SOFTWARE LIFE-CYCLES

Beyond the Waterfall

Problems with the Waterfall Model

• sequentiality
• late testing paradigm
• late feedback to customer and engineer
• minimal risk management for customer and engineer

Analysis of the V-Shape Life-Cycle

• Improves the testing paradigm
  => Quality Assurance
• Does NOT really improve:
  – sequentiality
  – feedback
  – risk management (during development)
Analysis of Incremental Development

- Assumes independent subsystems!
- Improves (by delivering smaller units):
  - feedback: stepwise
  - testing
- Avoids monolithic product
- Does not really improve:
  - risk management during development
  - sequentiality: subsystems

(Rapid) Prototyping

- Goals:
  - break away from sequentiality
  - speed up feedback
  - minimize risks for
    customer and engineer
  - incomplete but executable
  - cheap and fast

Prototyping

- Definition (A. Davis):
  A prototype is a partial implementation of
  a system, constructed primarily to
  enable customer, user, or developer to
  learn about the problem or its solution.
- Types:
  - evolutionary / throw-away
  - horizontal / vertical

Horizontal Prototyping

Vertical Prototyping

Combinations of Prototypes
Analysis of Pure Prototyping

- Improvements:
  - breaks sequentiality
  - supports fast feedback
  - opportunity for risk management
- Problems:
  - missing organisational structure

==> combine with a life-cycle

The Spiral Model

- Goals:
  - risk management
  - compatible mix between clear structure (life-cycle) & flexible prototyping
  - supports fast feedback & quality assurance

Analysis of the Spiral Model

- very generic ==> framework:
  - specific life-cycle must be derived as a specialisation of that framework
- evolutionary enhancement of other models:
  - can collapse into a waterfall life-cycle
- decisions are risk-oriented:
  - risk-driven instead of document-driven

End of Section 2c

coming up:
methodologies for analysis & design