

# Software Engineering

Session 12

INFM 603

# Software

- Software represents an aspect of reality
  - Input and output represent the state of the world
  - Software describes how the two are related
- Programming languages specify the model
  - Data structures model things
  - Structured programming models actions
  - Object-oriented programming links the two
- A development process organizes the effort

# The Waterfall Model

- Key insight: invest in the design stage
  - An hour of design can save a week of debugging!
- Three key documents
  - Requirements
    - Specifies what the software is supposed to do
  - Specification
    - Specifies the design of the software
  - Test plan
    - Specifies how you will know that it did it

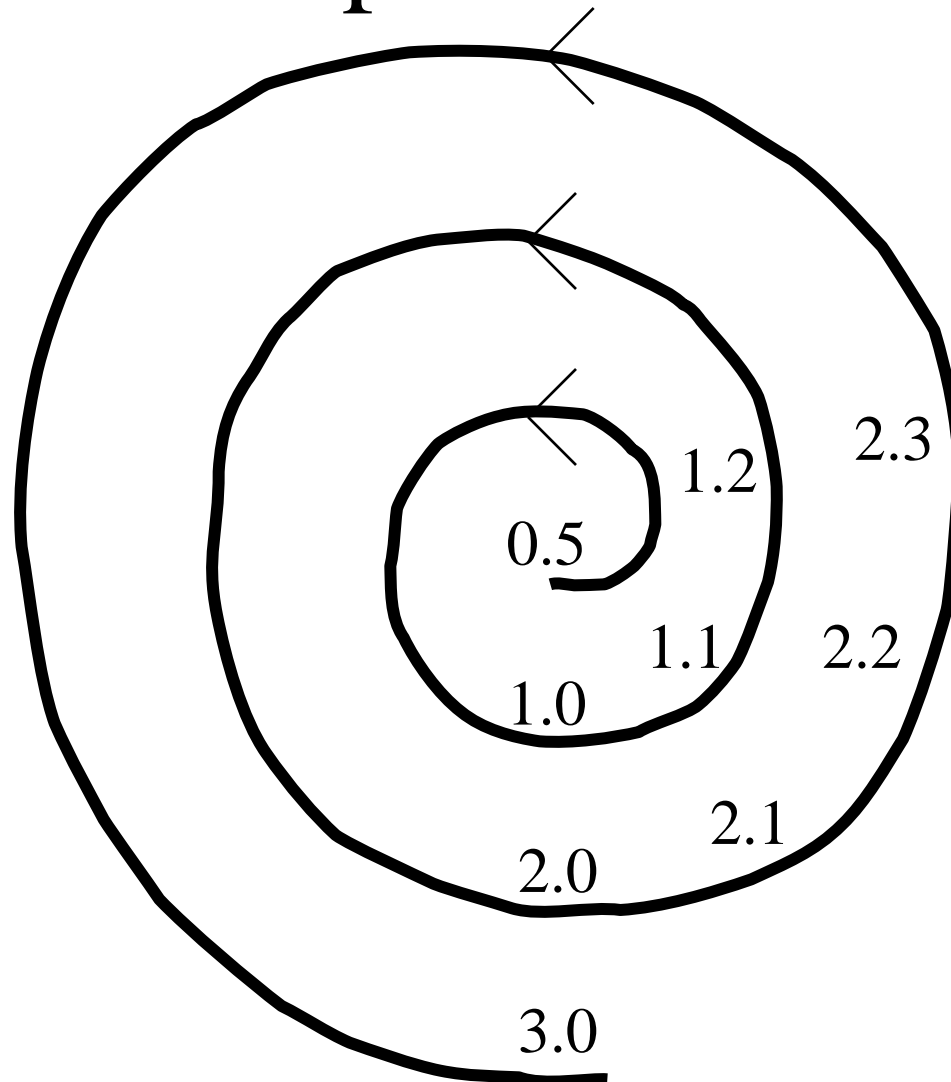
# The Waterfall Model



# The Spiral Model

- Build what you think you need
  - Perhaps using the waterfall model
- Get a few users to help you debug it
  - First an “alpha” release, then a “beta” release
- Release it as a product (version 1.0)
  - Make small changes as needed (1.1, 1.2, ....)
- Save big changes for a major new release
  - Often based on a total redesign (2.0, 3.0, ...)

