Methodologies: Extreme Programming and Scrum

Introduction into Software Engineering
Lecture 23

Bernd Bruegge
Applied Software Engineering
Technische Universitaet Muenchen

Outline of the Lecture

- Examples of Methodologies
  - Extreme Programming
  - Scrum

XP (Extreme Programming)

- XP is an agile software methodology
  - Higher priority on *adaptability* (“empirical process control model”) than on *predictability* (“defined process control model”)
    - Change in the requirements is normal during software development
    - Software developer must be able react to changing requirements at any point during the project
  - XP prescribes a set of day-to-day practices for managers and developers to address this problem.

History of XP

- Original cast
  - Kent Beck, Ron Jeffries, Ward Cunningham (also created Wiki)
- Application of XP in the Chrysler Comprehensive Compensation project (C3 Project) in 1995
- Lots of initial excitement but later a lot of problems:
  - Daimler actually shut down the C3 Project in 2000 and even banned XP for some time
    - (See Additional References).

XP Day-to-Day Practices ("XP Mantras")

1. Get Rapid feedback for open issues
   - Confronting issues early results in more time for resolving issues. This applies both to client feedback and feedback from testing
2. Focus on simplicity, in particular in the design
   - The design should focus on the current requirements
   - Simple designs are easier to understand and change than complex ones
3. Incremental change
   - One change at the time instead of many concurrent changes
   - One change at the time should be integrated with the current baseline.

XP Mantras (continued)

4. Embracing change
   - Change is inevitable and frequent in XP projects
   - Change is normal and not an exception that needs to be avoided
5. Quality work
   - Focus on rapid projects where progress is demonstrated frequently
   - Each change should be implemented carefully and completely.

How much planning in XP?

- Planning is driven by requirements and their relative priorities
  - Requirements are elicited by writing stories with the client (called user stories)
- User stories are high-level scenarios or use cases that encompass a set of coherent features
  - Developers decompose each user story in terms of development tasks that are needed to realize the features required by the story
  - Developers estimate the duration of each task in terms of days
  - If a task needs more than a couple of weeks, it is further decomposed into smaller tasks.

How much planning in XP?

- Ideal weeks
  - Number of weeks estimated by a developer to implement the story if all work time was dedicated for this single purpose
- Project velocity
  - Inverse of ideal weeks
    - i.e., how many ideal weeks can be accomplished in fixed time
- Fudge Factor
  - Factor to reflect overhead activities (meetings, holidays, sick days...)
  - Also takes into account uncertainties associated with planning.

How much planning in XP?

- Stacks
  - The user stories are organized into stacks of related functionality
- Prioritization of stacks
  - The client prioritizes the stacks so that essential requirements can be addressed early and optional requirements last
- Release Plan
  - Specifies which story will be implemented for which release and when it will be deployed to the end user
- Schedule
  - Releases are scheduled frequently (e.g., every 1–2 months) to ensure rapid feedback from the end users.

Team Organization in XP

- Individual developers may write experimental prototypes experiments or proof of concepts, but not production code
- Production code is written in pairs (pair programming)
- Moreover, pairs are often rotated to enable a better distribution of knowledge throughout the project.

How much modeling in XP?

- No explicit analysis/design models
  - "Minimizes the amount of documentation"
  - "Fewer deliverables reduce the duplication of issues"
- Models are only communicated among participants
  - The client is the "walking specification"
- Source code is the only external model
  - The system design is made visible in the source code by using descriptive naming schemes
- Refactoring is used to improve the source code
  - Coding standards are used to help developers communicate using only the source code.

How much process in XP?

- Iterative life cycle model with 5 activities:
  Planning, design, coding, testing and integration
  - Planning occurs at the beginning of each iteration
  - Design, coding, and testing are done incrementally
  - Source code is continuously integrated into the main branch, one contribution at the time
  - Unit tests for all integrated units; regression testing
- Constraints on these activities
  - Test first. Unit tests are written before the component is written. They are written by the developer
  - When defects are discovered, a unit test is created to reproduce the defect
  - Refactor before extending the source code.

How much control in XP?

- Reduced number of formal meetings
  - Daily stand up meeting for status communication
  - No discussions to keep the meeting short
- No inspections and no peer reviews
  - Pair programming is used instead
  - Production code is written in pairs
- Self-organizing teams with a leader:
  - The Leader communicates the vision of the system
  - The leader does not plan, schedule or budget
  - The leader establishes an environment based on collaboration, shared information, and mutual trust
  - The leader ensures that a product is shipped.

