a highly predictable and routinised development environment along the lines defined by both the Waterfall and the CMM models might alienate the creative personalities that constitute key sources of competitive advantage for a video game studio (See () and () for a discussion on intrinsic motivations for creativity).

On the other hand, creative freedom can, if unchecked, lead to the widely reported problem of 'feature creep', which emerges when more features than those initially specified are included in a product (Grantham and Kaplinski 2005). Ambitious or over-enthusiastic developers might lose focus of the client needs and go on 'wild goose chases', perhaps underestimating the resources required to implement a feature correctly, or without testing its impact on the performance and usability of the rest of the product. Although feature creep can take place in almost any product of design (Norman 1988), the apparent easiness with which additional functionalities can be included in a video game, uncertainty regarding quality parameters and the desire to target enlarged market niches makes this problem particularly severe in this sector.

c) What is the Agile Paradigm? How does it address these issues?

The Agile Manifesto (Beck, Beedle et al. 2001) signed by the creators of several innovative and, in their own words, 'organisationally anarchistic' software development methodologies (including Scrum, Extreme Programming, Adaptive Software Development. Crystal, Feature-Driven development and Pragmatic Programming) proposes an approach to software development very different from the rationalist, plan, document and process intensive strategies implicit in the Waterfall Model.

In the words of Kent Beck, Agile means 'accepting input from reality and responding to it' (Computerworld 2007). The Agile manifesto proposes a focus on 'individuals and interactions', 'working software', 'customer collaboration' and 'responding to change' in order to satisfy customers. There is an acceptance of the uncertain and shifting context in which software is developed, conditions intensified with the move to competition 'in Internet time' (Cusumano and Yoffie 1998). This is reflected in practices that emphasise constant and honest communication between disciplines and with the client, and a focus on the rapid implementation of a flexible feature set that is frequently tested through rapid iterations of the 'organically evolving' product. In this sense, the Agile paradigm is linked to the 'synch and stabilise' approach to development adopted by Microsoft in order to build its products (Cusumano and Selby 1995)

In this section we shall focus on two Agile methodologies, Scrum (Schwaber and Beedle 2002) and Extreme Programming (Beck and Andres 2005) which have become particularly popular with video game practitioners, briefly describing the way in which some of their key practices are purported to address the problems we have identified in the previous section (see Table 1 for a summary).

Prototype iterations in a short development cycle

Agile mandates the rapid implementation of working product features which can be integrated, tested and incrementally improved in a continuous cycle of iterations. Projects are split into short development cycles (denominated 'sprints' in the case of Scrum) at the end of which a team should deliver actual 'shippable' product functionality. Adopting this approach makes it possible to address uncertainties and risks early in development, to undertake short, focussed experiments with innovative features and to keep the team motivated by keeping the focus of work on the tangible output of their effort. The emphasis on delivering 'playable' features and 'vertical slices' of a game enables developers and their customers (publishers) to evaluate the quality (fun) of a growing video game from early stages of development, and tweak its design in order to address perceived shortcomings and follow up potential opportunities.

Focus on the user and honest communication

The features implemented in the course of Agile development should be relevant for the customer, who defines and prioritises them through, in the case of Scrum, the elaboration of a 'product backlog' listing all the features that she wants implemented in the product. Extreme Programming prescribes the redaction of 'user stories', defined as 'visible units of customer-visible functionality', and the adoption of 'test-first' programming, where the implementation of features is focussed by defining their minimal conditions of performance (this is, by establishing, as a first step, when would a feature be considered to be broken, and making it work).

Although the Agile approach enables rapid incorporation of user feedback into product development, it also emphasises the need for honesty when acting upon this feedback: Scrum allows and promotes modifications in the priorities of the contents of a product backlog, and the incorporation of new items in it in order to, for example, address changes in market conditions, but the product owner (in charge of managing the backlog) should communicate clearly the resource implications and trade-offs of these changes to the customer.

