COMP 5700/6700/6706
Software Process

Spring 2016
David Umphress

Common Process Elements

- Lesson: Common Process Elements
- Strategic Outcome: To understand components common to all software processes
- Tactical Outcomes:
  - to understand what a life cycle is
  - know the common types of life cycles
  - know how the common lifecycles affect the software production process
  - be able to select a lifecycle for a sample problem
- Support material:
  - Reading: The Spiral Model as a Tool for Evolutionary Acquisition
- Instant take-aways:
  - life cycles

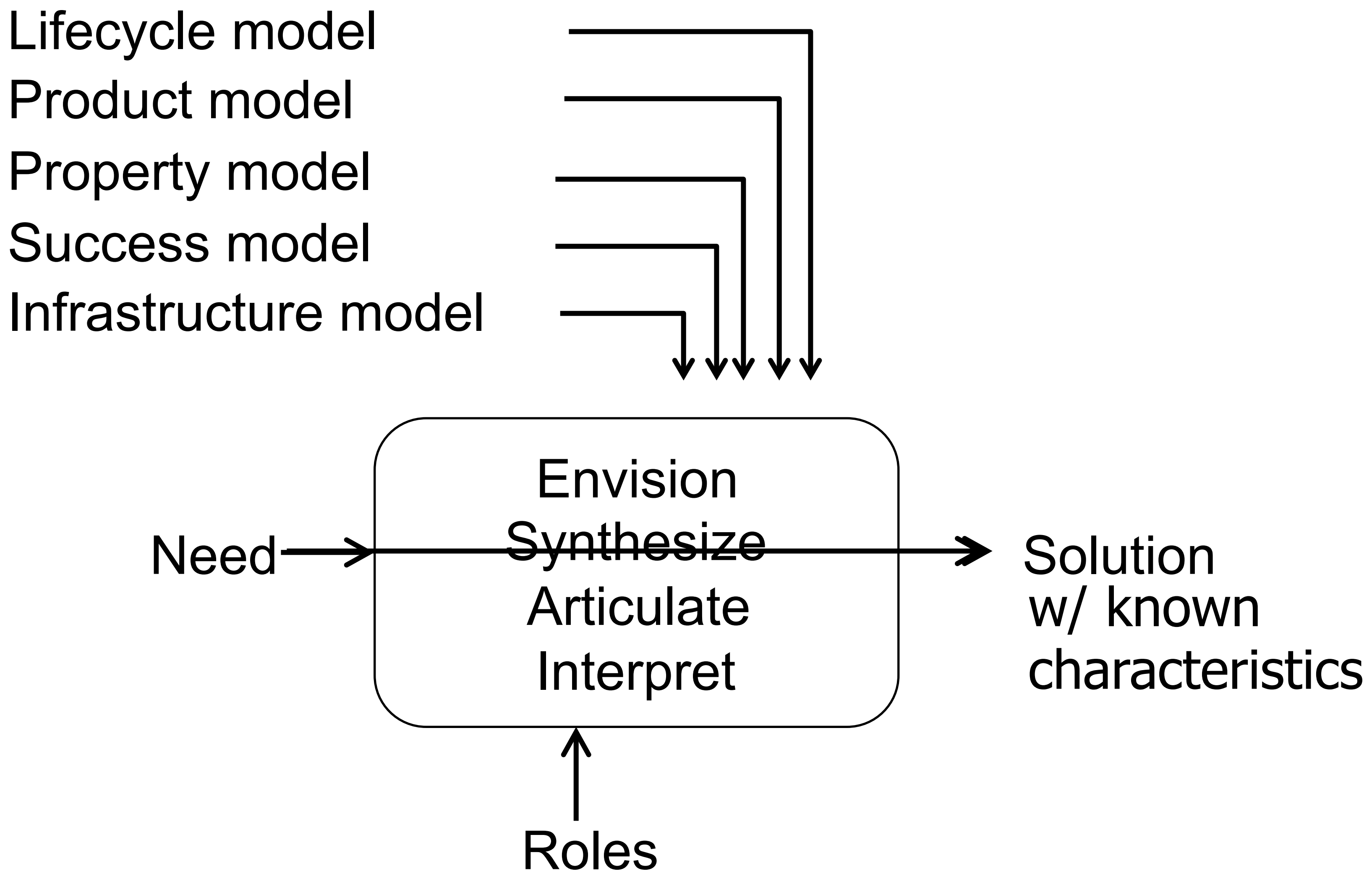
- Bookshelf items:
  - Boehm, B. 1989. A spiral model of software development and enhancement. *Computer*, *21*, 5. pp 61-72.
  - Royce, W. 1970. Managing the development of large software systems. *Proceedings of IEEE WESCON*. IEEE.

Syllabus

- Software engineering raison d’être
- Process foundations
- Common process elements
- Analysis
- Construction
- Reviews
- Refactoring
- Integration
- Repatterning
- Architecture
- Estimation
- Scheduling
- Measurements
- Process redux
- Process descriptions*
- Infrastructure*
- Retrospective

- **Process commonality**
- **Lifecycle models**
  - **foundation**
    - **engineering activities**
    - **lifecycle definition**
  - **models**
    - **ad hoc**
    - **linear**
    - **prototype**
    - **iterative**
    - **transform**

