IF2261 Software Engineering

Software Process - Agile

Program Studi Teknik Informatika
STEI ITB

An Agile View of Process

- Represents a reasonable compromise between conventional software engineering for certain classes of software and certain types of software projects
- Can deliver successful systems quickly
- Stresses continuous communication and collaboration among developers and customers
- Embraces a philosophy that encourages:
  - customer satisfaction,
  - incremental software delivery,
  - small project teams (composed of software engineers and stakeholders),
  - informal methods, and
  - minimal software engineering work products
- Stress on-time delivery of an operational software increment over analysis and design

** SEPA 6th ed, Roger S. Pressman*

Manifesto for Agile Software Development

- Proposes that it may be better to value:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
- While the items on the right are still important the items on the left are more valuable under this philosophy
- Note: although most practitioners agree with this philosophy in theory, many pragmatic issues surface in the real world that may cause items on the right to be as important as items on the left

** SEPA 6th ed, Roger S. Pressman*

12 Principles

- Highest priority: user satisfaction
- Welcome changing requirement
- Deliver working software frequently
- Business people and developers work together daily
- Build around motivated individuals
- Face-to-face conversation
- Working software: primary measure of progress
- Promote sustainable development
- Continuous attention to technical excellence and good design
- Simplicity is essential
- Self-organizing team
- Tune and adjust team behavior at regular intervals

3 Key Assumptions

- It's difficult to predict which requirement will change
- Design and construction are interleaved
  - should be performed in tandem
- Analysis, design, construction, and testing are not as predictable as we might like

→ unpredictability

How to manage unpredictability?

- An agile process must be adaptable
  - adapt to rapidly changing project and technical conditions
- An agile process must adapt incrementally
  - requires customer feedback
  - software increment must be delivered in short time

Human Factors

Key traits:
- Competence
- Common focus
- Collaboration
- Decision-making ability
- Fuzzy problem solving ability
- Mutual trust and respect
- Self-organization

Agile Process Models

All agile process models conform to the manifesto and the principles

Examples:
- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Dynamic Systems Development Method (DSDM)
- Scrum
- Crystal
- Feature Driven Development (FDD)
- Agile Modeling (AM)

** SEPA 6th ed, Roger S. Pressman*

Extreme Programming

- Relies on object-oriented approach
- Key activities:
  - Planning (user stories created and ordered by customer value)
  - Design (simple designs preferred - KIS, CRC cards and design prototypes are only work products, encourages use of refactoring)
  - Coding (focuses on unit tests to exercise stories, emphasizes use of pairs programming to create story code, continuous integration and smoke testing is utilized)
  - Testing (unit tests created before coding are implemented using an automated testing framework to encourage use of regression testing, integration and validation testing done on daily basis, acceptance tests focus on system features and functions viewable by the customer)

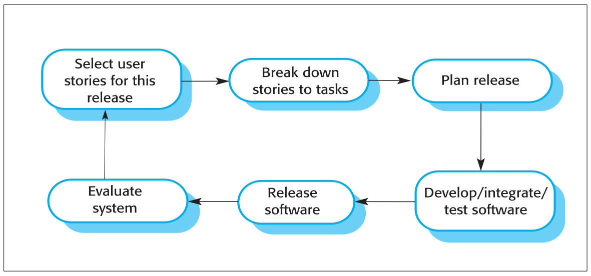
** SEPA 6th ed, Roger S. Pressman*

Extreme programming (2)

- Perhaps the best-known and most widely used agile method.
- Extreme Programming (XP) takes an 'extreme' approach to iterative development.
  - New versions may be built several times per day;
  - Increments are delivered to customers every 2 weeks;
  - All tests must be run for every build and the build is only accepted if tests run successfully.

** Software Engineering 7th ed, Ian Sommerville*

The XP release cycle


Select user stories for this release
Break down stories to tasks
Plan release
Develop/integrate/ test software
Release software
Evaluate system

** Software Engineering 7th ed, Ian Sommerville*

Extreme programming practices 1

Incremental planning	Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development ŌTasksÕ
Small Releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple Design	Enough design is carried out to meet the current requirements and no more.
Test first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

** Software Engineering 7th ed, Ian Sommerville*

Extreme programming practices 2

Pair Programming	Developers work in pairs, checking each otherÕs work and providing the support to always do a good job.
Collective Ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers own all the code. Anyone can change anything.
Continuous Integration	As soon as work on a task is complete it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of over-time are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site Customer	A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

*** Software Engineering 7th ed, Ian Sommerville***

XP and Agile Principles

- Incremental development is supported through small, frequent system releases.
- Customer involvement means full-time customer engagement with the team.
- People -not process- through pair programming, collective ownership and a process that avoids long working hours.
- Change supported through regular system releases.
- Maintaining simplicity through constant refactoring of code.

*** Software Engineering 7th ed, Ian Sommerville***

Requirements scenarios

- In XP, user requirements are expressed as scenarios or user stories.
- These are written on cards and the development team break them down into implementation tasks. These tasks are the basis of schedule and cost estimates.
- The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates.

