Process Models

**EEC 521/421: SOFTWARE ENGINEERING**

- General Characteristics of Prescriptive Models
  - Framework activities populated with explicit tasks for software engineering actions
  - The models *prescribe* how the various actions are to be executed
  - The models prescribe a *workflow*
The Waterfall Model

- Sequential approach to software development

- Good for projects that can be planned completely right at the beginning
  - Well-defined requirements
  - No changes anticipated

- Most real projects cannot be built sequentially
- Most new projects have inherent uncertainty
- No way of checking if requirements are met
The Incremental Model

- Iterative application of the Waterfall
- Product is “grown” through several increments

The Incremental Model (2)

- The model is an *evolutionary* approach
  - The product gains more features with every iteration
- Similar to Waterfall
  - Each increment is built sequentially
  - Changes are expensive inside an increment
- Each increment is a *usable, finished* product
  - Not a prototype
Prototyping

- Requirements are very fuzzy
  - Idea is there, but no details
  - Customer is not able to clearly state requirement
- Communication is key

Communication ➔ Quick plan ➔ Modeling and Quick Design ➔ Prototype construction ➔ Deployment, Delivery & Feedback ➔ Communication

Prototyping (2)

- Great way to elicit requirements
  - Can serve as requirements gathering technique for other process models
  - Saves a lot of change later on in the life-cycle
- Customer may not like the “throw-away” nature of the prototype
  - May want to shorten schedule
- Bad development choices may be made
The Spiral Model

- The most mature of evolutionary models
  - Controlled and systematic, yet evolutionary
- Risk-driven process model
  - Cyclic approach
    - Increase degree of definition and implementation
    - Decrease degree of risk
    - Anchor point milestones

The Spiral Model (2)
**The Spiral Model (3)**

- Realistic model for large projects
- Periodic risk evaluation leads to better utilization of resources
- When is the project *finished*?
- Considerable risk assessment expertise is required

---

**The Unified Process**

- Unifies important features of other process models
- Specifically targeted at Object-Oriented system development
- Based on the concept of **use-cases**
- Includes a set of design tools
  - Unified Modeling Language
The Unified Process (2)

- Elaboration
- Construction
- Transition
- Production
- Release
- Software increment
- Communication
- Planning
- Modeling
- Deployment

The Unified Process (3)

- Defined set of work products
  - Use-case model (inception)
  - Non-functional requirements (elaboration)
  - Analysis model (elaboration)
  - Design model (construction)
  - Implementation model (construction)
  - Test cases (construction)
  - Beta test reports (transition)
  - User feedback reports (transition)
Other Prescriptive Models

- Component-Based Development
  - Integrating components with well-defined interfaces to build system

- Formal Methods Model
  - Specification, development, and verification of system using rigorous mathematical notation

AGILE MODELS
Change is Inevitable

- Software engineering is the translation of tacit knowledge into tangible code
- Requirements are always shifting
  - Market moves too fast for the process
  - New product definition takes a very long time
- Process must account for change

Defining Agility

- Effective response to change
- Including the customer in the development process
- *Highest Priority*: Delivery of operational software
  - Deliver increments *frequently*
- Favor *face-to-face communication* over documentation
- *Simplicity* is essential
- Self-organizing teams
  - Team reflects on its behavior and changes accordingly
What is an Agile Process?

- Key assumptions of major software projects
  - Impossible to predict how customer requirements will change
  - Design and construction activities must be performed in tandem
  - Stages of conventional life-cycle are not as predictable as we want
- Agile processes are **incrementally adaptable**
- Incremental development strategy

Key Traits of an Agile Team

- Competence
- Common focus
- Collaboration
- Decision-making ability
- Fuzzy problem-solving ability
- Mutual trust and respect
- Self-organization
Extreme Programming (XP)

- Simple design
- CRC Cards
- User stories
- Values
- Iteration Plan
- Spike solutions
- Prototypes
- Pair programming
- Design
- Refactoring
- Coding
- Unit test
- Continuous integration
- Test
- Acceptance testing
- Release
- Software increment

User stories
Values
Acceptance test criteria
Iteration Plan

User stories
Customer assigns value
Developers assigns cost

Project Velocity
- Computed at the end of the first release
  - Number of stories implemented in first release
  - Estimates for future release
  - Guard against over-commitment

Keep-it-Simple
- Design of extra functionality is discouraged
Extreme Programming (3)

- Spike Solutions
  - Operational prototype intended to clear confusion
- Refactoring
  - Modify internals of code
  - No observable change
- Pair programming
  - “Two heads are better than one”
  - Continuous integration
- Acceptance tests
  - Specified by customer

Adaptive Software Development

- Focus is on human collaboration and team self-organization
  - Synergistic emergent outcome from individuals in team
- Life-cycle
  - Speculation
  - Collaboration
  - Learning
Adaptive Software Development (2)

Adaptive cycle planning
- Mission statement
- Project constraints
- Basic requirements
- Time-boxed Release Plan

Collaboration
- Requirements gathering
  - JAD
  - Mini-specs

Speculation
- Software increment
  - Adjustments for subsequent cycles

Learning
- Components implemented/tested
  - Focus groups for feedback
  - Formal technical reviews
  - Postmortems

Release

Dynamic Systems Development Method (DSDM)

- Modified Pareto principle: “80% of app can be delivered in 20% of the time”
- Life-cycle: 2 studies followed by 3 iterative cycles
  - Feasibility study
  - Business study
  - Functional model iteration
    - Functional prototypes
  - Design and build iteration
  - Implementation
Scrum

Chicken: Let’s start a restaurant!
Pig: What would we call it?
Chicken: Ham n’ Eggs!
Pig: No thanks. I’d be committed, but you’d only be involved!

- Key idea: identify the real stakeholders, and allow them to make decisions
- Key features
  - Backlog
  - Sprints
  - Scrum meetings
  - Demos

The Politics of Agile Methods

- A “raging” debate between supporters and opposers of agile methods
  - Almost of a level of scorn for each other
  - Largely results from lack of understanding of each other
- Traditional methodologists are viewed as “old-school”
  - Agilists don’t believe their processes will allow change
- Agilists are viewed as glorified hackers
  - Process champions don’t believe that agile methods will scale to big projects