DOI: 10.1145/1409360.1409387

BY PATRICK STACEY AND JOE NANDHAKUMAR

Opening Up to Agile Games Development

IN FAST-PACED BUSINESS ENVIRONMENTS like computer games, Agile would seem to be the appropriate style of software development. However, our study of three computer game studios revealed that game development does not deploy Agile methods as such, but rather it shares some of Agile's practices and values. We were intrigued by how agility was triggered in game development; triggers undocumented by proponents of Agile methods. This article distils our findings into guidelines for nurturing and enhancing agility in creative software organizations.

Game development is both inspirational and unpredictable. A game's features may never be fully known at the outset of a project, but emerge as the developers continually play-test it. This is because a commercial game must be fun, entertaining and compelling;[10] qualities that are only really assessable when a game is compiled and played. Agile methods would suggest that game developers write a test script. This is perhaps more feasible when building commercial task-oriented software, but a computer game has an added aesthetic dimension to it; not just in terms of the look and feel, but the game-play too, i.e. the rules and level of difficulty of the game. In this respect, a game also needs to be tested intuitively by the developers; it may functionally run, but is it fun?

As far as Agile developers are concerned, the issue is mostly about having working code.[6] In Agile methods such as XP,[2] testing is part of a programmer's everyday life.[2] In XP, testing takes place within the context of a small cycle or iteration alongside other activities such as analysis and design. Iterative development is believed to be largely responsible for enabling agility, as the team can react expeditiously to changes in the environment.[1] Such flexible practices are infused with Agile values of simplicity, communication, feedback, and courage,[6] and demand developers who can "hang out on the edge."[5]

In this article we present new findings on agility in game development, and what triggers it, based on a study of three computer games studios.

Three Computer Games Studios

We conducted studies at CGS (Singapore), Miko (Singapore) and Goo (London). All are well known game studios, and Goo is one of the world's largest developers of mobile games. CGS has developed mobile phone and PC games in association with studios in Europe, where their games are also widely distributed, and Miko is well-known throughout the Asia-Pacific region as being at the forefront of location-based mobile games. To honour confidentiality agreements we use pseudonyms to protect the names of the companies, their employees and games.

We pursued a research approach that allowed us to remain open-minded. So, instead of only asking prepared questions, we engaged in fluid interviews with developers at each company, with the aim of understanding their game development process and practices. At CGS we conducted twenty interviews with developers, including one group meeting, between January and April 2004,[9] with several follow-up visits up until July 2005. In our other two cases, we conducted thirteen interviews at Miko and six at Goo to similarly elicit and understand their game development process and practices. We draw on these two cases to elabo-

rate on what we found at CGS. We used a software package called *nVivo* to aid the analysis of the transcripts and field notes.[8] Our analysis brought out essential activities of game development, and helped us to understand how agility was provoked.

Game Development Process at CGS. The study at CGS mainly focused on the work practices of the managing director, a project manager, the lead programmer, a computer graphics (CG) programmer and an intern programmer; these people constituted the core development team (see Table 1). These and other game developers were located in two adjoining small offices, each with a mixture of engineers and artists who sat in rows in one office and around the perimeter in the other.

As we explored the game development process with each interviewee at CGS, we found they talked about one activity in particular, namely play-testing. The object here was not solely to identify bugs, but to evaluate the game-play experience. Evaluating the aesthetic appeal or 'fun factor' led to changes in the way the game was conceptualized, designed, and coded; a rapid feedback occurred between conceptualize, design, code and play-test. This meant the game development process was tailored almost daily, as in many Agile approaches.