Internal communication and conflict resolution

Agile development is carried out by Small and collocated multidisciplinary teams. This facilitates communication between disciplines and the identification of contentious issues as part of the conversations and debates that take place during the workday. Scrum prescribes short, stand-up meetings at the beginning of every day as a way of ensuring that everyone in the team knows what everyone else is doing, and that any interdependencies between tasks are quickly identified. The exposure to issues from a range of disciplines also enables participants in the process to understand and internalise different points of view, and facilitates learning. Individual incentives are aligned though a strong focus on the delivery of product functionality at the end of the short development cycle.

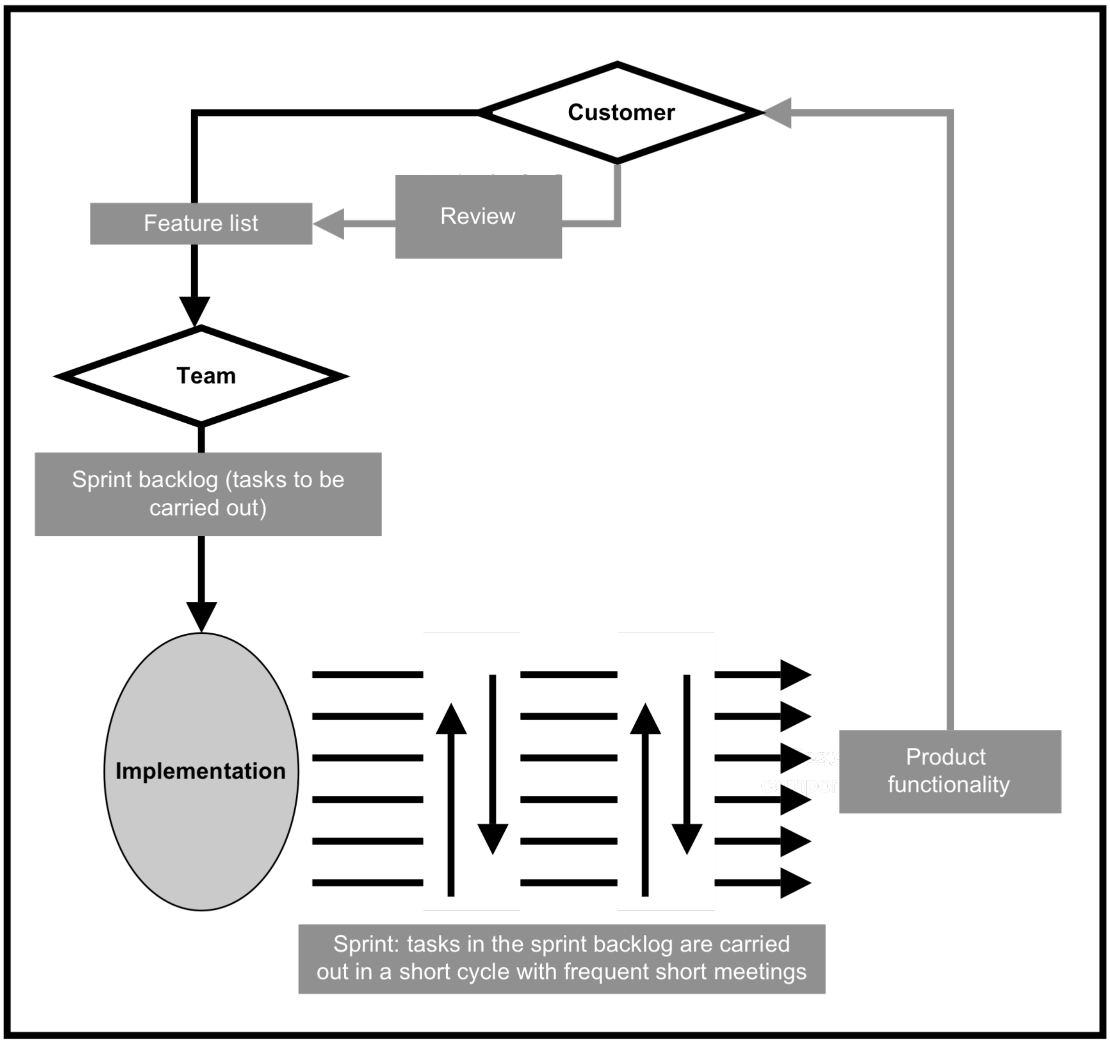
Pair Programming is another Extreme Programming practice aimed at promoting communication between members of a team. In this case, certain programming tasks are carried out by pairs of individuals who sit in front of the same computer screen, can correct each other mistakes, and brainstorm to solve problems creatively. These pairs are frequently rotated in order to favour dialogue between members of the team.

