Why OO-Developers like Agile Methods

- Personal Experiences and Insights -

Roland Trauter
DaimlerChrysler AG - Research Center Ulm
roland.trauter@daimlerchrysler.com
Personal experiences with „methods“:

1985  First experiences with rigid waterfall model: Functional decomposition, HIPO-Diagrams, Structured Programming, Entity Relationship Model, Database Design
1988  Short period with Structured Analysis / Structured Design, still waterfall model.
1989  Experiments with Rapid Prototyping, OOA, OOD, Smalltalk. **RAD discovered.**
1993  First real Smalltalk project with monthly delivery of evaluation prototypes. **RAD works!**
1997  Starting work on structure analysis tool **Ada Cross Referencer** with **RAD** using C++.
1998  Consulting for large defense project with German **V-Modell**, **UML** and C++.  
**Documentation-oriented process modell mixed with prototyping.**
2000  Starting work on dataflow analysis tool **DATAFAN** with **RAD** using C++.
2002  Discovering **Agile Methods renames RAD.**
What‘s new with Agile Methods (AM)?

AM is not my current field of work, but seen from the distance my impression is ....

- Tom DeMarco‘s Same Old Stuff Index is probably rather high?
- Mainly a marketing campaign, a new buzzword for good (old) concepts?
  „Good opportunity to write another book and organize another conference.“

Basically a Merger of: OO (Smalltalk) Gurus and Rapid Application Developers / Prototypers.

New Focus: The result-driven development process and the interaction of people.

The good news:
Successful sw-development is possible without highly formalized, plan-driven, documentation-oriented, mature processes.

Agile OO-Developers feel good, process model is recognized!
„AM“ Project Example 1 (1993):

Planning System for Electric Power Stations

• Electric power stations and a network of local distribution stations supply consumers with electricity.
• Each of these stations is typically an individual configuration of basic hardware components, and software components to control the system.
• These systems consist of thousands of components with complex relationships and dependencies.

Planning System Characteristics

• The system leads the planner through the whole planning process
  - with input forms (about 40) for each planning step.
  - with correctness checks for each input value.
  - with consistency checks for the overall configuration.
• The system generates the configuration files to install and control the power station.
“AM“ Project Example 1 (1993):

Planning System for Electric Power Control Systems

• Project team / Roles:
  - One person as customer / system user representative (User).
  - One project leader / system analyst (Analyst).
  - Two experienced Smalltalk developers.

• Work packages / Process:
  - Jan.-April: User with Analyst, collect and analyze the requirements, design of GUI / Screenshots.
    Analyst designs the data model, selects Implementation Tool Visualworks/Smalltalk.
  - May: Analyst with developers, decisions about architecture (VW MVC) and further process.
  - May- Dec: Monthly delivery of prototypes to user with successive extension of functionality,
    weekly implementation wrap-up, inline documentation, internal implementation reports
    and inspections. Monthly workplan based on user feedback with priorities.
  - Jan. 94: Successful of delivery of planning system to customer site.

Development process proved to be highly efficient!

- A compiler-independent tool for structural analysis of large Ada software assets.
- Easy navigation through all structure elements and their relationships.

• Application fields: Maintenance, Reengineering, Quality Assurance.

• Project team / Roles:
  - Project leader defining requirements, user interface design, tool tester.
  - Programmer / Architect (3 persons / one person at a time).
  - One main evaluator as customer / business unit representative.

• History / Process:
  1996 Need for static structure analyzer for large Ada sources came up in reuse project.
  
  Start with small functionality. Continuous extension + refinement based on customer feedback.
  Delivery cycles to customers: Bugfixes a few days, 3 to 9 months for extensions of analysis.
  Several redesigns / partial reimplementations: Type resolution, Ada83 to Ada95, portable GUI.

2003 Complete cross referencing for Ada 83 /95 with flexible list and diagram views.

Supplements development environment at MTU Aero Engines!

- A tool for automatic detection of severe errors (non-initialized variables) in Ada programs.

Reduced risk of system failure and cost reductions for quality assurance.

- Project team / Roles:
  - One customer (BU consultant) as tool visionary (idea / requirements).
  - One technical leader for solution concept and tool architecture.
  - Up to 4 persons for design, implementation and test.

- History / Process:
  2000  Agreement with customer on expected tool capabilities / requirements.
        Solution concept and proof-of-concept with Smalltalk prototype. Evaluation of existing tools.
        First tool-architecture with ASIS-Frontend. Team members works on separate modules / tasks.
        Restart of implementation with C++. Test environment for regression tests developed.
        One major architectural change through replacement of frontend.
        Delivery of new evaluation versions to customer varied between weekly and several months.
  2003  Ongoing integration with Ada Cross Referencer.

Promising feedback from first pilot project at EADS Military Aircraft.

The problem: Safety-critical and innovative, „historically grown“ software implemented with C.
The solution: Behaviour preserving restructuring and redocumentation to improve maintainability and extend system lifetime within a new architecture.

Project team / Roles:
- One customer representative trying to understand the software.
- Two experienced C++-programmers.

• Implemented XP-Practices:
  - Refactoring: Behaviour preserving changes for C-code.
  - Customer tests: Simulation on PC, test drives with vehicle every 1 or 2 month.
  - Collective Code Ownership: Repository with CVS accessible from several DC sites.
  - Test-Driven Development: using department-internal tool TestEnvironment.

Additional:

  Documentation requested by customer! Generated with doxygen.
Resume

• Each project is different. Development processes have to be flexible and adapt to a project's unique situation. Compiler runs are repeatable, a development process as a whole is not repeatable!

• Prototypes with a modular architecture are essential, they can become products step-by-step.

• Documentation is necessary as well, to be up-to-date it is generated from implementation.

• Small „agile“ projects (5 to 10 persons) can lead to better results than larger projects.

• Larger projects need more rigid processes with a lot of planning, documentation and administration overhead. They are less productive by their nature.

• Managers at MB vehicle development divisions refuse „AM“ for mainstream series development.

Expectations

• In practice there will be a mix between the planning and documentation dinos and the agile methods.

• Prototyping will be recognized as supplement of necessary planning and documentation activities.