Process Commonalities



Lifecycle model

Product model

Property model

Success model

Infrastructure model

Need

Envision

Synthesize

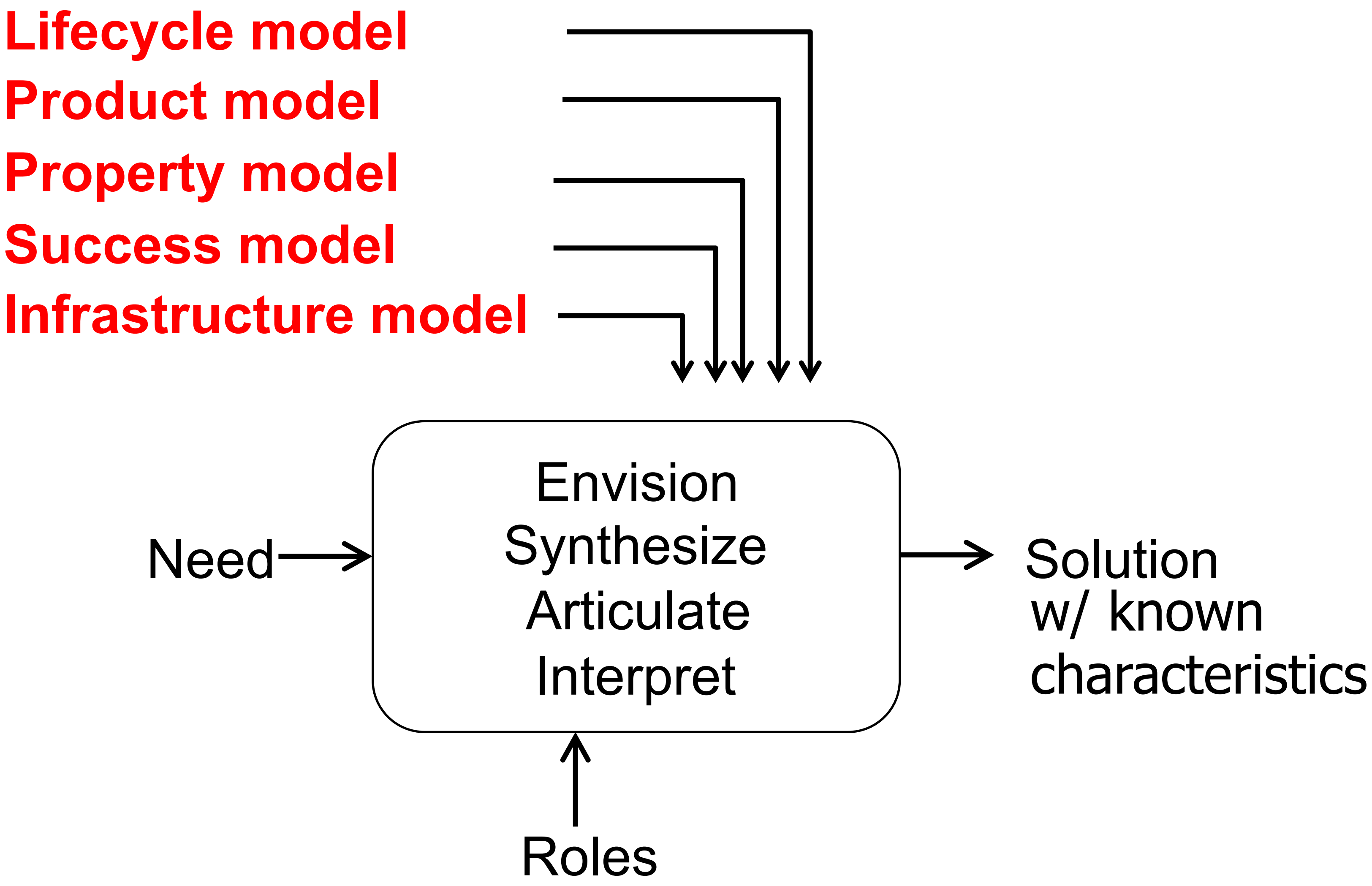
Articulate

Interpret

Solution w/ known characteristics

Roles

Process Commonalities



Lifecycle model

Product model

Property model

Success model

Infrastructure model

Need

Envision
Synthesize
Articulate
Interpret

Solution
w/ known
characteristics

Roles

Process Commonalities

- Lifecycle model
  - tasks needed to produce a solution
  - purpose: to identify how the pieces of the development puzzle fit together
  - more about this in a minute …

Process Commonalities

- Product model
  - mechanisms to produce a deliverable in a specific form
  - purpose: to ensure that the product contains what it should (and doesn’t contain what it shouldn’t)
  - examples
    - configuration management
      - configuration identification
      - configuration control
      - configuration status accounting
      - configuration auditing
    - system/software architecture
    - language-specific logistics

Process Commonalities

- Property model
  - mechanisms to guide production of the solution
  - purpose: to ensure that the product is “engineered”
  - examples
    - schedule
    - cost
    - methodology employment

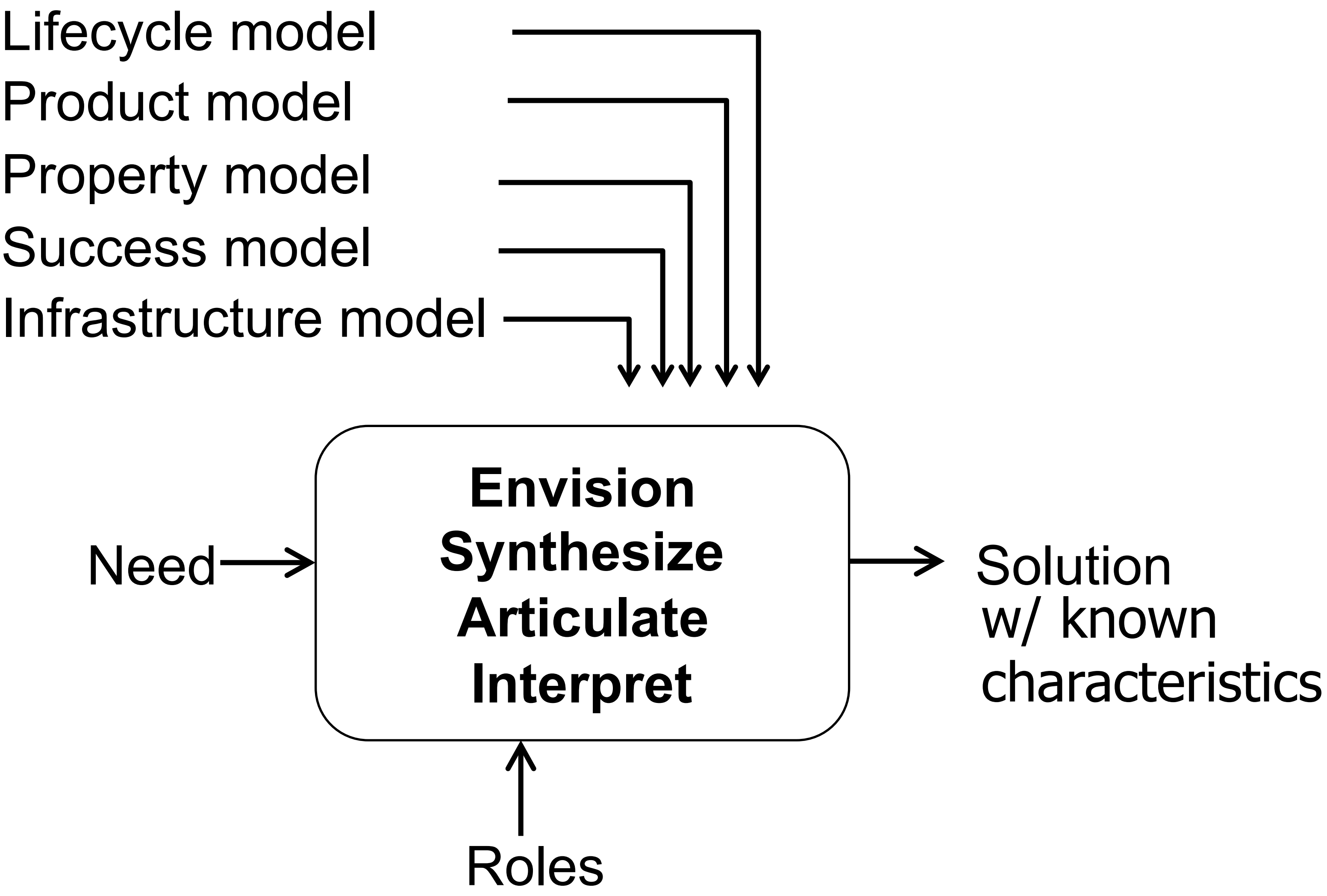
Process Commonalities

- Success model
  - mechanisms to assure and assess “correctness”
    - includes both functional and non-functional needs
  - purpose: V&V
    - verification: “are we building the product right”
    - validation: “are we building the right product”
  - examples
    - proof of correctness
    - formal review
    - walkthrough
    - audit
    - certification
    - testing (unit, integration, system, etc)
    - process integrity

Process Commonalities

- Infrastructure model
  - mechanisms needed to carry out a project
  - purpose: to permit all engineers to engineer
  - examples
    - tools
    - training
    - administration
    - quality assurance
    - process