Testing Within the Context of Play

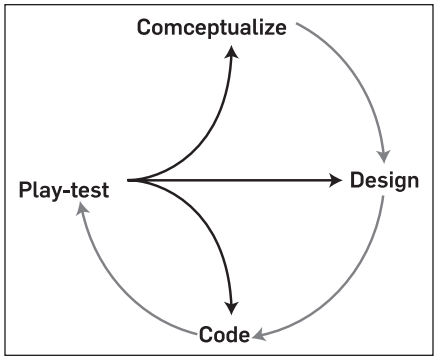
Since play-testing was afforded so much attention by the interviewees, we decided to investigate this and related practices further. Gradually we began to think of it in terms of a "boomerang" (see Figure 1).

The faint lines in Figure 1 indicate a weak flow of development activities, i.e. game development usually began with conceptualization following through design, code and play-test. The bold lines indicate the possible and more common trajectories the process could take once a game had been play-tested. For instance, if the game concept needed much re-working it would be "thrown back" to the conceptualization stage. If, however, the concept was deemed sound, but changes to the technical and/or aesthetic design were necessary, then the process would be thrown back to design, before subsequently picking up the original process trail flowing towards coding. Lastly, if there were bugs in the code, made obvious by such unexpected behaviors as cones not flying into the air when a car hits them (rigid shape physics errors), then the process was thrown back to the code stage, before being play-tested again. However, whichever of these activities occurred, the process destination was always play-test; rather like a boomerang returning to its thrower.

Table 1. Core Development Team at CGS

Alf	John	Richard	Mac	Angelina
Managing Director/ Game Designer	Project Manager/ Programmer/ Process Cop	Lead Programmer/ Technical Consultant/ Intern Overseer	CG Programmer	Programmer
Co-founder	*Recently joined from IBM*	*Co-founder*	*Co-founder*	*Intern*

Figure 1. Play-test boomerang



The CG programmer explained that sometimes code bugs inspired new creative directions. For example, the lead programmer found a place in the virtual world of the game Horizontal where his character could stand without being shot. Instead of seeing this as a bug however, it inspired the team to create an additional enemy that uniquely had the range to reach that part of the game-world. Before the boomerang was thrown back to the conceptualize stage however (see Figure 1), this candidate feature was discussed and evaluated.

At CGS, whether this bug-inspired feature was added to the game was ultimately the game designer's decision, since it had implications for the storyline, game-play and the very identity of the game. Due to Alf's business commitments outside the development context (in his capacity as managing director), he (as game designer) was largely unavailable to make a decision. This dependency temporarily stalled development. So, the developers began making their own design change decisions. This is reminiscent of the calls for champions in software teams3 and the Agile value of 'courage', such as, making bold decisions and taking responsibility for them.

So, while play-testing evoked a variety of possible creative directions for a game project, the team had to find ways of reaching in-house consensus on this and make a decision before throwing the boomerang again. CGS did not always have a customer with whom they could rely on to make decisions, contrary to many Agile projects. This was particularly the case when they developed original games, such as Horizontal, "we are the customer, we are gamers," said the lead programmer. Getting feedback is an important value in Agile development. The source of that feedback is the distinction in this case; in-house as opposed to customer.

Reaching Consensus on Play-test Results. The executive producer at Goo told us the best games they had made came about when the team communicated really well and fluidly, for example, "looking over someone's shoulder and saying hey that looks great." In this way, in-house consensus emerged naturally regarding how to improve the game. Such fluid communication is an important value in Agile development too. Weekly production meetings also helped keep the team on-page. If consensus regarding play-test next-steps could not be reached, then the team simply took a vote during these meetings.

Consensus emerged rather differently when the team reached a project milestone. Then, the whole company of 200 people were emailed a hyperlink that pointed them to where the game was stored. They were invited to play-test it and provide their feedback. We call this milestone play-testing. Milestone play-testing elicited no less than fifty different suggestions per milestone. This degree of feedback is be-

yond what those on some Agile projects would be accustomed to. It produced a buzz at Goo, which provoked random comments even from secretaries lighting a creative spark in the team sometimes. The challenge for the executive producer was to manage this large and diverse volume of feedback and forge consensus. His personality was instrumental in this; as one interviewee told us, "he has the expertise and the overall vision. And everyone respects that, everyone respects him."