Summary of the XP Methodology

Planning	Collocate the project with the client,write user stories with the client, frequent small releases (1-2 months), create schedule with release planning, kick off an iteration with iteration planning, create programmer pairs, allow rotation of pairs
Modeling	Select the simplest design that addresses the current story; Use a system metaphor to model difficult concepts; Use CRC cards for the initial object identification; Write code that adheres to standards; Refactor whenever possible
Process	Code unit test first, do not release before all unit tests pass, write a unit test for each uncovered bug, integrate one pair at the time
Control	Code is owned collectively. Adjust schedule, Rotate pairs, Daily status stand-up meeting, Run acceptance tests often and publish the results.

Scrum

- What is Scrum?
- History of Scrum
- Agile Alliance
- Agile Project Management
- Functionality of Scrum
- Components of Scrum
  - Scrum Roles
  - The Process
  - Scrum Artifacts
- Scaling Scrum
- Evolution of Scrum
- Conclusion

Introduction

- Classical software development methodologies have some disadvantages:
  - Huge effort during the planning phase
  - Poor requirements conversion in a rapid changing environment
  - Treatment of staff as a factor of production
- Agile software development methodologies
  - Minimize risk by making iterations very short
  - Focus on real-time communication, preferably face-to-face. This allows to minimize written documentation
  - www.agilealliance.org

Scrum

- Definition (Rugby): A Scrum is a way to restart the game after an interruption,
    - The forwards of each side come together in a tight formation and struggle to gain possession of the ball when it is tossed in among them
- Definition (Software Development): Scrum is an agile, lightweight process
    - To manage and control software development when change occurs rapidly (changing requirements, changing technology)
    - Based on improved communication and maximizing cooperation.

History of Scrum

- 1995:
  - Jeff Sutherland and Ken Schwaber analyze common software development processes
    - Conclusion: not suitable for empirical, unpredictable and non-repeatable processes
  - Proposal of Scrum
  - Enhancement of Scrum by Mike Beedle
    - Combination of Scrum with Extreme Programming
- 1996: Introduction of Scrum at OOPSLA
- 2001: Publication "Agile Software Development with Scrum" by Ken Schwaber & Mike Beedle
- Founders are also members in the Agile Alliance.

Manifesto for Agile Software Development

- http://www.agilemanifesto.org/
- Individuals and interactions are preferred over processes and tools
- Working software is preferred over comprehensive documentation
- Customer collaboration is preferred over contract negotiation
- Responding to change is preferred over following a plan.

