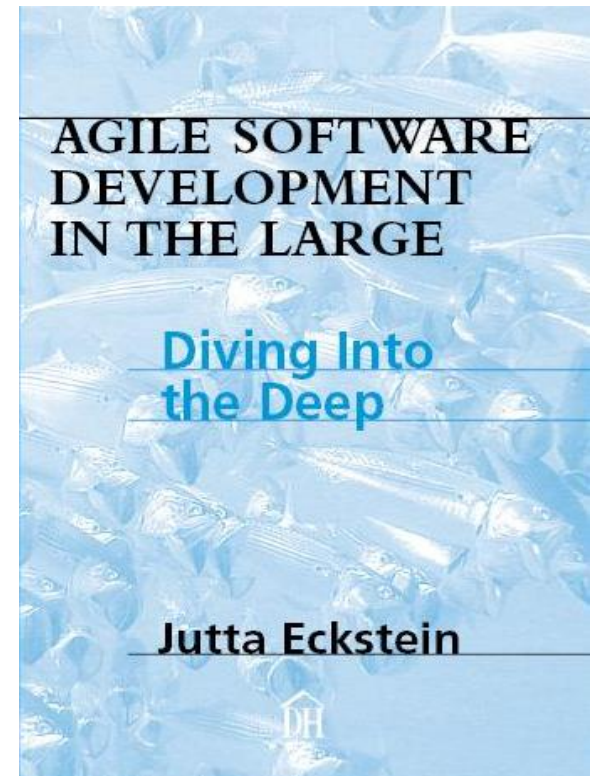


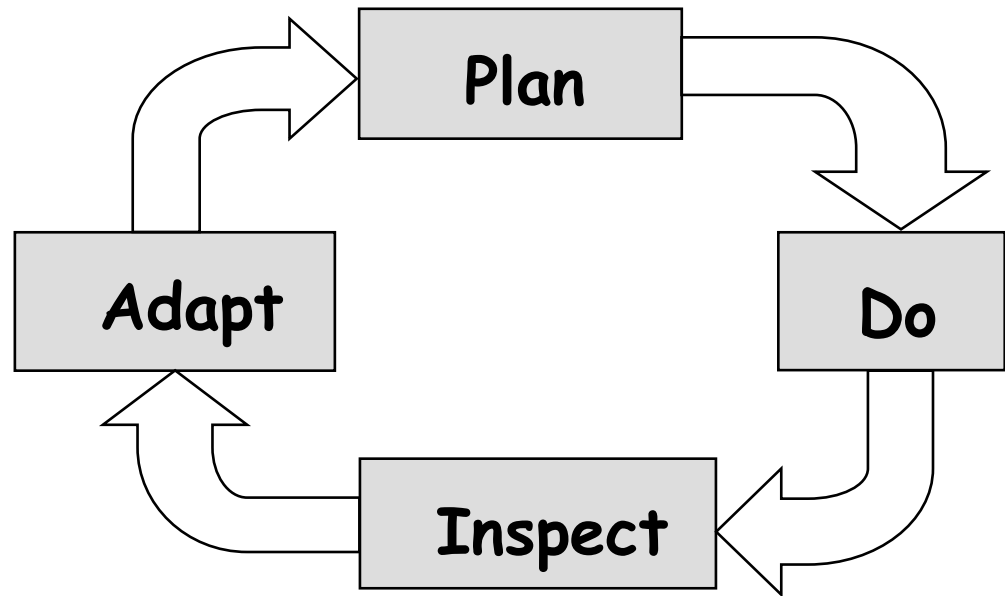
Planning, Estimating and Correction in an Agile Project

Jutta Eckstein
Encontro Ágil 2009



Agile Development

- Planning, estimating, correcting and thus - learning are ongoing activities
- Core of agility:



Gathering Features

Features

- aka (also known as)