Process Commonalities



Lifecycle Models

- software engineering activities to be managed:
  - envision
  - synthesize
  - articulate
  - interpret
- characteristics:
  - pros
    - organized
    - management visibility
  - cons
    - no unified theory
    - rigidifies thinking (changes difficult to handle)
    - built-in assumptions

Point of Discussion

- ad hoc
  - code-and-fix
- linear
  - waterfall
- prototype
  - prototype
- iterative
  - incremental
  - spiral
- transform
  - transform

Ad hoc

- Concept: quickly get code on the road
- Example: Code-and-fix
- Pattern
  - repeat
    - code
    - fix
  - until working

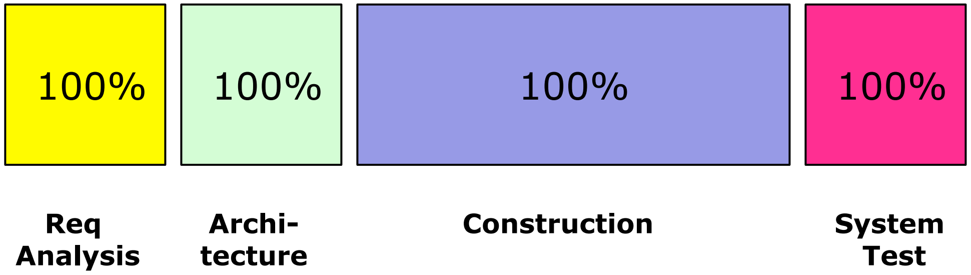
Can you say “Level 1”?

Ad hoc … code-and-Fix

- Characteristics
  - – advantages
    - no overhead
    - requires little expertise
  - – disadvantages
    - no management visibility
    - errors seldom detected in advance
- Used for
  - – small throwaway programs
  - – fluid systems (where may not be cost effective to formally engineer or document)

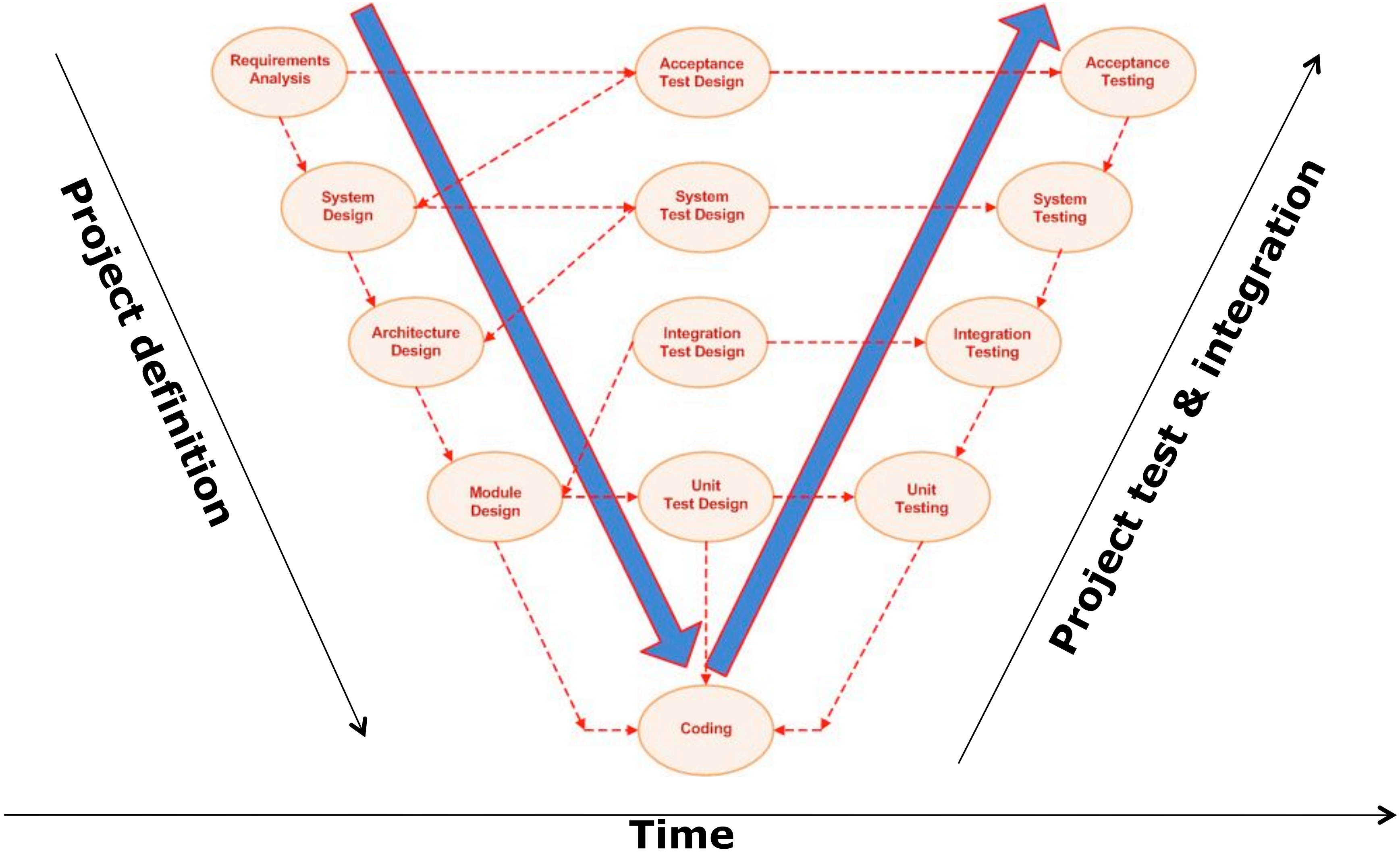
Linear

- Concept: step-by-step construction
- Pattern
  - analyze then
  - design then
  - code then
  - test then
  - maintain