# The Spiral Model



# Unpleasant Realities

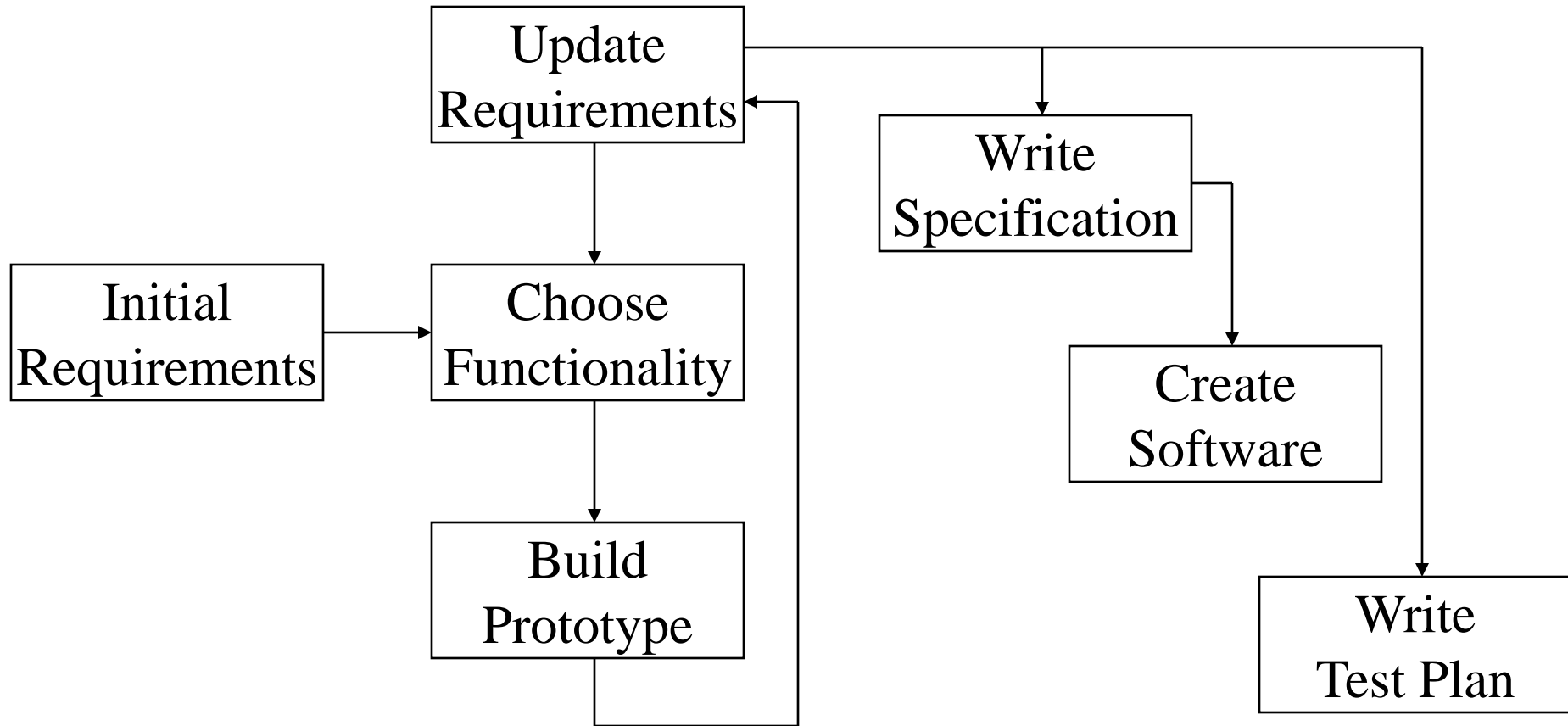
- The waterfall model doesn't work well
  - Requirements usually incomplete or incorrect
- The spiral model is expensive
  - Rule of thumb: 3 iterations to get it right
  - Redesign leads to recoding and retesting

# The Rapid Prototyping Model

- Goal: explore requirements
  - Without building the complete product
- Start with part of the functionality
  - That will (hopefully) yield significant insight
- Build a prototype
  - Focus on core functionality, not in efficiency
- Use the prototype to refine the requirements
- Repeat the process, expanding functionality



# Rapid Prototyping + Waterfall



# Objectives of Rapid Prototyping

- Quality
  - Build systems that satisfy the real requirements by focusing on requirements discovery
- Affordability
  - Minimize development costs by building the right thing the first time
- Schedule
  - Minimize schedule risk by reducing the chance of requirements discovery during coding

# The Specification

- Formal representation of the requirements
- Represent objects and their relationships
  - Using a constrained entity-relationship model
- Specify how the behavior is controlled
  - Activity diagrams, etc.