- User stories
- Use cases
- Requirements

- Brief statements

- Of functionality
- From the user's perspective

⇒ **A feature creates a business value, which is measurable**

Defining Features

■ You can use this template

As a <user type>,

I want to <goal>

so that <reason>

■ It's ok to use a different form,

- But ensure that all features

- Provide a business value for somebody
- Are measurable

- Therefore each feature needs to clarify

- Who can accept it
- By what criteria

Product Backlog

■ Product backlog

- *"Consists of all work that can be foreseen for a product."* (Scrum)
 - Work is prioritized and estimated
 - More details when addressed
- Anyone can contribute
- Maintained and posted visibly
- One list for multiple teams
- Has a product owner

Product Owner

■ Responsible for

- Ensuring common vision for the project
- Establishing priorities
- Defining conditions of satisfaction for the iteration

■ Typically

- Marketing / product management
- User / analyst
- Project funder

⇒ **Represents the customer**

Estimating Features

Estimating Features

- **Estimate complexity**
 - And not duration
- **Relative estimates**
 - Feature A is twice as difficult as feature B
 - Feature C is equal difficult as feature A
 - ...
- **Estimation unit doesn't matter**
 - Ideal time
 - Feature points (might be easier)

Planning Poker

- **Collaboratively estimate features**
 - Everybody chooses from his deck of cards at the same time
 - Possible estimates:
 - 1, 2, 3, 5, 8
- **Clarify the features for coming to an agreement**
- **Split features if necessary**
- **Repeat estimating till you come to an agreement**

Iteration Planning

Iterations (aka Sprints)

- **Duration or: How often can you deliver?**
 - Are they short enough...
 - to deal with the uncertainty of the users need
 - to keep priorities unchanged
 - Are they long enough...
 - to accomplish something measurable

- **Examination**
 - Functionality
 - Estimates

How can this be planned?

- **Expectation:**
 - Plan - develop - deliver
- **Difficult:**
 - Activity-oriented planning
 - Not really measurable
 - Neither executable nor testable
 - Component-oriented planning
 - Based on dependencies of the components
 - Difficult to test regarding acceptance
- **Therefore:**
 - Result-oriented planning

Result-oriented Planning

- **Plan for accomplishing a business feature**
 - Accomplishment means delivery
- **Detailed planning only for the next steps**
 - Estimating and planning are continuous activities
- **Sustainable development**
 - Amount of work must match amount of time

Iteration Planning (aka Sprint backlog)

- Iterations are steered by features
- Fine grained iteration planning
 - Responsibility of developers
 - Plan for completing features
 - Estimate features by splitting them into tasks
 - Estimate only tasks that add value to the project
 - Estimate „natural“ tasks as long as they are not natural
 - Responsibility of product owner
 - Define acceptance criteria

Estimating Iterations

■ 1st iteration: project velocity is unknown

- How much time do we have available?

• Example:

- Length of iteration: 1 week, 3 developers = 15 ideal days (ideal time)

- Slowing down by 2,5 = 15 days / 2,5 = 6 real days (real time)

➔ Team accepts tasks for next iteration \leq real time

■ Estimating further iterations

- Team will have same velocity in the next iteration as in the previous one

Slicing Features

- **Plan for accomplishing a valuable feature**
 - Definition of done-done has to be defined
- **Sometimes the iteration seems to be too short for accomplishing a feature**
 - Don't split features into tasks (development steps)
 - Slice according acceptance criteria
- **Max size of a feature**
 - 2-3 people can complete it during an iteration
 - **Ideal: a team completes 5-10 features per iteration**

Planning Tools

■ Conservative tools

- Flip charts
- Index cards
- Excel

■ Sophisticated tools

- Trac (<http://trac.edgewall.org>)
- AgileTrac (<http://www.agile-trac.org>)
- PPTS (<http://ses-ppts.sourceforge.net/>)

Iteration Tracking

- Plans should be located at prominent place
 - Easy to access
 - Make progress visible
 - Read- and writeable by everyone
- Tracked in daily synchronization

Release Planning

- Reach the production release via several internal releases
- Internal releases are defined in terms of features or feature packs
- Coarse grained release planning
 - Steered by product owner
 - Marketing
- Shows big picture of the project