Team self-organisation

Agile places a strong emphasis on team parity. For example, in Scrum teams only one member has a 'formal' title as 'Scrum Master'. Her role is to ensure that the Scrum rules and practices are followed, and remove any impediments that might get on the way of the Scrum Team's progress, rather than to engage on traditional management of other members' activities. Another example of the collective nature of the development effort that Agile promotes is the adoption of 'shared code' practices in Extreme Programming, where anyone is allowed to change any of the product's code if this is deemed to be necessary. In the context of Agile, no member of the team is seen to have ownership over specific areas of the product being developed.

The goals of a Scrum sprint (this is, which features from the product backlog will be implemented) are determined by the Scrum team, and codified in a list of tasks (the 'sprint backlog') to which the team members sign up autonomously: although the product backlog establishes the set of goals of development, the team decides independently the path to follow in order to achieve them.

Figure 2- An agile development cycle


Customer
Review
Feature list
Team
Sprint backlog (tasks to be carried out)
Implementation
Product functionality
Sprint: tasks in the sprint backlog are carried out in a short cycle with frequent short meetings

3- Empirical section

a) Methodology

The empirical part of this paper is based on fieldwork conducted in 7 leading UK video game studios. We carried out semi-structured interviews with lead developers and managers of these organisations at their offices. Every interview was recorded, and its content analysed using Nvivo qualitative research software with the aim of identifying some key emergent themes in the areas of creativity and project management.

We were, when we undertook the interviews, aware of the growing popularity of Agile approaches to video game development and tried to elicit the opinions of our interviewees in the area. We found that three of the studios we were interviewing had already implemented Agile techniques. We focussed our interviews with informants in these teams on the reasons for this, as well as perceived advantages and shortcomings of the methodology. In this section we describe the experiences of these three studios, presenting them as case studies of the adoption of Agile approaches.

Having done this, we undertake a discussion of our findings where we also provide evidence from informants in other studios we interviewed. The goal of this effort is to map different approaches to the adoption of Agile, and its benefits and limitations in order to determine the extent to which the theoretical claims presented in the section above are supported by the concrete experiences of video game companies.

b) Case studies

Full embrace

Studio A is a small organisation (12 developers at the time of the interview) engaged in the creation of casual games for handheld platforms. Studio A has fully embraced the practices of Extreme Programming, and produces its games using a highly iterative incremental approach with internal releases of a playable version of the game every 48 hours. These versions are always highly robust because they are programmed using test-first techniques that focus on the production of code that will not crash.

The production cycle is of six weeks, at the beginning of which a set of 'visual stories' (features) are created and displayed in a storyboard. Members of the studio decompose these stories into tasks and implement them sitting in pairs. This approach is adduced to improve efficiency by reducing defects in code through peer review, while constant recombination of pairs facilitates the diffusion of specialist skills throughout the studio. It is also useful to ensure that developers work throughout the day: since they work with a peer, potential 'shirkers' have 'nowhere to hide'.

The studio is organised in an egalitarian fashion, all members have the same 'senior' position, base salaries and royalty share. Everyone contributes ideas to games and is informed about any relevant information for their work through daily morning meetings, where developers describe what they intend to do, and problems are flagged and resolved. Business negotiations are the only area where information is less readily available. One of the informants report that this is due to the fact that most members of the studio lack business skills, and it can be quite time-wasting to discuss these issues with them. At the time of the interview, there were plans to initiate a course to address this perceived shortcoming in the skill-set of the studio.