Challenges to Play-test Consensus. Goo's executive producer could not always forge consensus easily and this could delay the development process. For example, positions on a game were sometimes taken along "floor lines", particularly between the marketing and production departments. At "Miko" a political dimension to the development process was also evident. The producer there told us how a game project was always subject to the whims of the management, as well as all the other stakeholders. In most of Miko's game projects, the managers and directors either vetoed features of the game or the entire project. The involvement of people outside the development team frustrated Miko's producer when they criticized his project. He was unable to leverage their feedback and was defensive.

Development Pressure. Such events at Miko could push some developers to resign. According to the lead programmer at CGS, a string of artists had joined and resigned from CGS. He said that this was more because they did not have the right knowledge-level, particularly with respect to mobile game development concepts, such as mobile information device profile. When developers resigned it took time to recruit freelancers, and so sometimes an engineer would take over the role of artist for a while; they had to adapt and improvise to keep the project moving. Viewed as job rotation, this is also visible in Agile development.

The Pathway to Game-Inspired Agility

We have described a number of practices in game development, which are reminiscent of Agile ones. We portrayed most of these in the Play-test boomerang diagram (Figure 1), but others include reaching consensus on how to refine a game after milestone play-testing (a variant on play-testing), and incumbents adapting to the situation when people resign. We now turn our attention to understanding what triggered these Agile practices and shed some light on the pathway to game-inspired agility.

Triggers of Agility in Game Development. As people from various departments "moved into" the development space, such as the 200 employees during play-testing at Goo, this provoked the developers into an intense evaluation exercise in which they had to respond to a large and diverse volume of feedback. In dealing with this situation they exhibited courage, self-belief, and initiative so they could move the project forward smoothly and quickly. So, the immense and diverse involvement in play-testing was a trigger for agility. The other side to this is the development team leader asked for this feedback; his openness and willingness to share his team's work with the wider corporate community was a trigger. The interplay of personality traits and organizational context produced agility. At Miko however, the attitude and personality of the producer led to rigidity and inertia; feedback had the opposite effect at Miko.

Conversely, with respect to people "moving out" of the development space, "a certain amount of staff turnover is good for the team"[2] such as when someone leaves who does not fit into the team. We found that as people resigned from CGS, the remaining developers were prompted to improvise around the absence created, which further involved taking the initiative and breaking their routines. This was evident as well when developers had to take the initiative with respect to design decisions in the absence of the game designer, and also at Goo and Miko where some of the developers attended industry events.

Nurturing Agile Software Practices

We now distil our findings into some guidelines to practitioners on nurturing agility. First, involve the development team in the wider corporate community, as well as involve this community in the (play) test/evaluation process. The Goo case demonstrated that by doing so, they were able to invite a large volume and diversity of useful feedback. Goo's development team leader welcomed this feedback, whereas the leader at Miko interpreted it as outsider interference. So, the personality of the development team leader is also an important part of nurturing agility; someone who welcomes criticism.

Second, cultivate a company culture that believes in and respects its development team. We saw at Goo in particular how much respect there was across the company for the team leader. This will encourage more constructive feedback. At CGS and Goo this was achieved by "star" developers attending meetings alongside the team leader, thereby opening the door on the team and building up the reputation of the team. Third, this also helps create an atmosphere where everyone can speak up. You never know where the next great idea is coming from – as we saw