Course Correction

Team Reflection

Make progress explicit

- Show working software frequently
- Show team estimations and accomplishments

Retrospectives

- cover two levels:
 - development level: evolution of the software
 - meta level: inspection of the process

Course Corrections

- **Working software and retrospectives**
 - Continuous improvement of the plan
- **Development**
 - Working software provides direct and concrete feedback
 - Integration, quality, tests, customer acceptance, architecture
 - Continuous small course corrections
- **Process**
 - Regular retrospectives
 - Permanent reflections and optimization of the process
 - Quality improvement of estimates and plan

Dealing with Change

Planning Magnitudes

- **Four variables:**

- Time
- Scope
- Resource
- Quality

- **If the scope is able to change:**

- At least one other variable has to cope with it

Agility means Welcoming Change

- **From a team point of view**
 - Each iteration is steered by high priority features
 - No difference if these features are
 - Old, new, or have changed
- **Sustainable development**
 - Amount of work must match amount of time
- **Keep in mind:**
 - No changes during a iteration

Adjustment

Plan driven versus planning driven

- There is no such thing as a *final* plan at the beginning of the project
- Feedback has a direct impact on the plan

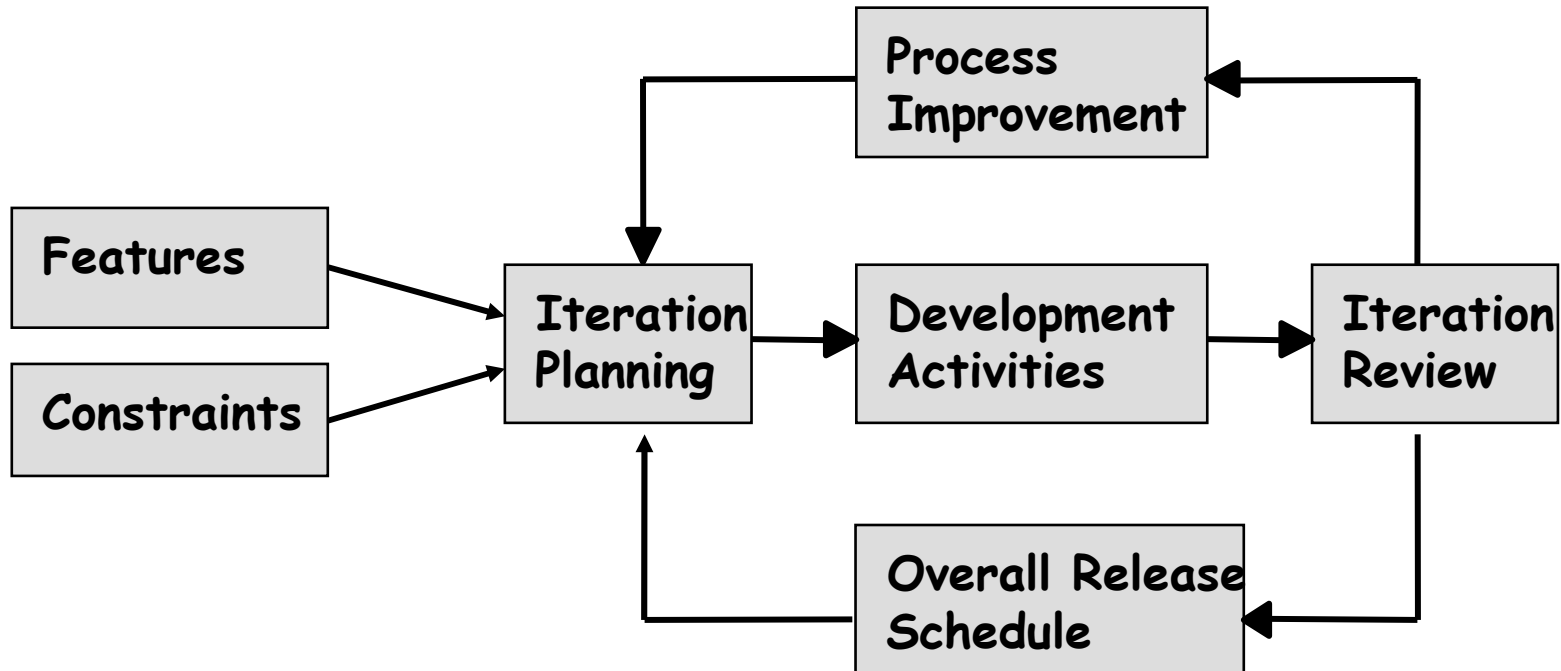
*„A plan is nothing;
planning is everything.“*