Studio A's relationship with its publisher is reported to be excellent. The publisher understands the studio and its approach to development. Trust is enhanced by open disclosure of information (including financial performance) to the publisher, as well as the availability of an updated, playable high quality version of the game under development at all times (which makes it possible for the publisher to assess the evolution of the project on a real-time basis).

Our informants at Studio A are highly satisfied with the outcomes of the adoption of Extreme Programming, and report that staff morale is high. They have not lost a developer yet, and have a large number of individuals wishing to join their team. They are also sceptic regarding the advantages of a less purist approach to the implementation of Extreme Programming, stating that the only occasions when the results of its adoption are disappointing is when studios do it in a 'wishy-washy' way.

Pick and mix

Studio B, which has traditionally been engaged in work-for-hire porting video games between platforms is currently working on an original Intellectual Property which includes several innovative features, particularly in the area of Artificial Intelligence (which determines the behaviour of a game's environment and its interaction with the player). Studio B has adopted Agile practices in certain areas. For example, an 'epic product backlog' has been set up in order to ensure that the game meets the expectations of its client. The use of this evolving document makes it possible for the client to define and prioritise the features she wants in the game from the point of view of functionality (e.g user experience), and the studio focuses on their technical implementation. There are, nevertheless, certain 'under the hood' features of the game, such as for example networking, which are being implemented and included in the game's schedule without being listed in this backlog.

Studio B has established a multifunctional Scrum team to implement the aforementioned Artificial Intelligence (AI) system. Several reasons are adduced for this, including the important interdependencies between different disciplines in this area: programmers need game levels to test the functionality of the AI, while the level designers need to design the levels around the AI functionalities, so it is desirable that they work together as one team. AI is perceived as a risk area, and the adoption of Scrum enables the team to deliver working functionalities on time, while adjusting the scope of the system depending on the evolution of the project. Finally, the adoption of Scrum in this area is seen as a learning experience potentially useful for other teams in the studio.

The Scrum operates following sprints for the implementation of functionalities specified in a product backlog owned by the game's lead designer and the game publisher. This backlog is co-ordinated with milestone schedules established using traditional component-based Waterfall methods. The composition of the Scrum team varies between sprints depending on the tasks that need to be undertaken, and developers with specialised knowledge are brought in for a sprint if required. This implementation of the practice breaks with the principles of Agile (which recommend team continuity), and exemplifies the pragmatic approach to Agile followed by Studio B. According to the game's project manager, video game studios should ensure they implement practices that are effective and address the needs of the project they are engaged with, which will vary depending on its characteristics. In this context, Agile is not a 'silver bullet' one size fits all methodology, but a toolkit with a potentially useful set of practices that will be more suitable in some contexts than others. One area where the Project Manager argues Agile could be particularly beneficial is when working in projects where the client is unsure about what she wants.

Members of Studio B expect Scrum to become more popular as successful studios and products demonstrate its effectiveness, but still expect other more traditional approaches to persist in particular areas where stability and predictability is at a premium. Regarding the Studio's own experience with its Scrum team, at the time of the interviews the approach had only been recently implemented, and it was still too early to evaluate its impacts.

Dysfunctional implementation

Studio C has used Scrum successfully in the past for the development of mini-game collections, as well as in a project exploring the possibilities of an innovative peripheral device. However, the implementation of the methodology in a recent, more ambitious project that constitutes this studio's first 'next-generation' effort has been challenging.

Attempts at adapting the Scrum methodology to a contractual framework based on the delivery of pre-established milestones were unsuccessful, and an initial failure to comply with the required feature list for the first milestone led to alarm regarding the future of the project. Scrum-style self-organisation and functionality-based product backlogs were replaced with a top-down process of low-level task specification with the goal of guaranteeing milestone deliveries.

