Moving from a Plan Driven Culture to Agile Development

Lessons Learned

ICSE 2005, St. Louis, 20-May-2005

Michael Hirsch
Zühlke Engineering AG
hm@zuehlke.com
Moving from a Plan Driven Culture to Agile Development

Topic of this Talk

In scope:

• Experiences and lessons learned in moving from “plan driven” to “agile” development practices

Out of scope:

• Advantages (or disadvantages) of agile development
• When to use an when not to use agile development
• Differences between particular agile methods such as XP, Crystal, Scrum, agile RUP configurations, etc.
Speaker and Company Background

- Active in software development since 1981 in various roles, including developer, architect, project manager and mentor

- Background mostly in control systems, information systems, and embedded systems for various industries

- Partner and process manager with Zühlke Engineering in Zurich / Switzerland

- Zühlke Engineering:
  - Development of custom software and systems
  - Offices in Zurich, Frankfurt and London with approx. 250 employees
  - OOA/OOD/OOP since 1992, incremental and agile development based on RUP since 1998
  - Typical projects are 1 to 20 person years, with teams of 3 to 12 developers
Contents

1. Plan Driven versus Agile processes
2. The impact on stakeholders
3. Lessons learned
4. Summary
Fundamental Assumptions

Plan driven

All desired properties of the end product can be known and precisely specified \textit{before} construction of the end product begins.

It follows that projects are predictable and therefore can and should be planned in detail from start to end. A deviation from the plan is a sign of sloppy work in earlier stages of the project.

Analogy:
Building a Bridge

Agile

The desired properties of the end product \textit{cannot be known} until at least part of the solution is built.

It follows that projects are in principle unpredictable and the development process must be optimized for situations where the detailed outcome cannot be known in advance.

Analogy:
Exploring where and how to cross a river
## Plan Driven versus Agile

<table>
<thead>
<tr>
<th>Discipline Process Model</th>
<th>Plan Driven</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Planning</strong></td>
<td>Waterfall</td>
<td>Iterative &amp; Incremental</td>
</tr>
<tr>
<td><strong>Requirements Engineering</strong></td>
<td>Detailed project plan from start to end, created early in the project</td>
<td>Coarse grained plan for overall project, detailed plans per iteration</td>
</tr>
<tr>
<td></td>
<td>Dedicated specification phase</td>
<td>Requirements evolve over the course of a project</td>
</tr>
<tr>
<td></td>
<td>Initial requirements are signed off; rigorous change request regime afterwards</td>
<td>No or very relaxed change request regime</td>
</tr>
<tr>
<td></td>
<td>Comprehensive requirements documents, often part of a contract</td>
<td>Easy access to customer rather than reliance on comprehensive requirements documentation</td>
</tr>
</tbody>
</table>
## Plan Driven versus Agile

<table>
<thead>
<tr>
<th>Discipline</th>
<th>Plan Driven</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture &amp; Design</td>
<td>Comprehensive architecture and design specs before implementation begins</td>
<td>Minimum upfront architecture and design work</td>
</tr>
<tr>
<td></td>
<td>Architecture and design try to accommodate for future extensions</td>
<td>Architecture is validated iteration by iteration</td>
</tr>
<tr>
<td></td>
<td>Programming work concentrated in “Implementation” phase</td>
<td>Tradeoff between YAGNI and DOGBITE</td>
</tr>
<tr>
<td>Implementation</td>
<td>Programming work may be contracted out (preferably to low cost countries)</td>
<td>Programming work spread out over entire project</td>
</tr>
<tr>
<td></td>
<td>Programmers are assigned to subsystems</td>
<td>Pair programming (ideally)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Collective code ownership</td>
</tr>
</tbody>
</table>
# Plan Driven versus Agile