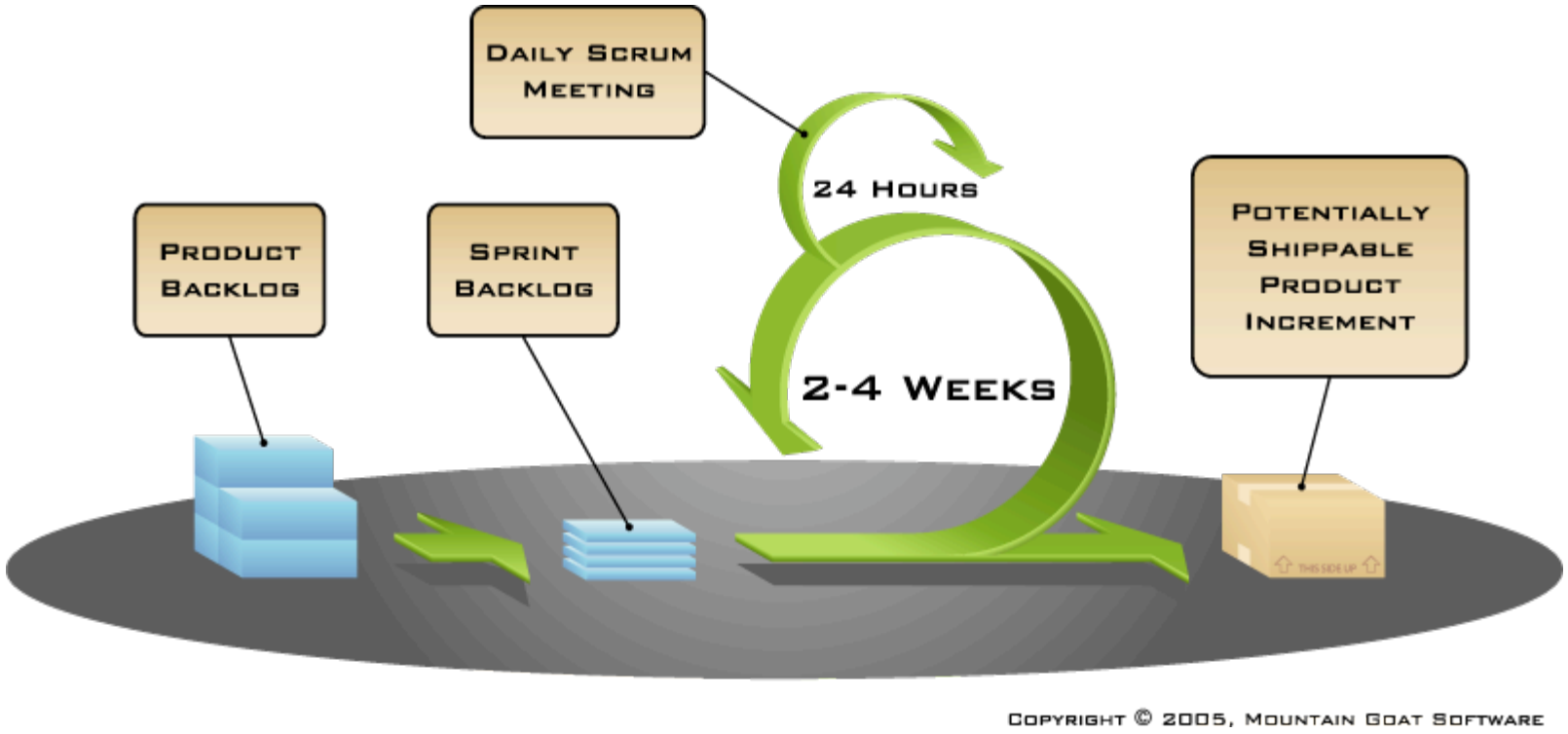
Methodology Issues

- Methodologies provide guidance, general principles and strategies for selecting methods and tools in a given project environment
- Key questions for which methodologies provide guidance:
  - How much involvement of the customer?
  - How much planning?
  - How much reuse?
  - How much modeling before coding?
  - How much process?
  - How much control and monitoring?

Scrum as Methodology

- Involvement of the customer
  - Onsite customer
- Planning
  - Checklists and incremental daily plans
- Reuse
  - Checklists from previous projects
- Modeling
  - Models may or may not be used
- Process
  - Iterative, incremental process
- Control and Monitoring
  - Daily meetings.

Overview of Scrum



Components of Scrum

- 3 Scrum Roles
  - Scrum Master, Scrum Team, Product Owner
- 5 Process Activities
  - Sprint Planning Meeting
  - Kickoff Meeting
  - Sprint (~~ Iteration in a Unified Process)
  - Daily Scrum Meeting
  - Sprint Review Meeting
- 3 Scrum Artifacts
  - Product Backlog, Sprint Backlog
  - Burndown Charts

Scrum Master

- Represents management to the project
- Typically filled by a project manager or team leader
- Responsible for enacting scrum values and practices
- Main job is to remove impediments.

The Scrum Team

- Typically 5-6 people
- Cross-functional (contains programmers, UI designers, testers, etc)
- Members are working full-time on the project
- The team has no leader, but is self-organizing
- Team membership can change only between sprints.

Product Owner

- Knows what needs to be build and in what sequence this should be done
- Typically a product manager

Scrum Process Activities

- Project-Kickoff Meeting
- Sprint Planning Meeting
- Sprint
- Daily Scrum Meeting
- Sprint Review Meeting

Project-Kickoff Meeting

- A collaborative meeting in the beginning of the project
  - Participants: Product Owner, Scrum Master
  - Takes 8 hours and consists of 2 parts (“before lunch and after lunch”)
- Goal: Create the Product Backlog.

Sprint Planning Meeting

- A collaborative meeting in the beginning of each Sprint
  - Participants: Product Owner, Scrum Master and Scrum Team
- Takes 8 hours and consists of 2 parts (“before lunch and after lunch”)
- Goal: Create the Sprint Backlog.