The relationship between the studio and the publisher has been difficult, and the aforementioned concerns regarding the survival of the project have informed a management of the product backlog tolerant with the publisher's constant changes of mind and redefinition of the scope of the game. According to our informants, the embrace of Scrum's flexibility and responsiveness to customer demands has not been balanced with honesty communicating the trade-offs that this flexibility entails. The failure to fulfil unrealistic commitments at the high degree of quality expected by the publisher has only increased tensions and the feeling of constant panic in the studio.

The Studio had reorganised the project recently before we conducted the interviews with the aim of addressing these problems and increase team morale through a firmer adherence to the principles of Scrum. For example, Scrum teams have been redefined in order to reduce interdependencies between them, and focus more strongly on feature implementation including developers from different disciplines (previously they were very much organised functionally). The discipline leads have been designated Scrum Leaders in different areas and given training in the methodology. Higher management has met the publishers in order to agree a redefined scope for the project. The perception in the studio is that this management layer will ensure that the publisher is aware of the resource implications of future changes in the scope of the game from now on, leaving developers able to concentrate on the implementation of features in the product backlog.

Although some of our interviewees report that this shift has started bringing benefits to studio morale almost immediately, there are doubts about its applicability in certain areas. For example, one of the lead artists mentions his concerns about the extent to which self-organising teams can develop a coherent visual style for a game. Another informant is sceptic about the applicability of the Scrum model to the implementation of monolithic architectures. According to him, Scrum is based on self-organising teams working in parallel with as few interdependencies as possible, something that cannot be easily achieved in the case of non-decomposable systems such as video games.

The creation of multi-disciplinary teams can also become a potential source of tensions when adopted in a functionally organised structure: one of the worries that recur during our interviews has to do with discipline lead accountability over work undertaken by developers nominally subordinated to her but working in a Scrum Team outside hers. Again, this particular problem appears as a consequence of the need to implement the Scrum system on top of a pre-existing organisational structure initially established for a different development methodology.

c) Discussion

Our fieldwork highlights several issues that influence the outcomes of the implementation of Agile techniques for video game development:

First, our evidence shows that although Agile practices work well when adopted by small teams, scaling them in order to tackle larger and more ambitious projects presents important challenges. The Agile paradigm advocates team independence and empowerment, but this independence needs to be balanced with co-ordination when several teams are working on interdependent tasks, or integrated components. The prescriptions of Agile seem to assume the existence of one team developing a product in close contact with the customer. The presence of several teams raises the need for communication between them unless the system being developed is perfectly decomposable, something very uncommon in the case of video game artefacts. Determining product backlog owners internally, and making sure that information regarding changes in feature priorities and delivery schedule does not seem a trivial task and could expected to raise important governance challenges.

The strategy adopted by Studio B addresses these issues by applying Agile techniques in a very circumscribed area of development, and managing its interfaces with the rest of the team through the creation of an internal product backlog managed by the project lead designer.

This highlights a second finding of our fieldwork: the majority of our informants report that the Agile approach is particularly well suited to address uncertainty in innovative or exploratory projects. Its incremental, organic development strategy makes it possible to discover slowly the sort of game that needs to be created and reduces the risk of investing too many resources in the upfront design of an architecture that is found to be unsuitable when its components are integrated at a late stage of development. On the other hand, informants in Studio C state that Scrum's focus on game play and 'finding the fun' can lead to the neglect of a game's 'visual pay-off', and the delivery of a finished and polished product, which are key for publishers and market success.

The Agile approach challenges some basic rules that have historically regulated the interaction between video game studios and their publishers, particularly the traditional emphasis on (apparent) predictability codified in milestone deliverables requiring upfront planning and specification of development tasks. The fuzzy Agile approach, based on the delivery of key functionalities, with constant adjustment of scope in order to adapt to changes in the development conditions, and the 'carrying of features' over between development cycles depending on the circumstances might seem difficult to digest for certain publishers. The essential tension between delivering set-on-stone milestone requirements and the exploratory developmental approach favoured by Agile are illustrated in the case of Studio C.