*** Software Engineering 7th ed, Ian Sommerville***

Story card for document downloading

Downloading and printing an article

First, you select the article that you want from a displayed list. You then have to tell the system how you will pay for it - this can either be through a subscription, through a company account or by credit card.

After this, you get a copyright form from the system to fill in and, when you have submitted this, the article you want is downloaded onto your computer.

You then choose a printer and a copy of the article is printed. You tell the system if printing has been successful.

If the article is a print-only article, you can't keep the PDF version so it is automatically deleted from your computer.

*** Software Engineering 7th ed, Ian Sommerville***

XP and Change

- Conventional wisdom in software engineering is to design for change. It is worth spending time and effort anticipating changes as this reduces costs later in the life cycle.
- XP, however, maintains that this is not worthwhile as changes cannot be reliably anticipated.
- Rather, it proposes constant code improvement (refactoring) to make changes easier when they have to be implemented.

*** Software Engineering 7th ed, Ian Sommerville***

Testing in XP

- Test-first development.
- Incremental test development from scenarios.
- User involvement in test development and validation.
- Automated test harnesses are used to run all component tests each time that a new release is built.

*** Software Engineering 7th ed, Ian Sommerville***

Task cards for document downloading

Task 1: Implement principal workflow

Task 2: Implement article catalog and selection

Task 3: Implement payment collection

Payment may be made in 3 different ways. The user selects which way they wish to pay. If the user has a library subscription, then they can input the subscriber key which should be checked by the system. Alternatively, they can input an organisational account number. If this is valid, a debit of the cost of the article is posted to this account. Finally, they may input a 16 digit credit card number and expiry date. This should be checked for validity and, if valid a debit is posted to that credit card account.

** Software Engineering 7th ed, Ian Sommerville*

Test case description

Test 4: Test credit card validity

Input:
A string representing the credit card number and two integers representing the month and year when the card expires
Tests:
Check that all bytes in the string are digits
Check that the month lies between 1 and 12 and the year is greater than or equal to the current year.
Using the first 4 digits of the credit card number, check that the card issuer is valid by looking up the card issuer table. Check credit card validity by submitting the card number and expiry date information to the card issuer
Output:
OK or error message indicating that the card is invalid

** Software Engineering 7th ed, Ian Sommerville*

Test-First Development

- Writing tests before code clarifies the requirements to be implemented.
- Tests are written as programs rather than data so that they can be executed automatically. The test includes a check that it has executed correctly.
- All previous and new tests are automatically run when new functionality is added. Thus checking that the new functionality has not introduced errors.

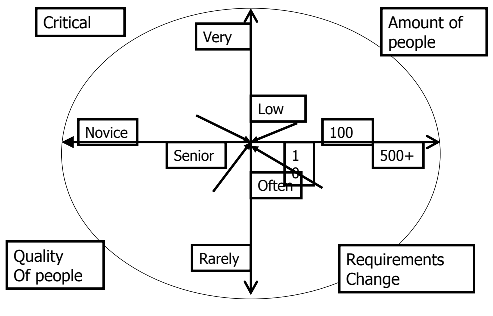
** Software Engineering 7th ed, Ian Sommerville*

Pair Programming

- In XP, programmers work in pairs, sitting together to develop code.
- This helps develop common ownership of code and spreads knowledge across the team.
- It serves as an informal review process as each line of code is looked at by more than 1 person.
- It encourages refactoring as the whole team can benefit from this.
- Measurements suggest that development productivity with pair programming is similar to that of two people working independently.

** Software Engineering 7th ed, Ian Sommerville*

When do we use XP?



FDD - Justin-Josef Angel

Adaptive Software Development

- Focus on human collaboration and team self-organization
- Three phases:
  - Speculation
    - The project initiated and adaptive cycle planning is conducted
    - Define a set of release cycles
  - Collaboration
    - People working together must trust one another
  - Learning
    - Focus group, formal technical reviews, postmortem (introspective)

** SEPA 6th ed, Roger S. Pressman*

Dynamic Systems Development Method

- Provides a framework for building and maintaining systems which meet tight time constraints using incremental prototyping in a controlled environment
- Uses Pareto Principle (80% of project can be delivered in 20% required to deliver the entire project)
- Each increment only delivers *enough* functionality to move to the next increment
- Uses time boxes to fix time and resources to determine how much functionality will be delivered in each increment

** SEPA 6th ed, Roger S. Pressman*

DSDM (2)

- Life cycle activities
  - Feasibility study establishes requirements and constraints)
  - Business study (establishes functional and information requirements needed to provide business value)
  - Functional model iteration (produces set of incremental prototypes to demonstrate functionality to customer)
  - Design and build iteration (revisits prototypes to ensure they provide business value for end users, may occur concurrently with functional model iteration)
  - Implementation (latest iteration placed in operational environment)