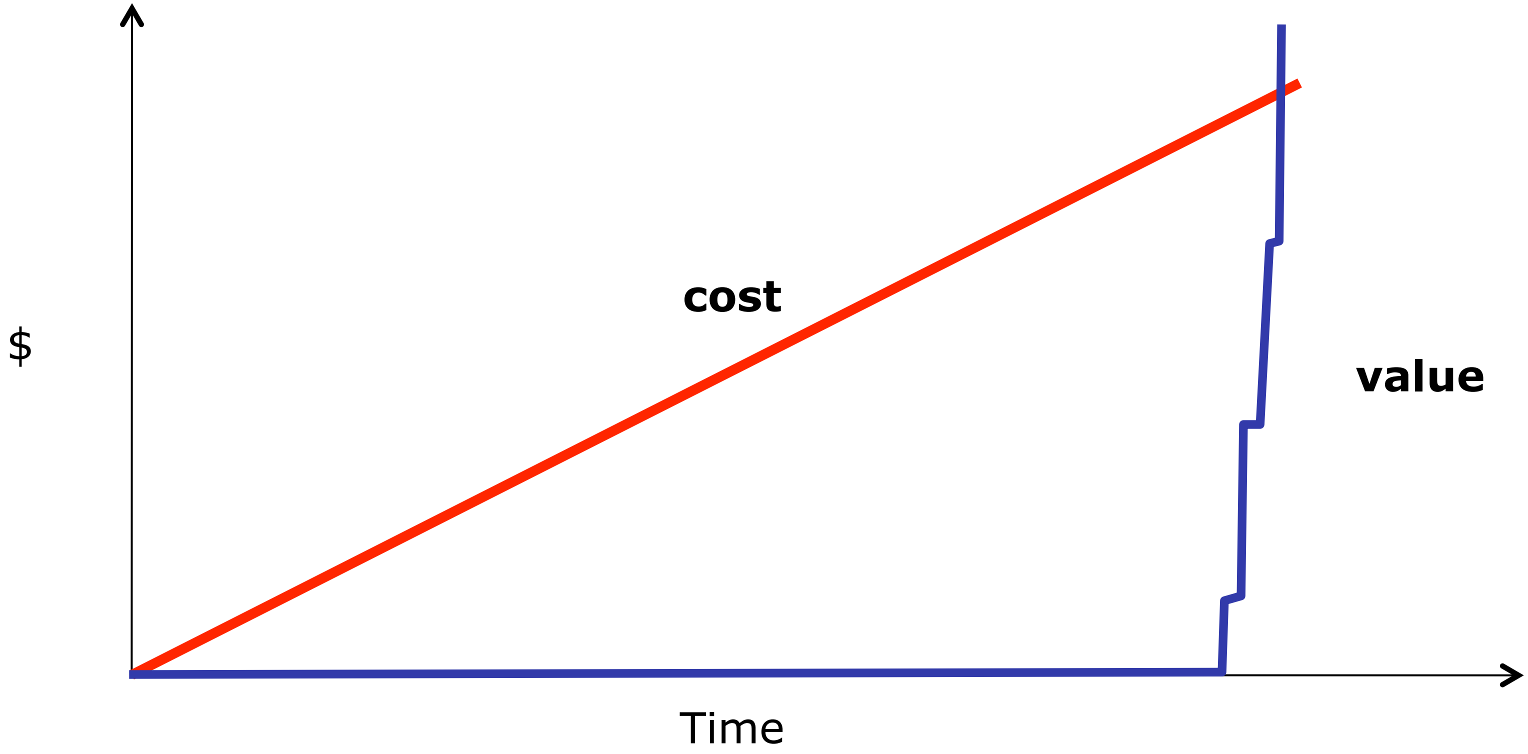


- Example: Waterfall
  - History
    - Royce 1970
      - what he said: phased development
      - what we heard: phase after phase development
    - used widely in procurement

V-Model ... a revised waterfall



Waterfall

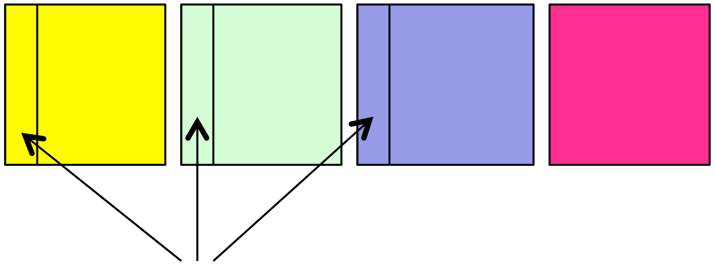


Linear … Waterfall

- Characteristics
  - pros
    - simple
  - cons
    - specification vs aspiration
    - unrepresentative of real life
    - assumes specs frozen
    - guidelines for swimming upstream vague

Prototype

- Concept: development process defines requirements
- Pattern
  - analyze
  - repeat
    - design (rapidly)
    - code (rapidly)
    - test (with user)
  - until system adequate
  - complete* and release



Possible Prototype Points

Note: not restricted to waterfall

*** Some sources say this is optional?**

- Example: Prototype
  - Assumption
    - user can not specify all requirements
    - development process will reveal requirements

Prototype

- Used for
  - HCI
  - research
  - software structure ill-defined
  - requirements ill-defined

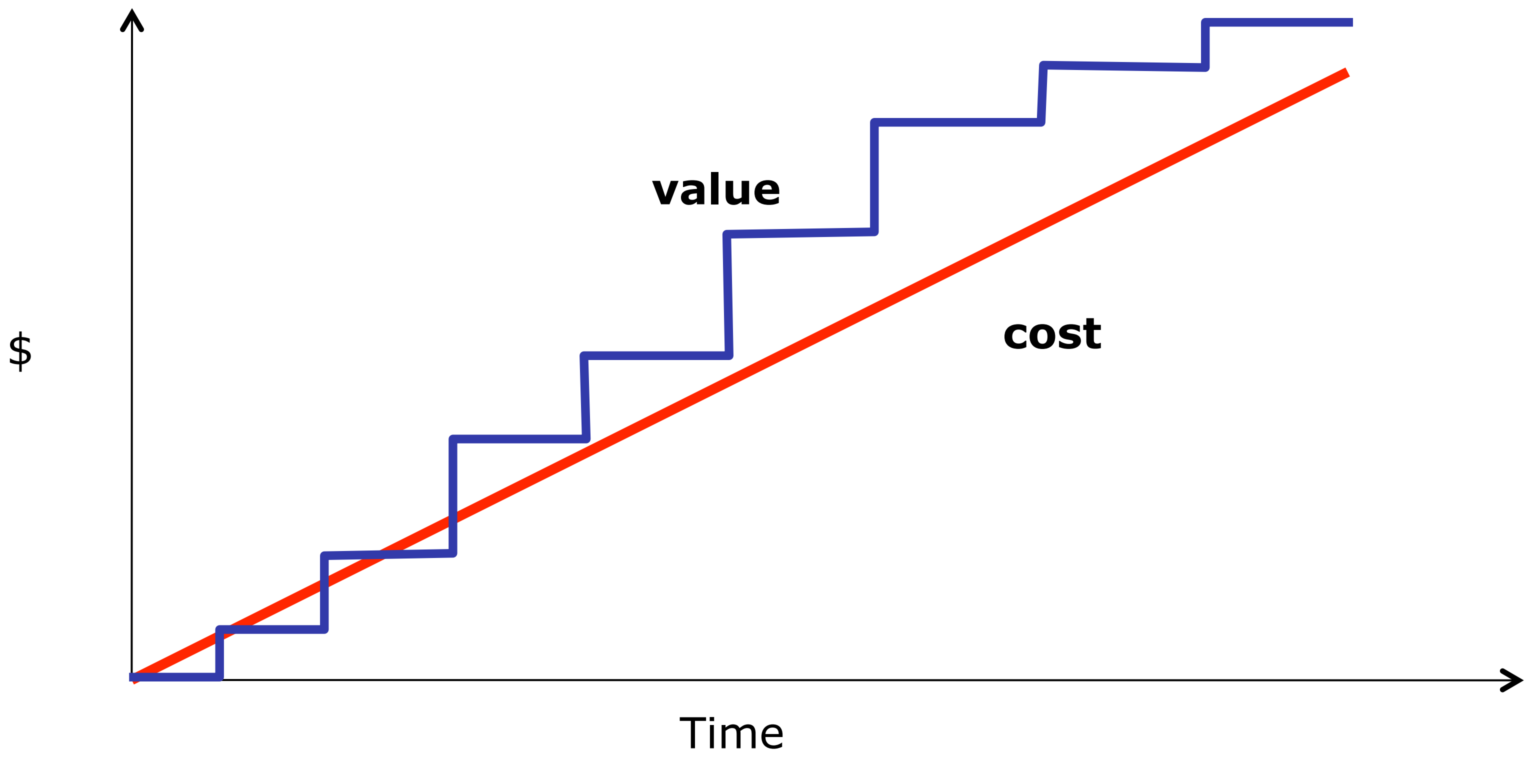
Prototype

- Characteristics
  - Pros
    - gaps in client-developer understanding discovered
    - difficult-to-understand-services refined
    - specs for quality system
    - limited working system at each iteration
  - Cons
    - early iteration may be used as real system
    - users may not use in same way as real system
    - may lead to unrealistic delivery expectations
    - questionable legal contractual vehicle
    - validation demonstrates adequacy, not conformance to spec