Table 2: Guidelines for Nurturing Agility

Guideline	For Example
1. Involve the development team in the wider corporate community, and involve this community in the (play) test/evaluation process	The Goo case demonstrated that engaging the wider corporate community invited a larger volume and diversity of feedback
2. To successfully garner company-wide buy-in to the (play) test/evaluation process, cultivate a company culture that believes in and respects its development team such that feedback is constructive	Instead of thinking of programmers simply as implementers get them involved to some degree in other activities. At CGS and Goo this was achieved by "star" developers attending meetings alongside the team leader, thereby opening the door on the team and building up the reputation of the team
3. Create an environment where people will want to speak-up	Encourage a level playing field where everyone can speak up. You never know where the next great idea is coming from – as we saw at Goo, some even came from secretaries.
4. Encourage a healthy spirit of discontent such that people are not afraid of debate	This was evident from play-testing in general at Goo and CGS
5. Break-up some routines by getting developers out of their comfort zones	The breaking up of routines helps cultivate an open atmosphere. In the cases we examined, there were little in the way of routines. That is the nature of the play-test boomerang (figure 1); practices are tailored daily.

at Goo, some even came from secretaries. Fourth, this suggests nurturing a culture that is open and encourages a healthy spirit of discontent. A culture where people are happy to listen, discuss new ideas, and are not afraid of debate. We found evidence of this from play-testing at Goo and CGS.

Finally, to cultivate this atmosphere break up routines that can silo working groups. In the cases we examined, there were little in the way of routines; that is the nature of the play-test boomerang (Figure 1); practices are tailored daily. DeMarco and Lister would suggest the opposite, to seal-off the development team, "The top performers' space is quieter, more private, better protected against interruption."[4] However, we found that developers who were unprotected from disruption and worked in an open culture, benefited enormously from this – it made them more agile; we gave details of the way the Goo team responded to and indeed encouraged feedback. Even in studies of *business* agility it has been suggested that IT departments should not be siloed such that they have little interaction with the rest of the world.[7]

Table 2 provides a summary of the above guidelines for nurturing game-inspired agility. Our study of three game studios revealed triggers of agility not previously documented by proponents of Agile. These relate to the diversity and scale of play-testing feedback, staff turnover, industry events, as well as the personality of the development team leader, i.e. not being defensive but being asking for and being open to company-wide feedback. Agility then is really in the hands of the developers; they themselves can initiate it, and is produced during the interplay of personality traits and organizational context.

References

1. Baskerville, R., Ramesh, B., Levine, L., Pries-Heje, J., Slaughter, S. Is "Internet-speed" software development different? *IEEE Software*, 20, 6 (2003), 70-77.
2. Beck, K., *Embracing change with extreme programming. IEEE Computer*, 32, 10 (1999), 70-77.
3. Curtis, B., Krasner, H., and Iscoe, N. A field study of the software design process for large systems. *Comm. of the ACM*, 31, 11 (Nov. 1988), 1268-1287.
4. DeMarco, T. and Lister, T. *Peopleware:Productive Projects and Teams*. Dorset House, New York, 1987.
5. Highsmith, J. *Agile Software Development Ecosystems*. Addison-Wesley, Boston, MA, 2002.
6. Lindstrom, L. and Jeffries, R. Extreme programming and Agile software development methodologies. *Information Systems Management*, 24, 3 (2004), 41.
7. Melarkode, A., From-Poulsen, M., and Warnakulasuriya, S. Delivering agility through IT. *Business Strategy Review*, 15, 3, (2004).
8. Miles, M.B. and Huberman, M.A., *Qualitative Data Analysis*. Sage Publications, Thousand Oaks, CA (1994).
9. Stacey, P. and Nandhakumar, J. Managing projects in a games factory: Temporality and Practices. *In Proceedings of the 38th Hawaii International Conference on System Sciences, Organizational Systems and Technology Track, IT and Project Management*. (Waikoloa, HI, 2005), IEEE.
10. Swartout, W. and van Lent, M. Making a game of system design. *Comm. of the ACM*, 2003. 46, 7 (July 2003), 33-39.

Patrick Stacey (p.stacey@imperial.ac.uk) is a research associate at Imperial College of Business School, London, U.K.

Joe Nandhakumar (joe.nandhakumar@wbs.ac.uk) is a professor of information systems at Warwick Business School, University of Warwick, U.K.