<table>
<thead>
<tr>
<th>Discipline</th>
<th>Plan Driven</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Testing</strong></td>
<td>Testing phase at the end of the project&lt;br&gt;Tests are designed and executed by test specialists</td>
<td>Testing spread out over entire project&lt;br&gt;Functional tests are specified and executed by end users</td>
</tr>
<tr>
<td><strong>Quality Assurance</strong></td>
<td>Formal QA role&lt;br&gt;QA role is responsible for formal and informal reviews, development process, configuration management, testing, code inspections, ....</td>
<td>No explicit QA role&lt;br&gt;No formal reviews&lt;br&gt;Attitude: quality is the result of how we work here</td>
</tr>
</tbody>
</table>
1. Plan Driven versus Agile processes
2. The impact on stakeholders
3. Lessons learned
4. Summary
How are Stakeholders affected by switching to Agile Development?
Moving from a Plan Driven Culture to Agile Development

Sponsor

- **Plan driven:** Buys a system with fixed functionality for a fixed price which will be delivered on a fixed date (so he/she thinks...)

- **Agile:** Buys a system without knowing all three of functionality, price and delivery date. Requires a lot of trust in development organization.

- **Typical objections against agile development:**
  - I can’t buy something if I don’t know exactly what it will be and how much it will cost
  - (Can I trust these newfangled agile hackers ?)

- **How to overcome objections:**
  - Appeal to track record of plan driven projects
  - Appeal to analogy with investment in shares
  - Deliver more than promised in all iterations
  - Work like in a glass house
Some Managers' perception of plan driven and agile methods

Plan Driven
Some Managers' perception of plan-driven and agile methods
Business Analysts & End Users

• **Plan driven:** Prepare specs at begin of project and then send them off to development team. Occasional questions from development team.

• **Agile:** Constant dialog with development team, participation in iteration planning and assessments, writing and executing test cases => more time needed!

• **Typical objections against agile development:**
  - I have no time
  - The developers should know this
  - (I don’t want to take responsibility for system scope)

• **How to overcome objections:**
  - Make sure BAs and EUs have the time they need
  - Appeal to “get it right the first time - less rework”
  - Communicate the customers role *before* the project starts
  - Assign BAs/EUs with authority (and willingness) to make decisions
### Minimum Customer Involvement for Agile Development

<table>
<thead>
<tr>
<th>Discipline</th>
<th>Customer Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Shared responsibility between customer and development organization. Customers role: (1) Must provide core requirements (2) May or may not participate in documenting requirements (3) Must review written requirements</td>
</tr>
<tr>
<td>Test</td>
<td>(1) Write and execute functional tests (2) Provide feedback to every development release</td>
</tr>
<tr>
<td>Configuration &amp; Change Mgmt</td>
<td>(1) Issue lightweight feature change requests</td>
</tr>
<tr>
<td>Project Management</td>
<td>(1) Assign priorities to requirements, change requests, and bugfixes (2) Participate in planning-, progress- and iteration assessment meetings</td>
</tr>
</tbody>
</table>
• **Plan driven:** Prepares project plan at begin of project. Occasional changes to the plan.

• **Agile:** Prepares a detailed plan for each iteration. Must keep 4 plans in his/her mind: overall project plan, plans of previous, current, and next iteration
  => places higher demands on project managers!

• **Typical objections against agile development:**
  - We can’t start unless we have all requirements in detail
  - Agile equals chaotic; agile projects can’t be controlled
  - (I hate uncertainties)
  - (I don’t like personal communication)

• **How to overcome objections:**
  - Most stated objections are based on misunderstandings
  - Most unstated objections are a sign that this person shouldn’t be a project manager, anyways
• **Plan driven:** Works on assigned subsystem(s). Heavy reliance on written interface specifications. State of personal work known to project manager (only).

• **Agile:** Heavy reliance on personal communication, state of personal work known to entire team, daily pressure to deliver working code

• **Typical objections against agile development:**
  - 2 - 4 weeks are too short to program something useful
  - (I don’t like that everybody on the team knows the state of my work)
  - (I don’t like personal communication)

• **How to overcome objections:**
  - Different skill levels of team members are OK, as long as everybody permanently works on improving his / her skills
  - People without a minimum willingness to communicate shouldn’t be software developers in first place
Contents

1. Plan Driven versus Agile processes
2. The impact on stakeholders
3. Lessons learned
4. Summary
Lessons Learned

1. The strongest motivator to switch to agile development is failure in previous projects

2. Expect resistance from Software Engineering Process Groups

3. Avoid “single stage” contracts

4. Don’t write off change requests
Lesson 1 - Motivators to Switch

The strongest motivator by far to switch to agile development is failure in previous projects.