Iterative

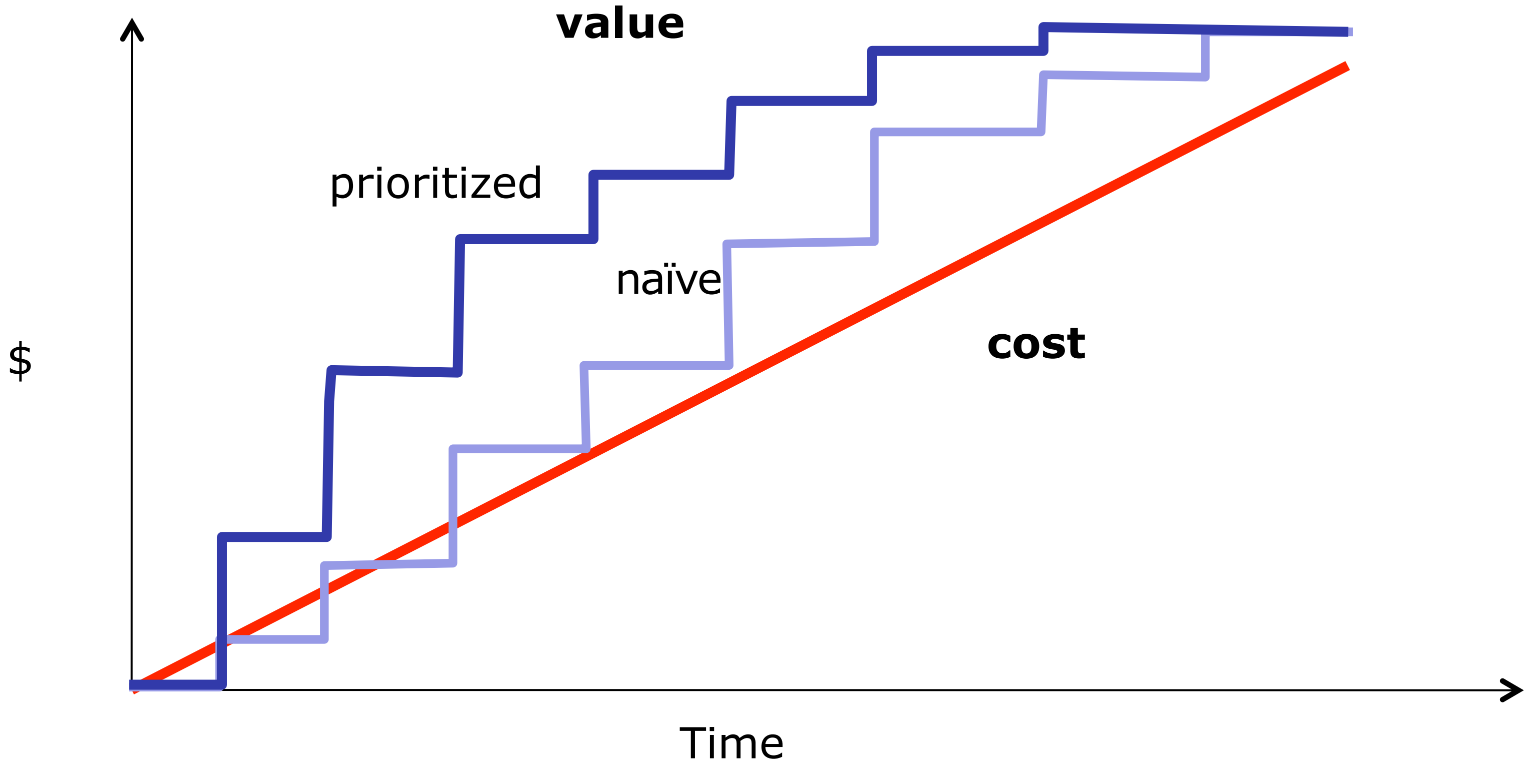
- Concept: build solution in parts
- General pattern
  - repeat
    - analyze (to desired level of abstraction)
    - design (to desired level)
    - code (to desired level)
    - test
    - deliver
  - until complete

