

Prescriptive Models

- Waterfall – sequential order of plan
 - Oldest structured approach to software
 - Most used (historically)
 - Still alive
- Disadvantages
 - No feedback or iteration
 - Must have all requirements up front, unchanging
 - Deployment late, if bugs are found, project fails

Incremental model

- Apply iteration to the waterfall method
- + core product
- + add features on subsequent versions
 - Time consuming
 - Requires massive management
- + low staff
- + makes management more simple

RAD- rapid application development

- Incremental model but with very short cycles
- “high speed” waterfall
- 60 – 90 days (2 – 3 months)
- * must have clear requirements
- * must have constrained project scope
 - Must have resources
 - Need buy-in by customer and team
 - Non-modular system cannot be built with RAD
 - Very high technical risks
 - Not useful for high performance software (requires fine tuning)

Evolutionary Models

- Prototyping (rapid prototyping)
- + fuzzy requirements
- + provides huge flexibility
- + early products
- “always build one to throw away”
- + highly regarded
 - Customer sees a “working” product early and wants it
 - Choose poor OS, programming languages, algorithms can reduce quality

Spiral Model (Boehm)

- Bend the waterfall into a spiral
- Complete a spiral for any product
- Use prototyping
 - Requires buy-in
 - Must have massive risk management
- Very good for sophisticated large projects

Concurrent Development

- Not a sequence
- More of a network
- + very flexible
- - it requires good management
 - “under development”, “awaiting change”
 - “under review”, “under revision”
 - “baselined”, “done”
- Get early product

General Drawbacks (evolutionary models)

- Prototyping has no clear end
 - Plan, manage, assess costs
- Evolution itself is hard to predict
- Focus on speed, flexibility, at the expense of quality

Component Based Development

- Build software from pre-packaged products
 - COTS – commercial off-the-shelf
- Steps
 - Research what is available
 - Analyze integration issues
 - Build software architectures
 - Do the integration (the hardest part?)
 - Comprehensive testing

Aspect Oriented Software Development

- Focus on customer concerns (security, accuracy, fault tolerance, etc)
- “cross-cutting” – the concerns have to be addressed across several components of the software project

Unified Process (UP)

- Focus on the user
 - Use cases
- Inception
- Elaboration
- Construction
- Transition
- Production
- Relies heavily on communication
- Achieved through UML – unified modeling language
- Rational Corporation

Agile Development

- Flexibility, adaptability
 - Change is guaranteed, expected, inevitable
 - Requirements, technology, people, market demands, competition, risks, schedules
- More in chapter 4...