Agile Essay Research

Successfully Scaling the Agile Methodology

Criticism of Agile

Scalability is a common Criticism of the Agile methodology.

Two Primary Scaling issues:

1. Large scope, complex projects
2. Large development teams

Complex projects

1. Requirement changes
2. Lack of precise initial designs
3. Mixing of Agile and Non-Agile teams

Large Teams

1. As teams grow larger, communication becomes more difficult
2. Lack of disciplined self-organisation
3. Scrum meetings become increasingly impractical

Possible Solutions

1. Scrum of Scrums
2. Lean Governance
3. Multiple Backlogs

Scrum of Scrums

One of the most common solutions for implementing the Agile methodology in larger groups.

1. Development divided among agile teams, each team has an dedicated Ambassador
2. The ambassadors from each team convene their own Scrum, communicating the progress of their own teams and discussing issues

Lean Development Governance

A framework for governing lean software development, created by Scott Ambler and Per Kroll of IBM.

Core principles of Lean Governance that can increase success when scaling agile development:

1. Risk-Based Milestones - Choosing high-value features to develop each sprint while aiming to reduce risk involved
2. Align Team Structure with Architecture - Team structure will heavily influence program structure
3. Staged Program Delivery - Each project in a program should choose their release dates and either meet them or skip them

Multiple Backlogs

A solution used by Nokia, utilising four levels of backlogs, along with a Scrum of Scrums.

Program Level:

1. Program Content Backlog - Top Level Requirements
2. Program Backlogs - User stories for each release of the program

Team Level:

1. Scrum Team Backlogs - User stories for each individual team
2. Sprint Backlogs - The tasks for each sprint

Relied on experienced managers to maintain long-term vision and ensure team cohesion.

Remaining Issues

Many scalability concerns that are not necessarily addressed by these suggestions.

In practice, some compromises may have to be made:

1. Lower expectations for change
2. Create basic architecture frameworks to provide a base to work off

Tomas

How to Start?

First of all I'm going to talk about the most common ways of making a project more pleasant to work on as well as more precise. For this part Mike Cohn's book *Agile estimating and planning* will help a lot. [**?**] His book does not only include explanations, but also pretty funny examples, which I may include.

What's Next?

While writing and discussing about importance of planning and estimating I will briefly talk about some estimating techniques [16] [21] and different plannings, which are release planning and iteration planning [**?**].

Agile Game Development

Continuing planning and estimating topics I will try and talk about that from game development side. Therefore, I'll include some experiments that show why a specific technique is more precise than any other [14][12].

Planning Poker

I will eventually mention planning poker not only as a precise estimating technique, but also as a very exciting and emotionally engaging [24]. Thus, it will increase the engagement level in the release planning process. Planning poker also solves some problems, which are quite an issue if one doesn't know how to deal with it [13].

Ryan

How can frequent play-testing during agile game development direct a project towards a more enjoyable product for the end user?

Agile software testing in a large-scale project

Focuses on implementing agile testing for a large government military project. It shows a cut by an order of magnitude the time required to fix defects, defect longevity, and defect-management overhead.

The agile process required a new mind-set at personal and organizational levels.

Communication between Developers and Testers in Distributed Continuous Agile Testing

Four main types of communication between tester and developer;

- Handover through issue tracker system
- Formal meetings
- Written communication
- Coordination by mutual adjustment

Early participation of the testers is very important to the success of the handover between testers and developers accomplished by attending planning, standups and review meetings.
Communication is not sufficiently effective through written communication and it must be augmented by informal communication.

Games User Research (GUR) for Indie Studios

For playtesting you must use your relevant target audience for the findings to confidently be applied to the game. Using a persona to match correct testers will make it easier to find the ones matching your target audience.

Having accurate data for not only the gameplay but also user facial expressions sound and actions outside of the game lets you more accurately messure what is enjoyable.

Developers being present at playtests to witness the players experience makes the devs more motivated to fix issues imediately and note what the player found enjoyable to provide similar experiences in the future

Using prototypes in early pervasive game development

This disscussed using "real" players and professional test players. Players who belong to the target group of the game usually provide more relevant data and are useful for understanding the players attitudes, opinions, and behavior. Using professional test players (e.g., colleagues) for testing enables faster iteration, and is also beneficial when new ideas are needed. If the prototype is very incomplete, it can be difficult for outsiders to understand, so it is useful to have both kinds of test players in the same project.

I have no words and I must design

Game design is ultimately a process of iterative refinement, continuous adjustment during testing, until, budget, schedule and management willing, we have a polished product that does indeed work beautifully, wonderfully, superbly. But your chances of getting that beautiful, wonderful, superb game will be much higher if you intentionality begin by thinking about the experiences you want your players to have, understand what makes a game, and understand what pleasures people find in them.

Louis

Question

1. Can learning be incorporated into production?
2. How fit is agile for learning?
3. Could it be better?

Can learning be incorporated into production?

- Yes – but not without its flaws
  - 'Ability to learn while working' itself is a vital quality to employers (Jussi Kasurinen et al., 2017)
  - Requires motivation
- Common issues that arise in group work include
  - Pressure
  - Scope
  - Issues in task setting

How fit is agile for learning?

- Focus on communication and interactions enables peer learning
- Iteration focuses on progress rather than goal
- Active task setting promotes autonomy
  - Autonomy considered a key factor in improving motivation (E. Deci and R. Ryan, 2012)

Could it be better?

- Consider adding 'what I want to learn today' to standups
  - Review and self-reflection has proven educational benefits (S. Edmunds & G. Brown, 2010)
- Promote 'working in pairs'
  - Some game companies suggest this helps learning and troubleshooting (M. Tran & R. Biddle, 2008)
  - Promotes active orientation–learning for teaching–which benefits learning (C. Benware and E. Deci, 1984)

Beren

Does Pair-Programming Impede Quality In Game Development?