Iterative



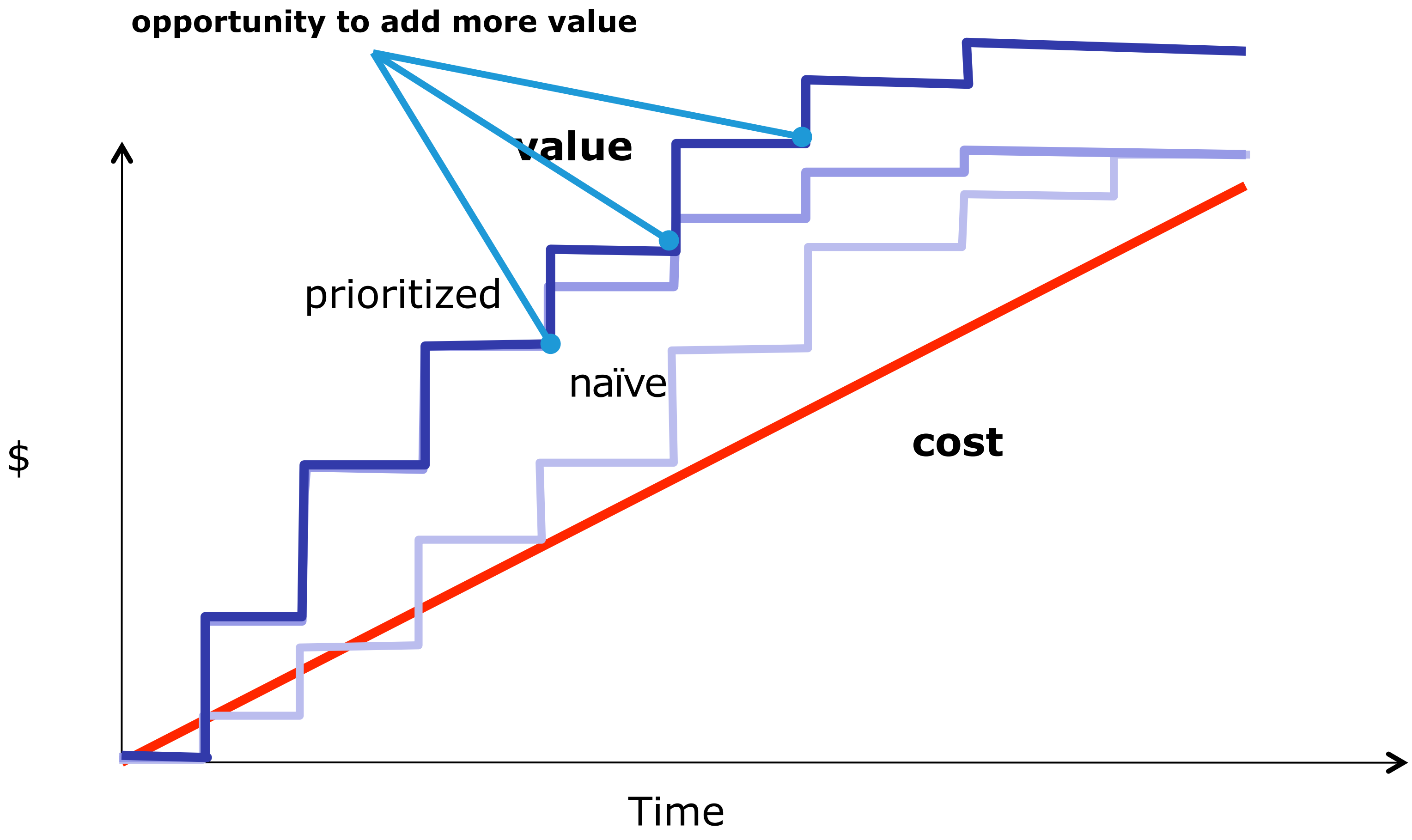
Value delivery … naïve iterations

Iterative



Value delivery … prioritized iterations

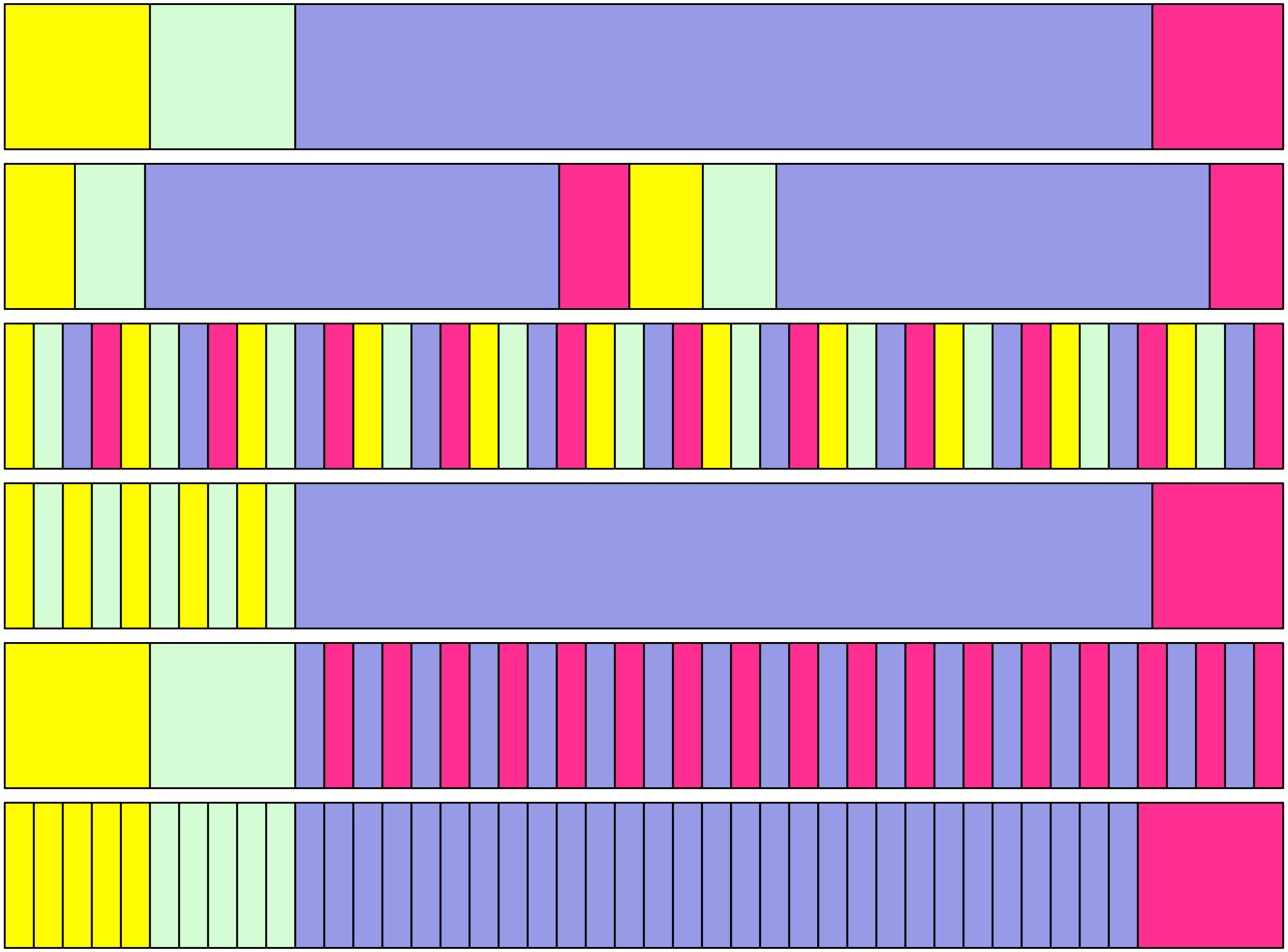
Iterative



Value delivery … prioritized iterations w/ feedback

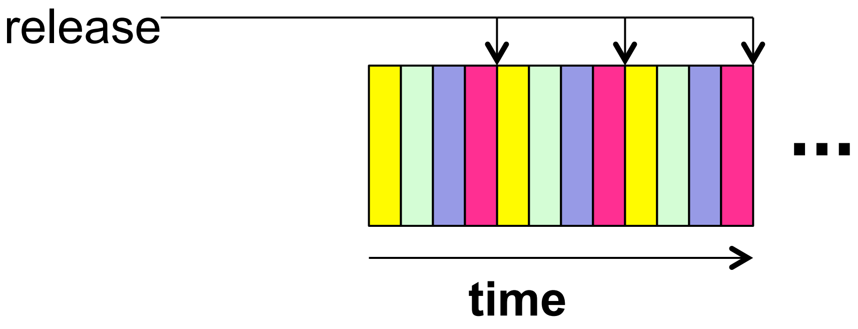
Iterative

variations on a theme

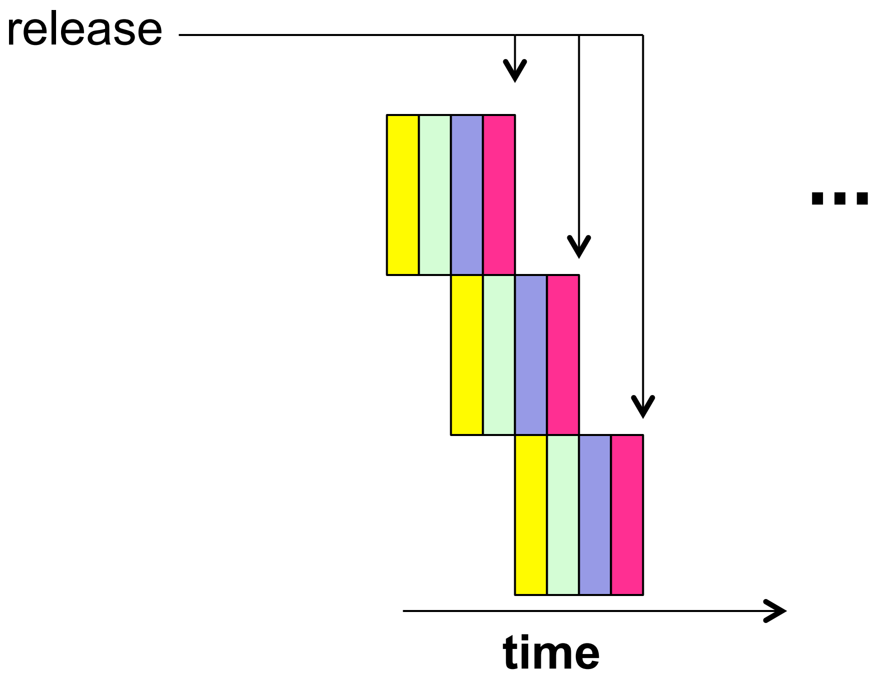


Iterative

- Major theme 1: Incremental waterfall
  - staged waterfall

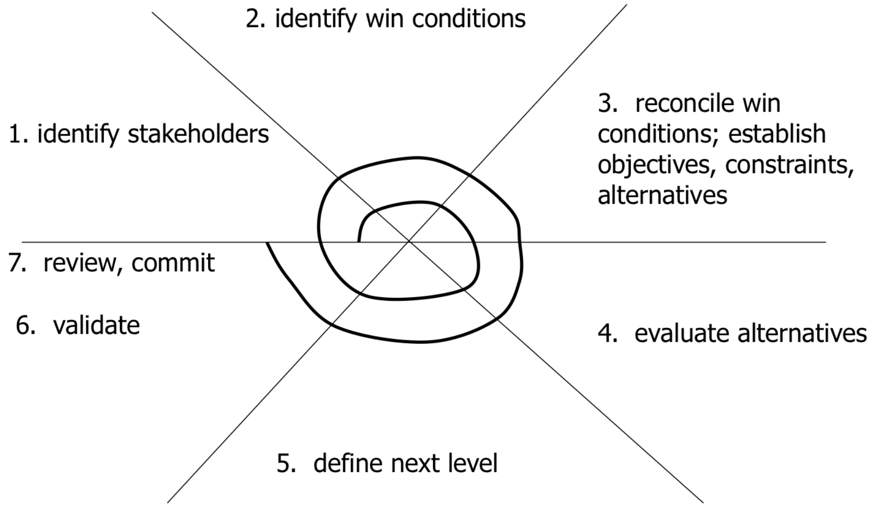


  - overlapping waterfall



Iterative

- Major theme 2: spiral
  - repeat
    - identify next-level stakeholders
    - identify stakeholders' win conditions
