Software Development Processes Today
Succeeding in Projects by Mixing Agile and Plan-Driven Approaches

zühlke
empowering ideas

Agile Software Development

Focus is on adapting change and „self-discipline"

Home Ground
- Projects with unstable or incomplete requirements
- Environments with low ceremony (tacit knowledge, informal communication)
- Teams are able to organize themselves and willing to do so
- Customer wants to be involved and wants to review intermediate results

Pitfalls
- Agility without discipline is "cowboy coding"
- Low-weight process definition:
  - Process must be steadily expanded and improved during the project
  - Expansions are based on feedback of iteration assessments
- Do not change process too often
- Involvement of customer is key success factor; ensure that he is sufficiently available

Agile software development processes must be **expanded**.

Expansion is a creative action. Expansion is carried out by development team itself.

eXtreme Programming (XP)

Characteristics
- Is more a collection of practices than a process
- People-oriented
- Based on 12 agile principles
- Based on 5 core values: communication, simplicity, feedback, courage and respect
- Pair programming
- Simple design
- Automated unit tests, test first approach

Pitfalls
- Needs many adaptations to be a complete process
- XP practices lived as a dogma
- Lack of customer involvement

Scrum

Characteristics
- Focuses on agile project management
- 15-30 days sprints (iterations)
- Daily scrum meetings
- Product owner prioritizes features
- Team defines set of features to be accomplished in next sprint (sprint goal)
- Sprint goal cannot change during sprint
- Burn-down chart for tracking progress

Pitfalls
- Team is not empowered to define sprint goals on its own
- Scrum master acts as project manager
- Tasks are assigned to individuals instead of team
- Requirements are not ready in necessary quality until sprint start
- Product owner cannot spend enough time on project

Agile Processes in Use

- Agile UP
- Crystal
- DSDM
- eXtreme Programming (XP)
- Feature-Driven Development
- Microsoft Solution Framework (agile version)
- RUP (tailored as agile process)
- Scrum

Process Equalizer

1. Select the optimal sound for your project

Set the levers of the five decision factors[1] (Size, Criticality, Team skills, Change, and Culture) according to your project's current situation.

2. Analyze the lever settings

All levers mainly at top, base on an agile process.

All levers mainly at bottom, base on a plan-driven process.

3. Fine-tune the process beat

Identify the lever(s) that deviate.
If required, select elements of other processes for these areas, e.g. from a plan-driven process if basing on an agile approach.

4. Repeat fine-tuning over and over again

Repeat these steps if the project's situation changes significantly.

Size

Number of people involved in the project.

3	▪ Direct communication ▪ Multiple roles per person
10	▪ Build up team culture and team spirit ▪ Team works together, collocated
30	▪ Define sub teams of 7±2 persons ▪ Informal inter-team communication
100	▪ Standardized reporting & controlling ▪ Formal inter-team communication ▪ Sub teams may work with agile approach; overall project works plan-driven
300	▪ Formalized project information ▪ Set up a project management office ▪ Verify continuously that the team understands the direction of the project

Criticality

Damage to the user due to impact of defect in the software.

Loss of comfort	▪ Informal acceptance procedure ▪ Little documentation ▪ Tester within the team
Discretionary money	▪ Basic risk management ▪ Formal acceptance of the risky parts
Essential money	▪ Tester within the team and autonomous external testing team ▪ High level of risk management ▪ Direct management reporting and involvement ▪ Documentation and tracing of decisions
Loss of life	▪ Very formal acceptance procedure ▪ Regulatory requirements

Team Skills

Percentage of developers able to work with a process (Level 1) versus developers able to design a process (Levels 2&3).

L 1	L 2&3	
10%	90%	▪ Control changes to process ▪ Do not optimize the process too often
30%	70%	
50%	50%	▪ Organize teams around the few level 2&3 people ▪ Empower level 1 people to achieve level 2
70%	30%	
90%	10%	▪ Engage external experts to define a process or to guide developers through the process

Change

Number of requirements that can or have to be collected at the beginning of the project and that remain unchanged until the final release.

10%	▪ Direct involvement of customer/user ▪ Short iterations ▪ Early and frequent releases
30%	▪ Create a prototype ▪ Backlog of features ▪ Prioritized features ▪ No change process, accept changes ▪ Fixed-price contract not feasible
50%	
70%	▪ Formal requirements ▪ Fixed-price contract possible ▪ Formal requirements change process
95%	▪ The only setting where waterfall approach is feasible

Culture

How well is agility accepted in your organization? Management and developers may have different stakes.

90%	▪ Adrenaline junkies ▪ Low ceremony ▪ Dynamic environment ▪ Few documents ▪ No formal procedures ▪ Flat hierarchy
70%	▪ Give enough freedom ▪ Avoid defining roles, policies and procedures
50%	▪ Define roles, policies and procedures ▪ Organization, project-specific process definition
30%	▪ Formal communication ▪ Formal procedures and artifacts ▪ Regulated environment ▪ Deep hierarchy ▪ High ceremony
10%	▪ Template zombies

Plan-Driven Software Development

Focus is on discipline in the meaning to „follow a plan"

Home Ground
- Projects requiring a certain degree of predictability and stability
- Environments requiring high ceremony (explicitly documented knowledge, formal communication)
- Projects where requirements can be agreed on upfront
- Projects based on fixed-price contracts or on regulations (FDA, MIL, …)

Pitfalls
- Discipline without agility is bureaucracy: cut off unnecessary artefacts and dead parts of the process
- Heavy-weight process definition:
  - Process must be tailored for specific project by project team (not by an organization-wide committee)
  - Becoming overwhelmed by vast amount of available templates
- Adapt and improve process during project; make iteration assessments
- A plan-driven process is not a waterfall process; it might be highly iterative

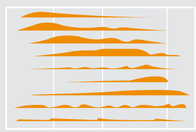
Plan-driven software development processes must be **tailored**.

Tailoring is difficult: if in any doubt, artifacts are not omitted. The process is often tailored for the biggest possible project, not for average projects the organization deals with most often.

Rational Unified Process (RUP)

Characteristics
- Framework to tailor software development processes
- Iterative and incremental development
- Covers almost all activities of software development
- Requirements are expressed as use cases and features
- Software architecture is of central importance
- Project management is risk oriented
- RUP is very well documented and a product of IBM
- Variations of RUP are available on the internet (e.g. OpenUP)

Pitfalls
- Misconceptions about RUP: no tailoring
- Document templates not adapted to project
- No serious iteration assessments
- Increments never demonstrated to customer



V-Modell® XT

Characteristics
- Verification and validation
- Highly adaptable to project conditions
- Modular system with self-contained process modules
- Supports both acquirer and supplier process
- Compliant with actual standards
- Change management
- Agile and incremental development supported by an additional execution strategy

Pitfalls
- Preparation of a consistent set of documents could become cost-intensive
- Overhead tends to increase continuously

Plan-Driven Processes in Use

- Hermes
- Microsoft Solution Framework (CMMI version)
- RUP (tailored as plan-driven process)
- V-Modell XT
- CMMI-based processes
- Waterfall-based processes

www.zuehlke.com/set-conference

[1] The decision factors are based on the book „Balancing Agility and Discipline, A guide for the perplexed" by Barry Boehm and Richard Turner, Addison-Wesley, ISBN 0-321-18612-5