Dwight D. Eisenhower

Summary

Summary

- An iteration is a timeboxed development cycle
- Features are accomplished during an iteration



Starting an Iteration

- **Product Owner presents the features for the next iteration**
- **Team**
 - Estimates the complexity of the features
 - Splits the features into tasks
 - Estimates the tasks in hours
 - Checks the estimated tasks against availability and velocity
- **Commitment**
 - The team commits to the features

During an Iteration

■ Team

- Synchronizes via stand-up meeting
- Estimates remaining effort
- Performs development activities for delivery

■ Coach / Scrum Master

- Ensures the process
- Ensures impediments get solved

■ Product Owner

- Provides feedback on features
- Selects new features for upcoming iteration
- (Re-) prioritizes the backlog if necessary
- Clarifies acceptance criteria

Reflection - Ending an Iteration

■ Iteration Review

- Product owner revisits goals of the past iteration
- Team presents deliveries
- Product owner accepts / rejects achievements
- All revisit velocity

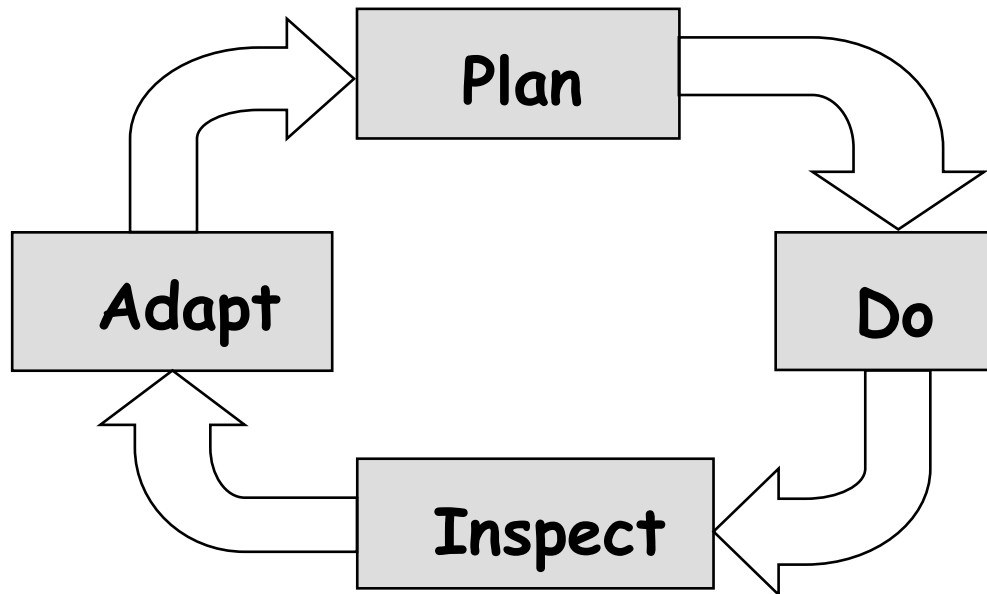
■ Retrospective

- Team reflects on (and defines an action plan on the top 3) for:
 - Things that worked well
 - Things that need to be improved
 - Things that are still difficult

Beware:

„If a project is on time and in budget that doesn't mean it was a successful project, but a successful estimate.“

Martin Fowler



Many Thanks!

Forthcoming book on:
**Agile Software with
Distributed Teams**

Contact information:
Jutta Eckstein
je@it-communication.com
www.it-communication.com