    - reconcile win conditions; establish objectives, constraints, alternatives
    - evaluate product and process alternatives; resolve risks
    - define next level of product and process
    - validate product and process
  - until (review decision = stop)
    or (project = complete)


2. identify win conditions
1. identify stakeholders
3. reconcile win conditions; establish objectives, constraints, alternatives
7. review, commit
6. validate
4. evaluate alternatives
5. define next level

Iterative … spiral (historical version)

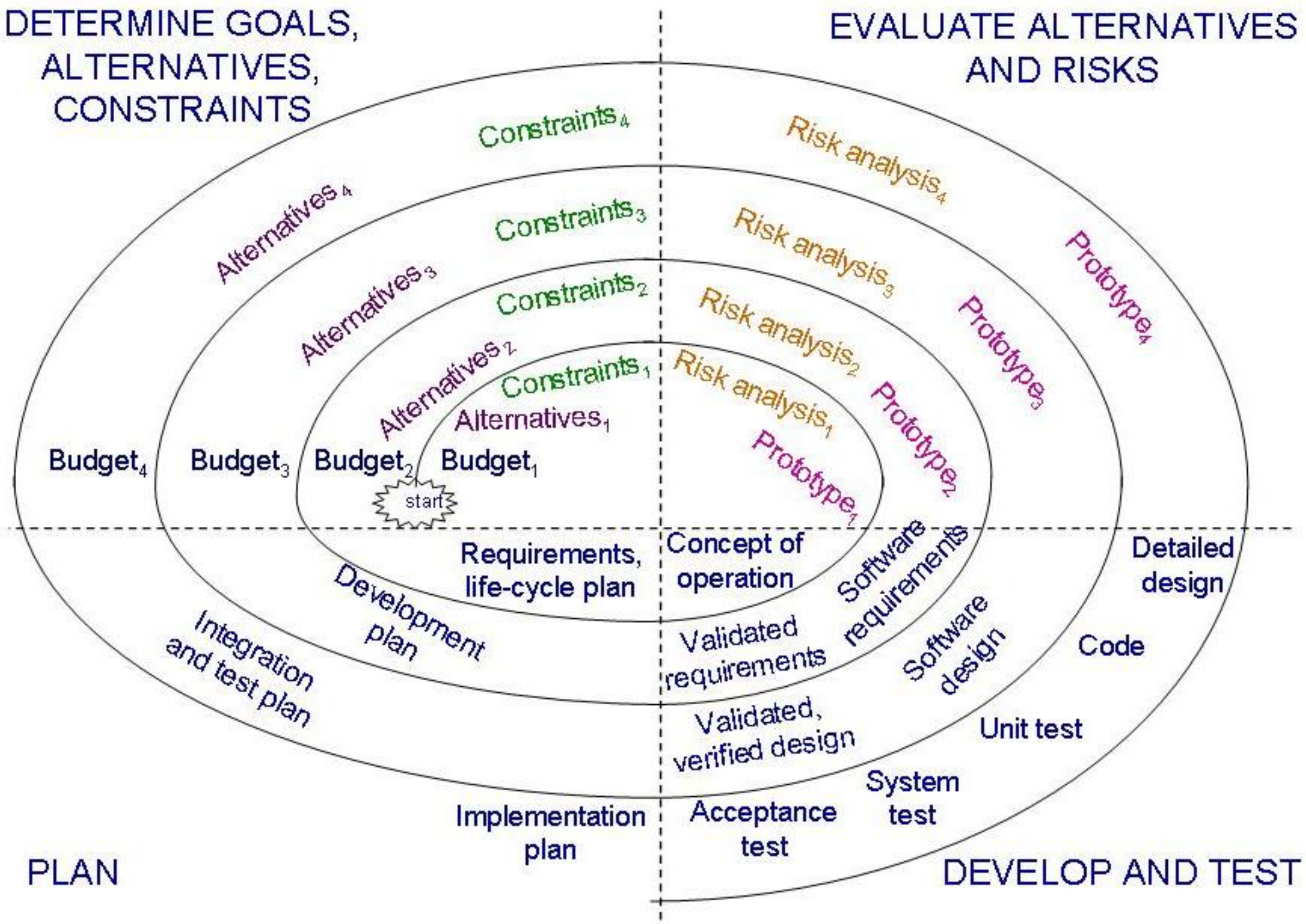


Figure 2.10 the spiral model.

