Agile Process Myths

Elke Hochmüller
E.Hochmueller@cuas.at
Carinthia Univ. of Applied Sciences
Klagenfurt, Austria

Roland T. Mittermeir
roland@isys.uni-klu.ac.at
Klagenfurt University
Klagenfurt, Austria

APSO'08 Workshop
Leipzig, May 2008

Motivation

- Empirical evidence of agile methods in practice
- What are the actual benefits and risks of applying agile practices?
- Identification of challenging topics in agile software development
- From myths to issues ...

Myth #1: Agility helps in any case.

- ... for any kind of project / in every application domain:
  - projects of different size - scalability?
  - interactive versus technical systems
- ... every organization:
  - large companies with well-defined structures and processes
  - planning perspective:
    - process experimentation
    - process evaluation
    - process transition
    - process implementation (institutionalization)
  - co-existence of different types of processes?

Myth #2: Agility is for free.

- investments in new processes:
  - planning effort
  - adaptations to organizational structures
  - cognitive factors (commitment)
  - customer relationships
  - technical merits (techniques, tools)
  - education
  - people factor

Myth #3: Agility results in the right products.

- functional correctness
- focus on coding and customer-orientation:
  - no validated specification documents
  - validation at post-code level (customer involvement)
  - overburdened customer representatives
    - omniscient person/group
    - prioritization of requirements
    - tacit knowledge of colleagues
    - conflicting requirements
    - subjective views

  → multi-stakeholder problem!

Myth #4: Agility results in higher quality.

- fitness for use (quality requirements)
- better chances to react on time
- but: the multi-stakeholder problem still exists!
  - e.g. usability:
    - early detection of usability deficiencies
    - but: validation of usability will be rather subjective

Myth #5: Agility reduces time to market.

- early delivery of a basic subsystem
- small new releases
- delivery of complete system earlier in case of
  - agile processes (how to reason about completeness?)
    or
  - traditional processes
- short development cycles - major challenges:
  - cost estimation and control (different time horizons)
  - guarantee of continuous operation (continuous data migration)
  - synchronization of production and deployment (delayed feedback)

Myth #6: Agility copes with changing requirements.

- in fact, some requirements are highly volatile
- agile philosophy (allowing for changing requirements) tends to encourage slipshod requirements elicitation
  - risk of cost explosion (unnecessary rework)
  - unstructured code (steady code changes)

Myth #7: Agility avoids maintenance.

- focus on forward engineering
  - corrective maintenance
- design for evolutionary maintenance?
  - portability
  - changeability
  - flexibility
- ongoing evolution after final delivery?

Myth #8: Agility needs no design.

- agile philosophy:
  - no "big upfront designs"
  - no designs being "carved in stone"

  → systems with focus on user interface components
- other types of systems need stable architectural/design decisions:
  - data-oriented systems - risk of
    - steady schema evolution
    - data migration
  - view integration
    - multi-stakeholder problem
    - iterations unfeasible
- in what cases and to what extent is architectural/ systems design feasible during agile processes?

Myth #9: Agility needs no documentation.

- focus on coding with minimal documentation effort
- documentation not as objective per se
- types of required documentation depends on
  - kind of project
  - applied agile practices
  - external factors (customer requirements)
  - addressees (background, skills, ...)
- useful (implementation) documentation:
  - fluctuation within the development teams
  - protection against legacy problems

Myth #10: Agility decreases development effort.

- philosophy of minimizing design and documentation effort
  - as a consequence, reduced effort in forward engineering?
  - risk of substantial rework during system operation

  → saved effort might be shifted towards maintenance
- short development iterations may lead to the assumption that potential shortcomings can be detected and repaired as early as possible
  - valid for small projects with few and well-known components
  - is refactoring the right way to avoid maintenance?
  - what is the "real value" of refactoring?
- the challenge is to find the right balance between
  - the amount of planning effort essential at the early project state and
  - decisions which can be left open for developers to deal with

Myth #11: Agility is nothing new.

- only a new notion for reinvented old practices?
- other alternatives to classical phase-driven models:
  - evolutionary (incremental) prototyping
  - iterative development
  - incremental delivery
  - ...
- what is new:
  - process description

    AND
  - set of practices, techniques and "how to's"

Myth #12: Agility just happens.

- Or: "We all are agile."
- Distinction between
  - a slipshod practice of undisciplined hacking
    and
  - the process of applying a set of defined agile practices
- Application of an agile process has to be
  - carefully planned
  - carried out with care (in a disciplined manner)

Agile Processes - Issues

- For what kinds of projects may agile practices be beneficial?
- What facts may inhibit the success of applying an agile process?
- What has to be obeyed when trying to adopt and maybe adapt an agile method?
- ...

→ avoid wrong expectations

Critical Remarks (Lutz Prechelt)

Freie Universität Berlin

- Many slides implicitly equate "Agile" with eXtreme Programming (XP)
  - but many statements may be invalid outside of XP
- Many of the myths far exceed the claims of the Agile Manifesto
  - Agile experts would not support such statements
  - The manifesto should be used for a definition of the term "Agile"
  - These myths hence represent hype and need not be taken seriously
  - (However, as some people really believe these myths, they are worth discussing nevertheless)

Thank you!