# Agile Methods: Crossing the Chasm

Frank Maurer
*Department of Computer Science,*
*University of Calgary*
*Calgary, Alberta, Canada*
*maurer@cpsc.ucalgary.ca*

Grigori Melnik
*Department of Computer Science,*
*University of Calgary*
*Calgary, Alberta, Canada*
melnik@cpsc.ucalgary.ca

## Abstract

*An armada of emerging agile methods of software development (with eXtreme Programming and Scrum being the most broadly used) is both gaining popularity and generating lots of controversy. This high-level tutorial gives an overview of agile methods and provides background to understand how agile teams are addressing modern software engineering challenges. Analysis of empirical evidence is used to discuss strengths and limitations of agile methods in various contexts. The participants are introduced to the innovation diffusion models and environments, and discuss what is needed for agile methods to cross the chasm and move into the mainstream of software development.*

## 1. Tayloristic foundations of software engineering

The principles of software engineering formulated by the Waterfall methodology and its variants are strongly rooted in the Tayloristic paradigm. Taylorism, or “scientific management” – a movement in management theory introduced at the turn of the 20th century – represented ambitious ideals: increased productivity through the application of quantitative analysis, scientific method and appropriate procedural modifications, and thereby reduced the cost of production while being able to increase the wage paid to the workers. Applied to software engineering, Taylorism promotes a strong conformance to a plan through upfront requirements gathering and upfront systems design. It also encourages strict division of labor and the use of role-based teams (of business analysts, system architects, programmers, testers etc.) These factors plus the reliance on repeatability of the process are the main reason Tayloristic methods are failing in software development [3].

## 2. Agile movement

A more humanistic and collaborative approach to software development is “agilism”, which promotes “individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan” [1]. It is recognized, that while there is value in the items on the right, the items on the left are valued more. Essentially, all agile methods encourage continual realignment of development goals with the needs and expectations of the customer.

## 3. Main agile practices

An overview of the main practices of individual methods (including eXtreme Programming [2], Scrum [3], and Lean Programming [5]) is given. Many of the practices are not fundamentally new (for example, software inspections were introduced in 1970s; rapid prototyping – in 1980s). A conceptual foundation of how these practices meld together and enhance these “older” practices is discussed. Some empirical evidence is analyzed and the application of agile methods is elaborated by discussing their strengths and limitations in the context of existing research and industrial cases.

Among other topics, the tutorial addresses the issues of knowledge sharing (in agile and Tayloristic teams), project management, quality assurance, and social aspects and implications of agile methods. The facilitators focus on value and people to help software development teams achieve higher velocity and deliver superior value to the customers.

## 4. Crossing the chasm

Several innovation diffusion models and environments that promote innovation and lead to sustainable leadership are presented. These include Rogers' Technology Adoption Lifecycle [6], which defines five categories of adopters of innovation based on the normal distribution: innovators, early adopters, early majority, late majority and laggards. Moore goes further and argues for the existence of deep and dividing gap (chasm) that lies between early market success with visionaries and the mainstream acceptance by more pragmatic adopters [4]. To achieve market success with an innovation, the originator must find some way of leaping over this chasm (Figure 1) so that the process becomes mainstream. In facilitators' opinion, agile methods are at the edge of this chasm from early adopters to the mainstream of software development. The tutorial participants will be engaged in a discussion of what contemporary actions are needed to cross this chasm and to supersede the outdated relic of Taylorism in software engineering.

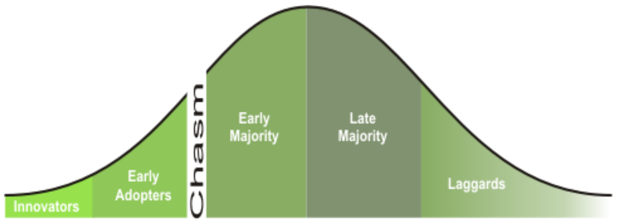

**Figure 1. Diffusion of Innovation.**

## 5. Topical outline

The tentative roadmap includes but is not limited to the following topics:

- I. Agile methods overview:
  - I.1. Motivation for agile methods: the current state of the industry;
  - I.2. Introduction to agile methods – defining agility: chaordic perspective, collaborative values and practices, barely sufficient methodology;
  - I.3. Agile methods vs. Tayloristic methods;
  - I.4. Why agile methods may work;
  - I.5. Sweet spots (for agile and Tayloristic);
- II. The big picture: crossing the chasm – getting to the mainstream.
- III. A survey of agile methods – practices/elements explained.
- IV. Knowledge sharing (agile vs. Tayloristic):
  - IV.1. Implications of Media Richness Theory;
  - IV.2. The role of externalised knowledge (documentation, experience factory etc.);
  - IV.3. Continuous knowledge sharing.
- V. Agile project management:
  - V.1. Business contracts;
  - V.2. Team formation;
  - V.3. Agile planning and estimation;
  - V.4. Agile project tracking and coordination;
  - V.5. Risk management.
- VI. Quality assurance:
  - VI.1. Quality-as-value-to-some-person perspective;
  - VI.2. The role of testing;
  - VI.3. Traditional QA vs. agile QA;
  - VI.4. Executable acceptance testing;
  - VI.5. Test-driven development;
  - VI.6. Dealing with para-functional requirements.
- VII. Initial empirical evidence.
- VIII. Conclusions.

## 7. Summary

Since all problems are different, all solutions and processes are situational and context-based. They depend on the context of the project and on the environment. Agile methods help to succeed in unpredictable environments, which are the reality today. They do it by encouraging continual realignment of development goals with the needs and expectations of the customer. Agile methods concentrate on significantly improving communications and interactions among all stakeholders, focus on "clean code that works", transparency, and relentless testing to achieve higher quality. Yet agile methods are not a silver bullet and agile practices only work in context.

## 10. References

[1] Beck, K et al. Manifesto for Agile Software Development, 2001. Online: http://www.agilemanifesto.org

[2] Beck, K. *Extreme Programming Explained – Embrace Change*, 2/e, Addison Wesley, Boston, MA, 2005.

[3] Beedle, M., Schwaber, S. *Agile Software Development with SCRUM*, Prentice Hall, NJ, 2001.

[4] Moore, G. *Crossing the chasm*. Harper Business, New York, NY, 1995.

[5] Poppendieck, M., Poppendieck, T. *Lean Software Development: An Agile Toolkit*, Addison-Wesley, Boston, MA, 2003.

[6] Rogers, E. *Diffusion of innovations,* 3/e. The Free Press, New York, NY, 1983.