- Case in point 1: Railway dispatch system Project aborted after 20 months of paper-based specification work and the contractor could still not come up with a reasonable estimate of overall project cost.

- Case in point 2: Income tax system Project aborted after 18 months of “classical” (meaning: very hierarchical) project management and insufficient communication among stakeholders.

- Other motivators: tight budgets, tight schedules, lack of alternatives (“last bet”), alternative to offshoring.
The CHAOS Authority Approves

Average cost overrun:
1994: 180%
2004: 43%

Source: Standish Group
www.standishgroup.com
Asked for the chief reasons project success rates have improved, Standish Chairman Jim Johnson says, “The primary reason is the projects have gotten a lot smaller. **Doing projects with iterative processing** as opposed to the waterfall method, which called for all project requirements to be defined up front, is a major step forward.”

Source: SoftwareMag.com

Source: Standish Group
www.standishgroup.com
Lesson 2 - Expect Resistance from SEPGs

Software Engineering Process Groups (SEPGs) in large organization are frequently opposed to agile development

Simply put:

• Large and heavy processes mean large and heavy SEPGs
• Lightweight and agile processes mean lightweight SEPGs with fewer members
• Most SEPG members have no hands-on experience in agile development
How to Overcome Resistance from SEPGs

- Resistance is founded in fear of loss of job
- Things to do:
  - Short term:
    Appeal to the “market value” of SEPG members
    What would you rather see in your CV:
    - XP / Scrum / Crystal / RUP specialist, or
    - specialist in company XYZ proprietary process ?
  - Long term:
    Reorganize the SEPG and have it staffed with active
    project managers and developers who serve part time
    in the SEPG. Rotate SEPG members over time.
Lesson 3 - Avoid “Single Stage” Contracts

The traditional procurement approach - write requirements specs, invite for tender, select (usually) cheapest offer - does not work for agile development

• This approach does not work because know-how is not efficiently transferred from the requirements writers to the team developing the system

• Worst form of this approach: WTO (world trade organization) contracts:
  - Company which prepared requirements specs is prohibited by law to bid for the project
  - Bidders have no personal contact with requirements writers, questions are formulated and answered in writing only
## Contracts for Agile Development

<table>
<thead>
<tr>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devel. Iteration</td>
<td>Devel. Iteration</td>
<td>Devel. Iteration</td>
<td>Transition Iteration</td>
</tr>
<tr>
<td>Transition Iteration</td>
<td>Transition Iteration</td>
<td>Transition Iteration</td>
<td></td>
</tr>
</tbody>
</table>

**Inception contract**
- time & material, 5 to 15% of overall effort

**Elaboration contract**
- fixed price or time & material, 15 to 35% of overall effort

**Development contract**
- fixed price, 50 to 80% of overall effort

**Milestones:**
- LCO = Lifecycle Objectives / LCA = Lifecycle Architecture / IOC = Initial Operational Capabilities / PR = Product release

**Lifecycle model:** Rational Unified Process (RUP)
Lesson 4 - Don’t Write Off Change Requests

Completely discarding change request processes would mean to throw out the baby with the bath water

- Without some form of change log, it is impossible to account for effort, cost and schedule at the end of a project
- Traditional “heavyweight” change request processes are not practical for agile development
- A lightweight change request process (“feature request process”) does not add overhead to a project and accounts for the turnarounds of a project
Lightweight Feature Request Process

1. Feature Request
2. Evaluate consequences
3. Explain consequences
4. Go / No go
5. Execute feature request

Customer

Development-O rganization

Feature Request Log
1. Plan Driven versus Agile processes
2. The impact on stakeholders
3. Lessons learned
4. Summary
Summary

• The biggest difficulty in moving from plan driven to agile development is dealing with objections, resistance and simply fear of the involved stakeholders.

• Expect 1 to 4 years to make the transition, depending on the size of the organization.