Sprint

- A month-long iteration, during which is incremented a product functionality
- No outside influence can interference with the Scrum team during the Sprint
- Each day in a Sprint begins with the Daily Scrum Meeting.

Daily Scrum Meeting

- A short (15 minutes long) meeting, which is held every day before the team starts working
- Participants:
  - Scrum Master (which is the chairperson), Scrum Team
- Every Team member should answer on 3 questions:

Questions for each Scrum Team Member

1. Status:

   What did I do since the last Scrum meeting?
2. Issues:

   What is stopping me getting on with the work?
3. Action items:

   What am I doing until the next Scrum meeting?

Summary

- XP and Scrum are agile software development methodologies with focus on
  - Empirical process control model
  - Changing requirements are the norm
  - Controlling conflicting interests and needs
- Very simple processes with clearly defined rules
- Self-organizing teams, where each team member carries a lot of responsibility
- No extensive documentation
  - Possibility for “undisciplined hacking”.

The end of the Tunnel

- Evaluation
- Final:
  - Organizational Issues
  - How to prepare for the final

Organizational Issues

- Admission requirements for final exam changed
  - Attendance criteria was dropped
  - Due to data loss in grundstudium tool
- Results of mini project available in the grundstudium tool
- List of students who have passed the admission requirements is available in the glass display in the waiting area opposite of my office (01.07.52/54)

Evaluation

- ☺ Examples and stories in the lecture
- ☺ The reverse engineering challenge
- ☺ Invited Talk
- ☺ The lectures were given in English
- ☹ The lectures were given in English
- ☹ Sound volume, audio problems in MW0001
- ☹ Structure of the lecture
- ☹ Don´t blame somebody for talking during the lecture;-)
- ☹ Exercises and mini-project too difficult (especially without Java experience)
- 😐 Other suggestions:
  - 😐 Publish slides in advance
  - 😐 Praktikum along with the lecture

Final Exam

- Date and Time:
  - 21st July 2007
  - 13:00-15:00
- Location: MW0001 und MW2001
  - Check the lecture portal for changes
- Resources:
  - No electronic devices allowed (Notebooks, etc.)
  - Closed Book

Preparing for the Final

- Review the lectures:
  - If a lecture takes 90 minutes, you should spend another 90 min to review the material
  - Read the text book, browse in the additional references, use Web search engines to search for terms
- Prepare with others, work in a team:
  - Practice the following categories of questions:
    1. Define a specific technical term introduced in the lectures.
    2. What is the difference between concept A and concept B?
    3. What are the pros and cons of concept A?
    4. Given a problem statement, create the corresponding UML model

Technical Term Questions

- Question: What is a methodology?
  - Answer: Lecture 21, Slide 5
- What is the spiral model?
- What is dynamic polymorphism?
- Define the strategy pattern:
  - Provide a textual answer and/or draw the pattern
- How do you map a UML class diagram into a table for a relational database?
- What is requirements engineering?
- What are the sub-activities of system design?

What is the difference between concept A and concept B?

1. What is the difference between defined process control and empirical process control?
2. What is the difference between a phase and an iteration in the unified process?
3. What is the difference between implementation and specification inheritance?
4. What is the difference between unit, integration and system testing?
5. What is the difference between Scrum and XP?
6. What is the difference between analysis, system design and object design?
7. What is the difference between requirements elicitation and analysis?

What are the pros and cons of concept A?

- What are the pros and cons of modeling?
- What are the properties of the waterfall model?
- What are the pros and cons of the empirical process control model?

Given a problem statement, create the corresponding UML model

- Hints:
  - Extract the use cases from the problem statement
  - Find the participating objects
  - Draw the class diagram
  - Use Abbot's technique
- If you run out of time, be pragmatic what you can do:
  - Provide us with a description or initial sketch of the model
  - Iterate on it a couple of minutes later again (maybe after having answered another question)
  - Don't waste your time on "cosmetic engineering"
- If you still run out of time, focus on one type of model, usually the class diagram.

Wanted: Web Designer in the Chair of Applied Software Engineering

- We are looking for a student who
  - has strong skills in web design
  - has basic experience with web programming-
  - is flexible and willing to learn about new technologies
  - can work 10 hours a week
- We offer
  - a fun job in a relaxed team
  - flexible working hours
  - great technical Apple based infrastructure
- Availability: Immediately
- If you are interested, contact Helma Schneider (helma.schneider@in.tum.de).