# Characteristics of Good Prototypes

- Easily built (about a week's work)
  - Requires powerful prototyping tools
  - Intentionally incomplete
- Insightful
  - Basis for gaining experience
  - Well-chosen focus (**DON'T** built it all at once!)
- Easily modified
  - Facilitates incremental exploration

# Prototype Demonstration

- Choose a scenario based on the task
- Develop a one-hour script
  - Focus on newly implemented requirements
- See if it behaves as desired
  - The user's view of correctness
- Solicit suggestions for additional capabilities
  - And capabilities that should be removed

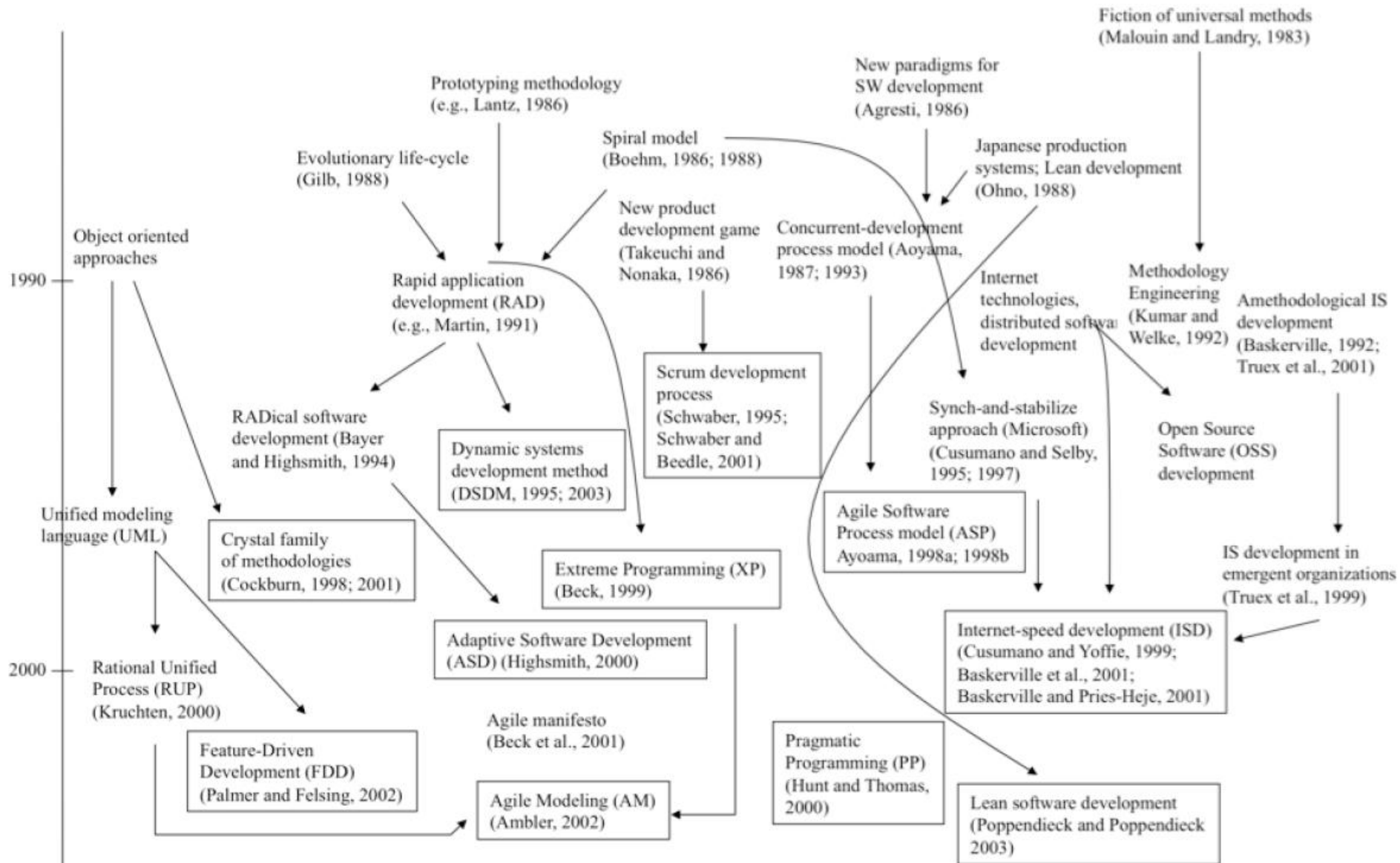
# A Disciplined Process

- Agree on a project plan
  - To establish shared expectations
- Start with a requirements document
  - That specifies only bedrock requirements
- Build a prototype and try it out
  - Informal, focused on users -- not developers
- Document the new requirements
- Repeat, expanding functionality in small steps

# What is NOT Rapid Prototyping?

- Focusing only on appearance
  - Behavior is a key aspect of requirements
- Just building capabilities one at a time
  - User involvement is the reason for prototyping
- Building a bulletproof prototype
  - Which may do the wrong thing very well
- Discovering requirements you can't directly use
  - More efficient to align prototyping with coding

# Agile Methods





# Comparing Agile Methods