** SEPA 6th ed, Roger S. Pressman*

Scrum

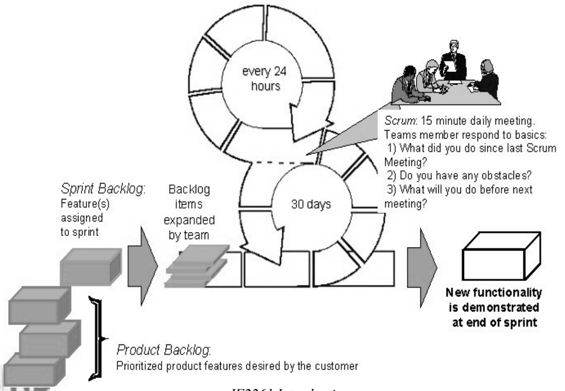
- Scrum principles:
  - Small working teams used to maximize communication, minimize overhead, and maximize sharing of informal knowledge
  - Process must be adaptable to both technical and business challenges to ensure bets product produced
  - Process yields frequent increments that can be inspected, adjusted, tested, documented and built on
  - Development work and people performing it are partitioned into clean, low coupling partitions
  - Testing and documentation is performed as the product is built
  - Provides the ability to declare the product done whenever required

** SEPA 6th ed, Roger S. Pressman*

Scrum (2)

- Process patterns defining development activities:
  - Backlog (prioritized list of requirements or features the provide business value to customer, items can be added at any time)
  - Sprints (work units required to achieve one of the backlog items, must fit into a predefined time-box, affected backlog items frozen)
  - Scrum meetings (15 minute daily meetings) addressing these questions: What was done since last meeting? What obstacles were encountered? What will be done by the next meeting?
  - Demos (deliver software increment to customer for evaluation)

** SEPA 6th ed, Roger S. Pressman*


every 24 hours
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
30 days
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

Crystal

- Development approach that puts a premium on maneuverability during a resource-limited game of invention and communication with the primary goal of delivering useful software and a secondary goal of setting up for the next game
- Crystal principles:
  - Its always cheaper and faster to communicate face-to-face
  - As methodologies become more formal teams become weighed down and have trouble adapting to project work vagaries
  - As projects grow in size, teams become larger and methodologies become heavier
  - As projects grow in criticality some degree of formality will need to be introduced in parts of the methodology
  - As feedback and communication become more efficient the need for intermediate work products is reduced
  - Discipline, skills, and understanding counter process, formality, and documentation
  - Team members not on the critical project path can spend their excess time improving the product or helping people who are on the critical path

** SEPA 6th ed, Roger S. Pressman*

Feature Driven Development

- Practical process model for object-oriented software engineering
- Feature is a client-valued function, can be implemented in two weeks or less
- Definition of features provides the following benefits:
  - User can describe them more easily
  - Can be organized into hierarchical business-related grouping
  - Team develops operational features every two weeks
  - Their design and code are easier to inspect effectively
  - Project planning, scheduling, and tracking are driven by the feature

** SEPA 6th ed, Roger S. Pressman*

FDD (2)

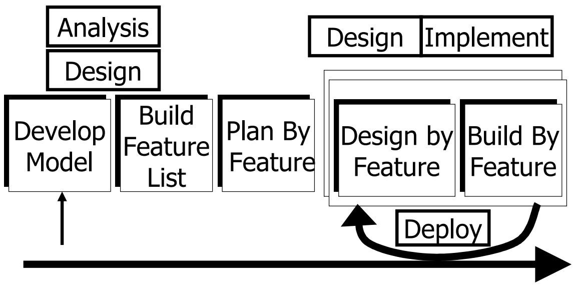
- Feature examples:
  - Add the product to a shopping cart
  - Display the technical-specifications of a product
  - Store the shipping-information for a customer
- Feature set example:
  - Making a product sale

FDD (3)

- Framework activities
  - Develop overall model (contains set of classes depicting business model of application to be built)
  - Build features list (features extracted from domain model, features are categorized and prioritized, work is broken up into two week chunks)
  - Plan by feature (features assessed based on priority, effort, technical issues, schedule dependencies)
  - Design by feature (classes relevant to feature are chosen, class and method prologs are written, preliminary design detail developed, owner assigned to each class, owner responsible for maintaining design document for his or her own work packages)
  - Build by feature (class owner translates design into source code and performs unit testing, integration performed by chief programmer)

** SEPA 6th ed, Roger S. Pressman*

The FDD Process



FDD - Justin-Josef Angel

Agile Modeling

- Practice-based methodology for effective modeling and documentation of software systems in a light-weight manner
- Modeling principles
  - Model with a purpose
  - Use multiple models
  - Travel light (only keep models with long-term value)
  - Content is more important than representation
  - Know the models and tools you use to create them
  - Adapt locally

**** SEPA 6th ed, Roger S. Pressman***

Agile Modeling (2)

- Requirements gathering and analysis modeling
  - Work collaboratively to find out what customer wants to do
  - Once requirements model is built collaborative analysis modeling continues with the customer
- Architectural modeling
  - Derives preliminary architecture from analysis model
  - Architectural model must be realistic for the environment and must be understandable by developers

Problems with Agile Methods

- It can be difficult to keep the interest of customers who are involved in the process.
- Team members may be unsuited to the intense involvement that characterises agile methods.
- Prioritising changes can be difficult where there are multiple stakeholders.
- Maintaining simplicity requires extra work.
- Contracts may be a problem as with other approaches to iterative development.

**** Software Engineering 7th ed, Ian Sommerville***