4- Conclusions

The previous discussion illuminates some of the key challenges faced by video game studios as they strive for the right balance between predictability and flexibility in a highly competitive environment. The prescriptions of the Agile paradigm place a strong emphasis on achieving the former, in order to promote creativity and innovation without losing focus of customer needs. Although this approach has important advantages illustrated by the successful example of Studio A, our research also shows boundary conditions that might constrain its benefits, and even be detrimental for a project. The extent to which the Agile approach can be used in order to implement large scale projects requiring co-ordination between teams without incurring in the sort of managerial overhead that it initially purports to avoid constitutes a potential limitation for its application. This finding seems to support Frederick Brooks predictions about the future of software engineering: there are good practices that can be adopted in order to improve the efficiency of software processes, but no silver bullet to solve them definitely (Brooks 1995) .

Additionally, the apparent vagueness of Agile approaches regarding the delivery of a finished product with a pre-defined set of a features at a particular moment of time, and on an agreed budget limits its acceptability for customers. Although there is a perception of the problems of trying to predict the evolution of innovative projects with large lead times, following traditional methods with clearly established implementation plan seems to be a step in the right direction towards achieving those goals (Parnas and Clements 1986). On the other hand, Agile seems to eschew them altogether. This has led to the emergence of hybrid approaches which try to splice traditional scheduling and Agile development practices (Keith 2007) in order to reach a balance between the embrace of change required by the uncertain conditions in which video game development is undertaken, and the need to predict when (and for how much) will the product be delivered.

This trade-off can be transposed to the conceptual discussion in (Tsoukas and Chia 2002) regarding managerial approaches to the understanding of organisational change. It seems clear that the practices advocated by proponents of Agile constitute an inadvertent answer to Tsoukas and Chia's question regarding '*what must organisation(s) be like if change is constitutive of reality'*. Software practitioners have reached, through processes of experimentation in a production context, organisational solutions based on bottom-up emergent structures that favour a multiplicity of interactions (through iterations) both internal and external analogous to those discussed by Tsoukas and Chia, lending empirical support to their contentions. Conway's Law implies that these emergent organisational structures should be able to evolve (and evolve along) product architectures more adaptive to changes in the environment, and responsive to the surprises and downfalls faced during development processes strongly characterised by uncertainty. They might also facilitate processes of learning necessary in order to react to architectural innovations (Henderson and Clark 1990), some of which are discussed, in the context of video game development, in Sapsed, Mateos-Garcia and Grantham (forthcoming).

The emphasis on change, flexibility and structural emergence advocated by the Agile approach needs to be balanced with an acceptance of the fact that stability can in some cases be a goal that organisations should strive for. This appears so in institutional contexts where product delivery date and feature set inform promotional strategies and constitute key parameters in contractual negotiations, and also in production environments where the co-ordination of activities between different teams of functions will be very difficult to achieve without a certain degree of stability and predictability in their performance. While the former barrier to the acceptance of Agile Practices identified in our empirical context is being challenged by proponents of the approach who argue that its success at delivering high quality products on time and budget will eventually lead to a shift in perceptions both inside studios (where many still remain sceptical about its applicability) and publishers, the latter limitation seems to be intrinsic to the technological and informational conditions where video game development unfolds. It might require the identification of interdependencies between product components through upfront planning, and the co-ordination of team through top-down management practices in principle antithetical to the principles held by the Agile Paradigm.

Accepting that a nuanced approach to the implementation of Agile practices might be necessary, particularly in the case of large projects, and that the interactions with external parties such as publishers or contractors need to be managed carefully in order to avoid the emergence of dysfunctional processes such as those described in the case of Studio C has methodological implications: it would seem that in some cases the methodological shift that Tsoukas and Chia advocate is necessary in order to attain a fuller understanding of processes of organisational becoming, and associated dynamics of creativity and innovation might inform the design of structures and practices aimed at restraining them more efficiently.

