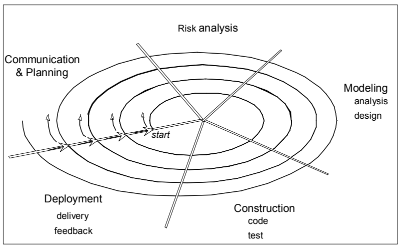
Software Process Models II

Suradet Jitprapaikulsarn

Spiral model



Derived from Roger S. Pressman, Software Engineering: A Practitioner's Approach, 6th Edition, McGraw-Hill, 2005

Spiral Model

The Good

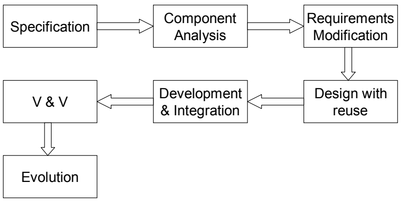
- Risks are explicitly handled
- Realistic approach for the development of large-scale systems and software
- Lower overall risks

The Bad

- Demands and relies on expertise on risk assessment
- Difficult to convince customers that the evolutionary approach is controllable
- The process usually conflicts with the normal procurement model

Component-based software engineering (CBSE)

- Relies on a large base of reusable software components


Specification
Component Analysis
Requirements Modification
Design with reuse
Development & Integration
V & V
Evolution

Component-based software engineering (CBSE)

The Good

- Reduce the amount of software to be developed
- High reusability
- Fast delivery

The Bad

- Compromised requirements
- Performance, scalability, and upgrading could be problematic
- Lost controls of evolution

Computer-Aided Software Engineering (CASE)

- Software used to automate software process activities
- A lot of hypes
- Improves software quality and productivity

Computer-Aided Software Engineering (CASE)

Limitations

- Software engineering is a design activity based on creative thought
- Software engineering is a team activity

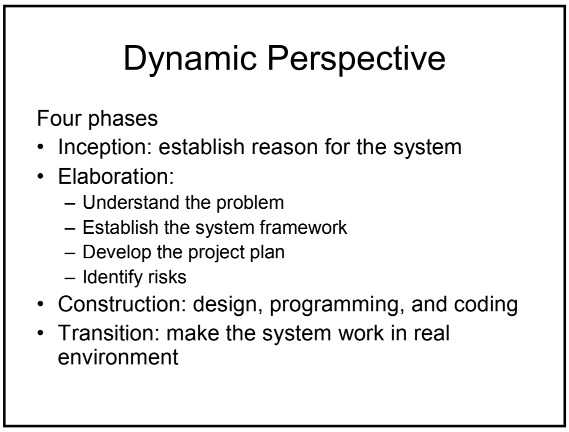
Rational Unified Process (RUP)

- Combining the best features of OO methods
- Developed by the three amigos: Grady Booch, Ivar Jacobson, and James Rumbaugh after they created a unified modeling language (UML)
- A hybrid process model
  - Brings elements from all of the generic process models
  - Supports iterations
  - Illustrates good practice in specification and design

Rational Unified Process (RUP)

Three views in one

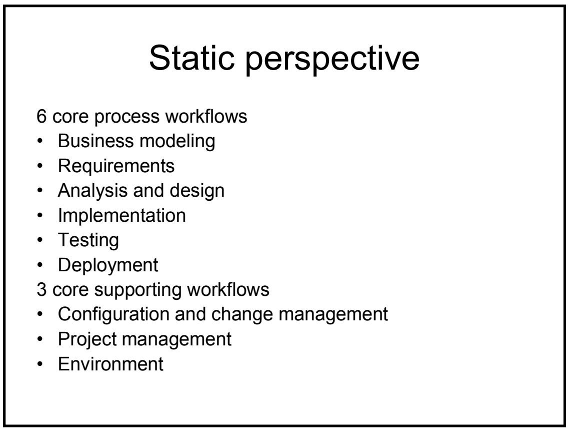
- Dynamic: phases over time
- Static: workflows
- Practice: good practices



Dynamic Perspective

Four phases

- Inception: establish reason for the system
- Elaboration:
  - Understand the problem
  - Establish the system framework
  - Develop the project plan
  - Identify risks
- Construction: design, programming, and coding
- Transition: make the system work in real environment



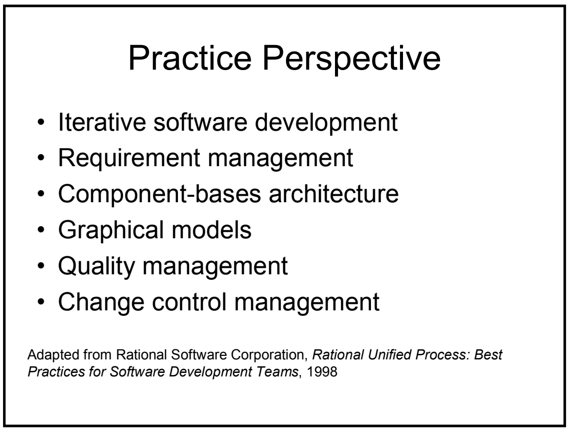
Static perspective

6 core process workflows

- Business modeling
- Requirements
- Analysis and design
- Implementation
- Testing
- Deployment

3 core supporting workflows

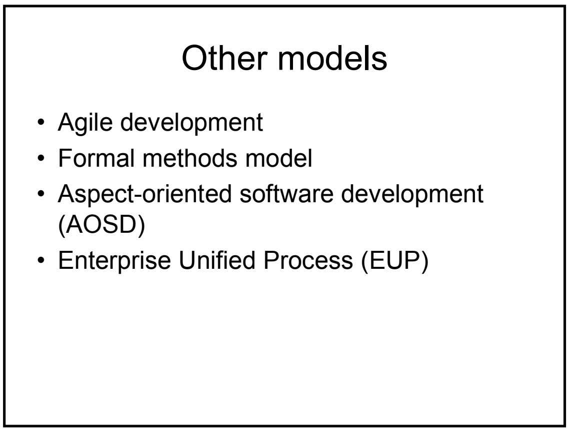
- Configuration and change management
- Project management
- Environment



Practice Perspective

- Iterative software development
- Requirement management
- Component-bases architecture
- Graphical models
- Quality management
- Change control management

Adapted from Rational Software Corporation, *Rational Unified Process: Best Practices for Software Development Teams*, 1998



Other models

- Agile development
- Formal methods model
- Aspect-oriented software development (AOSD)
- Enterprise Unified Process (EUP)