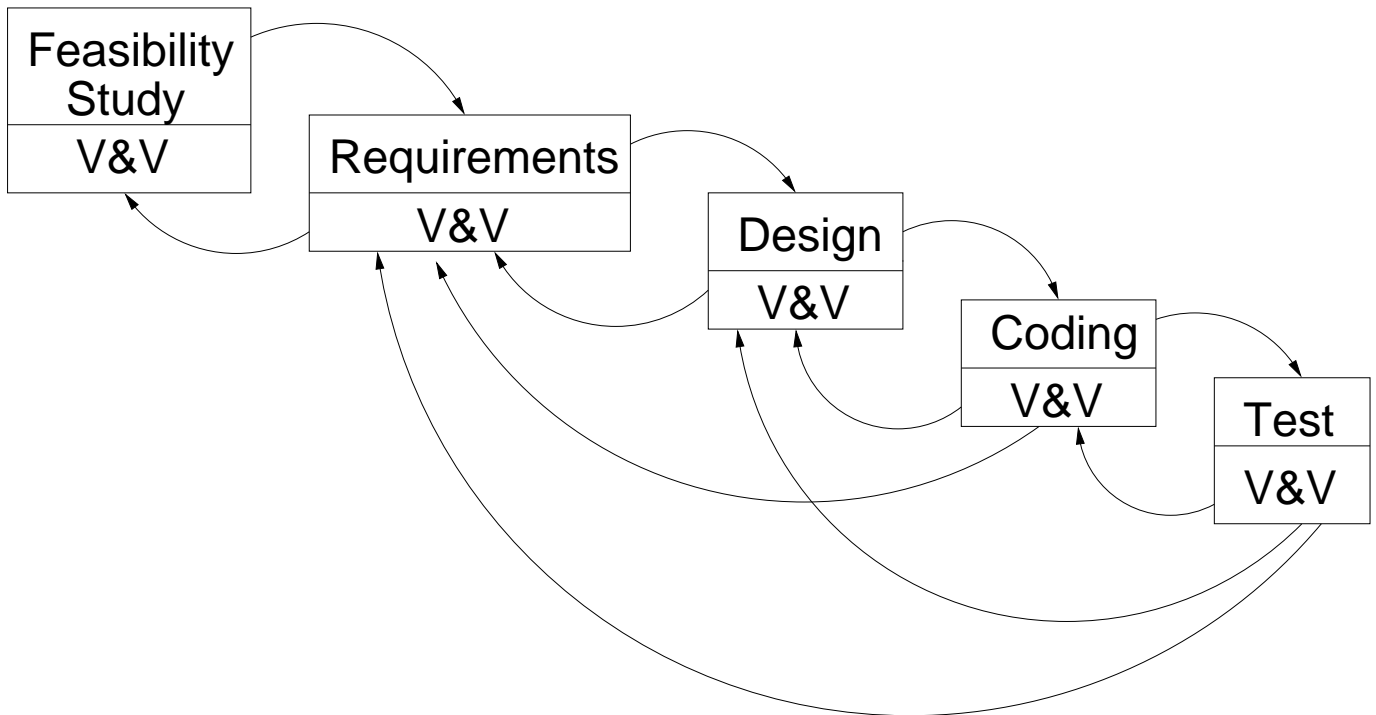


Waterfall Model



- Deliverables – baselines
- Document–driven process
- "Big Bang" testing, "stubs", daily build and smoke test
- "A Rational Design Process and How to Fake It"

Evolutionary Model

- Prototyping – "Do it twice"
 - to assess feasibility
 - to verify requirements
- May only be a front end or executable specification
Or develop system with less functionality or quality attributes
- 3 approaches:
 - 1) Use prototyping as tool for requirements analysis.
Need proper tools
 - 2) Use to accommodate design uncertainty.
Prototype evolves into final product
Documentation may be sacrificed
May be less robust
Quality defects may cause problems later
 - 3) Use to experiment with different proposed solutions
before large investments made.

Evolutionary Models (2)

- Drawbacks:
 - Can be expensive to build
 - Can develop a life of its own – turns out to be product itself
 - Hard to change basic decisions made early
 - Can be an excuse for poor programming practices
- Experimental Evaluation:
 - Boehm: prototyping vs. waterfall
 - Waterfall: addressed product and process control risks better
 - Resulted in more robust product, easier to maintain
 - Fewer problems in debugging and integration due to more thought-out design
 - Prototyping: addressed user interfaces better
 - Alavi: prototyping vs. waterfall applied to an information system
 - Prototyping: users more positive and more involved
 - Waterfall: more robust and efficient data structures

Incremental Model

- Functionality produced and delivered in small increments.
- Focus attention first on essential features and add functionality only if and when needed
- Systems tend to be leaner -- fights overfunctionality syndrome
- May be hard to add features later
- Variant: Incremental implementation only
 - Follow waterfall down to implementation
 - During requirements analysis and system design
 - Define useful subsets that can be delivered
 - Define interfaces that allow adding later smoothly
 - Different parts implemented, tested, and delivered according to different priorities and at different times.

Spiral Model

- Includes every other model
 - Risk driven (vs. document driven or increment driven)
 - Radius of spiral represents cost accumulated so far
-

Do you need one uniform process over entire project?

- In requirements analysis, identify aspects that are uncertain
e.g., library:

checkout and checkin (inventory control) – relatively certain
card catalogue, user search – relatively uncertain

then have separate processes for the different parts.

Software Factory

- Most software organizations strictly separated between initial development and later maintenance.
 - No incentive to produce a system that can be easily maintained.
 - No incentive to produce reusable components.
- Project management vs. product management
- Extend management responsibility to cover family of products rather than an individual product (product families)

Despite the rhetoric, CMM emphasizes control over flexibility and learning

- Control orientation seeks to maintain predictable operations, minimize variation, and avoid surprises.
- Learning orientation seeks to increase variation in order to explore opportunities.
- Formal bureaucratic control undermines intrinsic motivation needed for creative and flexible responses to uncertainty.
- Senge: humanistic values of caring and individual freedom are essential to building learning organizations.
- Carroll: "In too many TQM programs, it is the difficult-to-implement portions of the program that are being finessed or ignored and the rhetoric that is being retained."

Other CMM Problems

- Treats people as assembly line workers, i.e., replaceable, unreliable
Humans are subordinated to defined processes
- Why five levels? Why a rigid order?
- Creates inflexible organizations and the illusion of control
- Places the focus on the wrong things