• Agile development is best viewed as an evolution of plan driven development and not as a radical departure from previous practices as seen by some advocates of agile development.
### Productivity Figures from Real World Agile Projects

#### Some Projects at Zühlke, 1999 to 2003

<table>
<thead>
<tr>
<th>Project</th>
<th>Technology</th>
<th>Total Effort [PD]</th>
<th>Total LOC</th>
<th>Total Classes</th>
<th>Total Use Cases</th>
<th>FPs delivered</th>
<th>Productivity [LOC/PY] [FP/PY]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airport MIS</td>
<td>Java J2EE, Swing</td>
<td>327</td>
<td>15’000</td>
<td>202</td>
<td>12</td>
<td>430</td>
<td>9’175 260</td>
</tr>
<tr>
<td>ATM Software</td>
<td>Win2k native, C++</td>
<td>2’940</td>
<td>128’000</td>
<td>2’180</td>
<td>16</td>
<td>2’415</td>
<td>8’705 165</td>
</tr>
<tr>
<td>Workflow Engine</td>
<td>Java J2SE, Swing</td>
<td>252</td>
<td>17’700</td>
<td>188</td>
<td>11</td>
<td>505</td>
<td>14’045 400</td>
</tr>
<tr>
<td>Product Data Mgmt System</td>
<td>Java J2EE, Web client</td>
<td>461</td>
<td>40’000</td>
<td>470</td>
<td>12</td>
<td>1’140</td>
<td>17’315 495</td>
</tr>
<tr>
<td>Protocol Converter</td>
<td>Java J2SE</td>
<td>215</td>
<td>26’400</td>
<td>507</td>
<td>9</td>
<td>725</td>
<td>24’560 675</td>
</tr>
</tbody>
</table>

PD = Person Days / PY = Person Years / FP = Function Points / LOC = (effective) Lines of Code
some things are perfect,
“others need engineering.

Thank you for your attention!

Questions?

Michael Hirsch
Zühlke Engineering AG
hm@zuehlke.com

some things are perfect,
others need engineering.
Additional Material
Moving from a Plan Driven Culture to Agile Development

Typical Traditional Change Request Process

Change Request Board

Customer

Development-Organization

1. Change Request
2. Offer
3. Order
4. Confirmation, Bill

zühlke

Moving from a Plan Driven Culture to Agile Development
Most organizations have a bad track record concerning estimation accuracy. The introduction of agile methods is a good opportunity to improve the estimation process, too.

- There is a systematic tendency to underestimate projects
- Underestimation (and overestimation) are not “free”
- Bad estimation practices are encouraged by typical company specific plan driven development processes
- The agile community has a more realistic view on the limits of estimation accuracy
Consequences of Estimation Errors

Non-linear impact due to planning errors, upstream defects, high-risk practices

Linear impact due to Parkinson’s Law

Target as a Percentage of Nominal Estimate

Cost

Effort

Schedule

Underestimation

Overestimation

< 100%

100%

>100%

Source: Construx
The Illusion of Predictability

Requirements for accuracy of estimates in typical company specific plan driven process models:

- Requirements for accuracy of estimates in typical company specific plan driven process models:
  - Requirements: +/- 20%
  - Design: +/- 5%
  - Coding: System integrated
  - System complete: +/- 5%
Moving from a Plan Driven Culture to Agile Development

Limits of Estimation Accuracy

Source: (Boehm 1995)
• Create estimates with statistical methods (e.g. COCOMO) early in a project, with “best case” and “worst case” scenarios
• “Bottom Up” estimate for each iteration

Life cycle model: Rational Unified Process (RUP)
Is Agile Development Here to Stay?

Energy Balance of a typical car engine

Of 100% of the energy in the fuel, only 19% is available at the wheels, the rest is wasted.
The “Energy Balance” of Agile Development

- Time and money invested into project
- Negotiating contracts
- Fighting over change requests
- Too much documentation
- Overspecifying requirements

Time left for building software
The “Energy Balance” of Agile Development

Time and money invested into project

My take at agile methods:
Agile methods are here to stay, because they are simply the most efficient (and cheapest) way to build innovative software systems

Overspecifying requirements

Time left for building software