1. Promotes Teaching
2. Time Constraints
3. Code Quality and Creativity
4. Flow
5. Impact Of Gender and Background
6. Required Resources
7. Team Engagement

Pair programming promotes teaching within the pair, even within different knowledge levels both individuals benefit from each others knowledge. Individuals may have knowledge on certain tools or software, that others dont and implementing a pair system may decrease costs within a department to pair individuals.
However pair engagement and bonding may have a detrimental effect to the pair as the navigator may feel left out as the driver completes the task with little communication between the pair.

Time may affect quality in a time restricted development cycle. Pair programming may take a few weeks to get used to and therefore in short development cycles have a detrimental effect as more programmer hours are used to produce the same task, however programmers that are used to pair programming, the Pair produces code much faster than solo programming. The pair still produces code slower than a solo programmer and this may reduce the quality of the product as it comes to crunch time.

The pair still produces code slower than a solo programmer and this may reduce the quality of the product as it comes to crunch time. Code quality generally is improved by pair programming however creativity often is lowered as pairs tend to follow traditional solutions Flow is often disrupted by pair programming This would impede programming time and have a detrimental effect on the overall quality of the product.

Flow is often disrupted by pair programming This would impede programming time and have a detrimental effect on the overall quality of the product. Gender and background may affect the pair as the pair may feel uncomfortable, this however may improve the future bonding of a team and generate better teamwork. Extra hours from pair programing may outweigh any of the positive outcomes within a company, this also may cause less money being spent on other parts of a project which may impede quality in a game

Scott W Ambler.

The agile scaling model (asm): adapting agile methods for complex environments.

Environments, 2009.

Scott W Ambler.

Scaling agile software development through lean governance.

In *Proceedings of the 2009 ICSE workshop on software development governance*, pages 1–2. IEEE Computer Society, 2009.

SW Ambler and P Kroll.

Applying agile and lean principles to the governance of software and systems development.

IBM Software Rational, 2009.

Andrew Begel and Nachiappan Nagappan.

Usage and perceptions of agile software development in an industrial context: An exploratory study.

In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, pages 255–264. IEEE, 2007.

Carl A. Benware and Edward L. Deci.

Quality of learning with an active versus passive motivational set.

21:755–765, dec 1984.

Mike Cohn.

Agile estimating and planning.

Pearson Education, 2005.

Greg Costikyan.

I have no words &.

The game design reader: A rules of play anthology, page 192, 2005.

Daniela S Cruzes, Nils B Moe, and Tore Dybå.

Communication between developers and testers in distributed continuous agile testing.

In *Global Software Engineering (ICGSE), 2016 IEEE 11th International Conference on*, pages 59–68. IEEE, 2016.

Edward L. Deci and Richard M. Ryan.

Motivation, personality, and development within embedded social contexts: An overview of self-determination theory.

pages 85–107, jan 2012.

Sarah Edmunds and George Brown.

Effective small group learning: Amee guide no. 48.

32:715–726, sep 2010.

Jussi Kasurinen et al.

What concerns game developers? a study on game development processes, sustainability and metrics.

In *2017 IEEE/ACM 8th Workshop on Emerging Trends in Software Metrics (WETSoM)*, pages 15–21, Buenos Aires, Argentina, jul 2017.

Simon Grapenthin, Matthias Book, Thomas Richter, and Volker Gruhn.

Supporting feature estimation with risk and effort annotations.

In *Software Engineering and Advanced Applications (SEAA), 2016 42th Euromicro Conference on*, pages 17–24. IEEE, 2016.

James Grenning.

Planning poker or how to avoid analysis paralysis while release planning.

Hawthorn Woods: Renaissance Software Consulting, 3, 2002.

Nils Christian Haugen.

An empirical study of using planning poker for user story estimation.

In *Agile Conference, 2006*, pages 9–pp. IEEE, 2006.

Maarit Laanti.

Implementing program model with agile principles in a large software development organization.

In *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*, pages 1383–1391. IEEE, 2008.

Kjetil Molokken-Ostvold and Nils Christian Haugen.

Combining estimates with planning poker–an empirical study.

In *Software Engineering Conference, 2007. ASWEC 2007. 18th Australian*, pages 349–358. IEEE, 2007.

Naeem Moosajee and Pejman Mirza-Babaei.

Games user research (gur) for indie studios.

In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 3159–3165. ACM, 2016.


Elina MI Ollila, Riku Suomela, and Jussi Holopainen.

Using prototypes in early pervasive game development.

Computers in Entertainment (CIE), 6(2):17, 2008.


Donald J Reifer, Frank Maurer, and Hakan Erdogmus.

Scaling agile methods.

IEEE software, 20(4):12–14, 2003.


David Talby, Arie Keren, Orit Hazzan, and Yael Dubinsky.

Agile software testing in a large-scale project.

IEEE software, 23(4):30–37, 2006.


Ritesh Tamrakar and Magne Jørgensen.

Does the use of fibonacci numbers in planning poker affect effort estimates?

2012.


Minh Quang Tran and Robert Biddle.

Collaboration in serious game development: a case study.

In *Future Play '08 Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, pages 49–56, Toronto, Ontario, Canada, nov 2008.

Dan Turk, Robert France, and Bernhard Rumpe.

Limitations of agile software processes.

arXiv preprint arXiv:1409.6600, 2014.

Jason Yip.

Hands-on release planning with poker chips.

In *Proceedings of the 14th Conference on Pattern Languages of Programs*, page 11. ACM, 2007.