Iterative … spiral

- concurrent determination of key artifacts
- each cycle includes objectives, constraints, alternatives, risks, review, and commitment to proceed
- level of effort driven by risk considerations
- degree of detail driven by risk considerations
- use of anchor point milestones:
  - life cycle objectives
  - life cycle architecture
  - initial operational capability
- emphasis on system and life-cycle activities and artifacts

Iterative … spiral

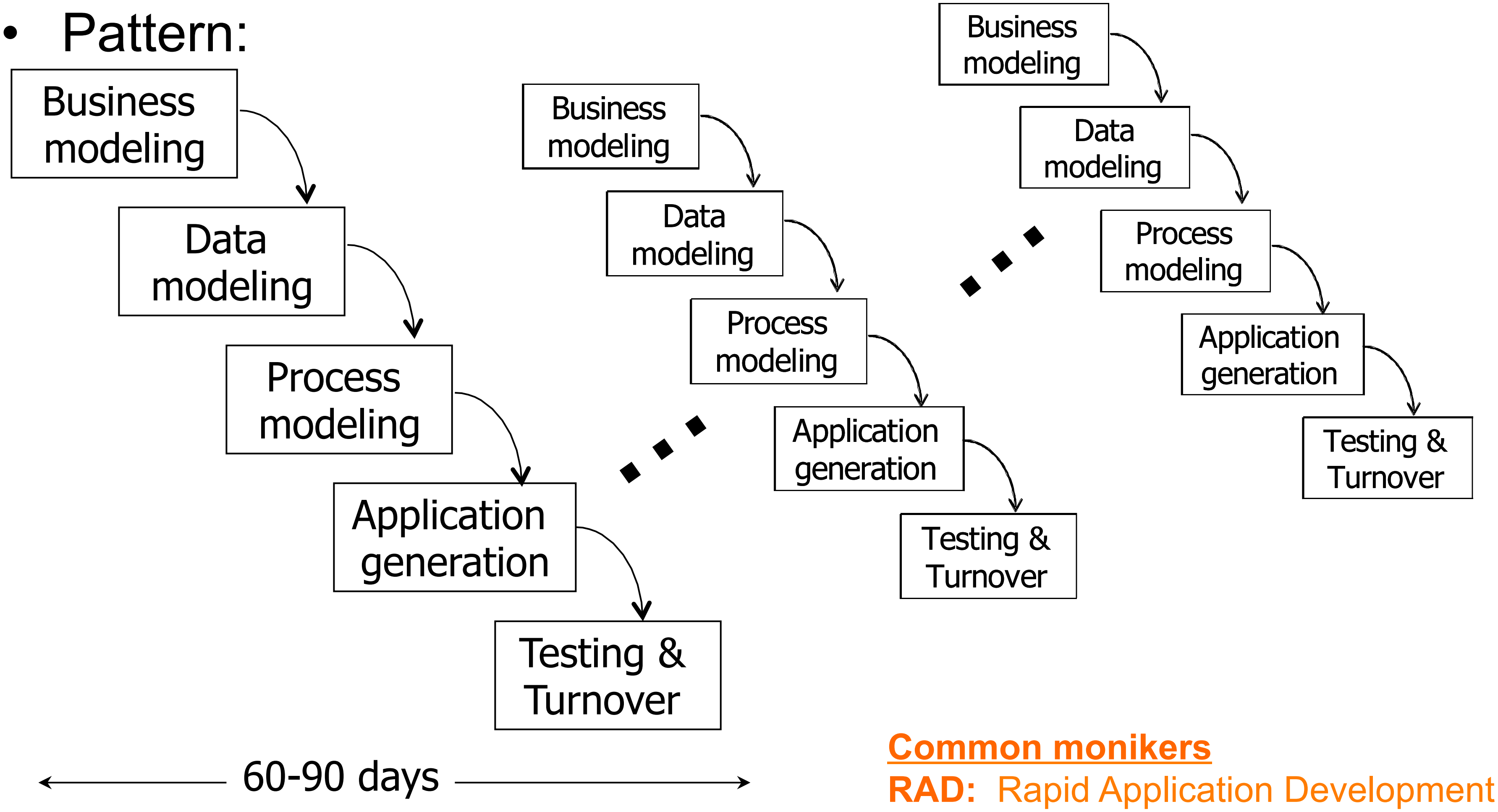
- Characteristics
  - Pros
    - flexibility
    - “systems” approach
    - provides termination condition
    - objectives vs requirements
    - answers “what should we do next”
    - addresses risk
  - Cons
    - contract acquisition nightmare
    - relies on project team educated in risk management

Transform

- Concept: automation
- General pattern
  - analyze
  - repeat
    - specification
    - translate to code
    - validate
  - until system adequate

Transform … RAD

- Example: Rapid Application Development
- Pattern:



Common monikers

RAD: Rapid Application Development

MDD: Model-driven Development

Transform

- Characteristics
  - Pros
    - modify upstream artifact (such as specs or models), not code
    - no design time
  - Cons
    - tracking costs difficult
    - cost estimation difficult
    - auto transform not always available

Summary

Topics

- **Process commonality**
- **Lifecycle models**
  - **foundation**
    - **engineering activities**
    - **lifecycle definition**
  - **models**
    - **ad hoc**
    - **linear**
    - **prototype**
    - **iterative**
    - **transform**

Key Points

- A product with integrity requires harmonized process elements
- Software engineers are faced with projects of multiple levels of complexity: lifecycle modeling is a way of dealing with complexity
- Lifecycles determine sequence of activities
- Common lifecycle patterns are ad hoc, linear, prototype, iterative, transform

Supporting slides

Linear … Waterfall

- Success criteria

Planning	Organizing	Staffing	Directing	Controlling	Misc
• upfront knowledge certain » req. » methods » arch » resource avail	• stable	• experienced	• traditional hierarchy	• guidelines on: » next phase » backup » stopping » req creep » stds	• technology: stable • ethics: audits

Prototype

- Success criteria

Planning	Organizing	Staffing	Directing	Controlling	Misc
• flexibility in scope, schedule, cost • User involvement	• flexible	• user-friendly • disciplined	• non-traditional hierarchy	• stringent cost monitoring • stopping criteria available • requirements creep containment • strong configuration management	• technology: architecture stable • ethics: user willing to assume risk

Iterative

- Success criteria

Planning	Organizing	Staffing	Directing	Controlling	Misc
• flexibility in scope, schedule, cost • User involvement	• flexible	• disciplined • dedicated	• non-traditional hierarchy	• stringent cost monitoring • stopping criteria available • strong configuration management	• technology: architecture unstable • ethics: early delivery

Iterative … spiral

- Success criteria

Planning	Organizing	Staffing	Directing	Controlling	Misc
• previous experience • flexibility in schedule and cost	• stable for duration of spiral sweep	• flexibility to vary task nature between spiral sweeps • risk conscious	• non-traditional or traditional	• guidelines on number of spirals	• technology: need not be stable

Transform

- Success criteria

Planning	Organizing	Staffing	Directing	Controlling	Misc
• Problem domain well defined • Previous experience in costing, scheduling	• stable	• theoretical • experienced	• non-traditional or traditional	• risk abatement if transform not possible	• technology: must be available

Iterative … synchronize and stabilize

- Example 2: Synchronize and stabilize
- Pattern
  - analyze
  - specify as “feature sets”, sort sets by criticality
  - for the each feature set loop
    - diurnal loop
      - build feature set
      - synchronize
        - integrate
        - fix
      - exit when feature set built to satisfaction
    - end loop
    - stabilize (baseline)
  - end loop

Iterative … synchronize and stabilize

- Characteristics
  - Pros
    - repeated synchronization yields piece-wise integration
    - early insight into inter-component interaction
    - working software early
  - Cons
    - working software early
    - delicate team dynamics