Table 1- The Agile response to problems in software development

Problem	Agile response	Relevant practices (Scrum)	Relevant practices (Extreme Programming)	Relevant practices (common)
Plans are disrupted by emergent problems and unpredicted dependencies	Engage in constant iterations of the product from the onset of the development, organise the work in small teams in order to identify dependencies rapidly.	Sprints, daily scrum meetings.	Incremental design, frequent automated testing, weekly development cycles and daily builds, pair programming	small cross-functional teams, open working environment
Plans are disrupted by changes in the environment	The feature list is flexible and subject to change depending on feedback from the product iterations and new market information.	Product Backlog	Stories, customer involvement	
There is conflict between the goals and incentives of different stakeholders	Development is carried out by small teams whose members engage in constant communication, code is owned collectively and testing is not separated from implementation. Development team is insulated from external pressures by the management team.	Product Owner role, daily scrum meetings,	Pair programming, shared code	Small cross-functional teams
Trade off between routinisation and creativity	Team self-organisation, Features are prioritised, while the emphasis on the delivery of functionality keeps the team focussed.	Team definition of sprint targets, Product backlog,	Shared code, stories, test-first programming, customer involvement.	

References

Beck, K. and C. Andres (2005). Extreme Programming Explained. Boston, Addison-Wesley.

Beck, K., M. Beedle, et al. (2001). Manifesto for Agile Software Develpment.

Boehm, B. and R. Ross (1989). "Theory-W Software Project Management: Principles and Examples." IEEE Transactions on Software Engineering **15**(7): 902-916.

Brooks, F. (1995). The Mythical Man Month. Chicago, Addison-Wesley.

Caves, R. E. (2002). Creative Industries: contracts between art and commerce. Cambridge, Mass., Harvard University Pres.

Computerworld (2007). Extreme Programming Inventor talks about agile development. ComputerWorld.

Conway, M. (1968). "How do Committees Invent?" Datamation **April 1998**.

Cusumano, M. (2006). Fast & Flexible Software Development. M. S. S. o. Business.

Cusumano, M. and R. Selby (1995). Microsoft secrets : how the world's most powerful software company creates technology, shapes markets, and manages people. New York, Free Press.

Cusumano, M. and D. Yoffie (1998). Competing on Internet Time: lessons from Netscape and its battle with Microsoft. New York, Free Press.

Dijkstra, E. (1972). "The Humble Programmer." Communications of the ACM **10**: 859-866.

Grantham, A. and R. Kaplinski (2005). "Getting the Measure of the Electronic Games Industry: Developers and the Management of Innovation." International Journal of Innovation Management **9**(2): 183-213.

Henderson, R. M. and K. C. Clark (1990). "Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms." Administrative Science Quarterly **35**: 9-30.

Hobday, M. and T. Brady (2000). "A fast method for analysing and improving complex software processes." R&D Management **30**(1).

Hyman, P. (2007). "For Better or Worse: A Quality of Life Update." Game Developer Magazine(June/July 2007).

Keith, C. (2007) Scrum and Long Term Project Planning for Video Games. Gamasutra **Volume**, DOI:

McGuire, R. (2006). Paper Burns: Game Design With Agile Methodologies. Gamasutra.

Norman, D. (1988). The Design of Everyday Things. London, MIT Press.

Parnas, D. and P. Clements (1986). "A Rational Design Process: How And Why To Fake It." IEEE Transactions on Software Engineering **SE-12**(2): 251-257.

Paulk, M., B. Curtis, et al. (1993). "The Capability Maturity Model for Software." from http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr24.93.pdf.

Royce, W. (1970). Managing the Development of Large Software Systems. Proceedings of IEEE WESCON.

Schwaber, K. and M. Beedle (2002). Agile Software Development with Scrum. Upper Saddle River, NJ., Prentice Hall.

Somerville, I. (2006). Software Engineering. London, Pearson Education.

Tsoukas, H. and R. Chia (2002). "On Organizational Becoming: Rethinking Organizational Change." Organization Science **13**(5): 572-582.