WaterSluice
A Software Engineering Methodology

Ron Burback

February, 1998

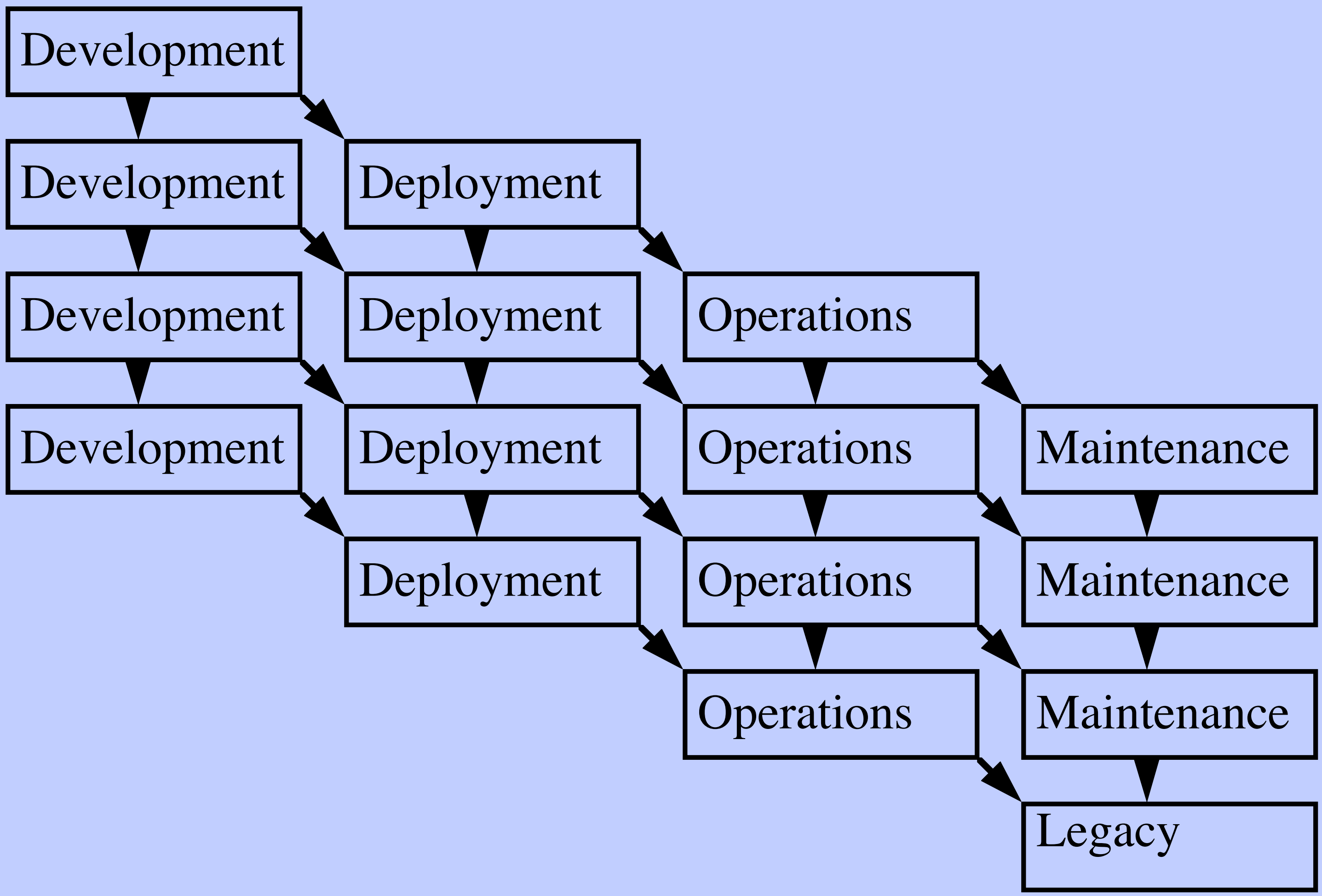
The Field of Software Engineering

Feedback	plan repair, re-planning, process changes, plan optimization, chronic problem management, ...
Measure	number of faults both reported and fixed, lines of code, closeness to plan, resource utilization, performance, ...
Strategies	**methodologies**, architecture, paradigms, mission, risk analysis, scheduling, priority setting, resource utilization, decision making, life cycle management, ...
Tools	compilers, debuggers, environments, quality assurance, CASE, version control, databases, operating systems, networks, file systems, GUI builders, composition, ...
People	group interactions, skill development, group dynamics, communications, goal setting, ...

Methodology

The body of methods, rules, postulates, procedures, and processes that are used to manage a software engineering project through many life cycle stages.

Life Cycle Stages



Four Fundamental Phases

- Define Goals $\longrightarrow$ Analysis
- Establish Plan $\longrightarrow$ Design
- Do the Work $\longrightarrow$ Implementation
- Improve Quality $\longrightarrow$ Testing

Every stage in the life cycle has these four phases.

Phases in the Development Stage

$\boxed{\text{A}}$ • **Analysis:**
- requirements, domain ontology, things, actions, states, events, typical and atypical scenarios

$\boxed{\text{D}}$ • **Design:**
- architecture, implementation plan, performance analysis, test plan

$\boxed{\text{I}}$ • **Implementation:**
- the code

$\boxed{\text{T}}$ • **Testing:**
- quality improvements, regression test, internal testing, unit testing, application testing, stress testing

Process Management Options

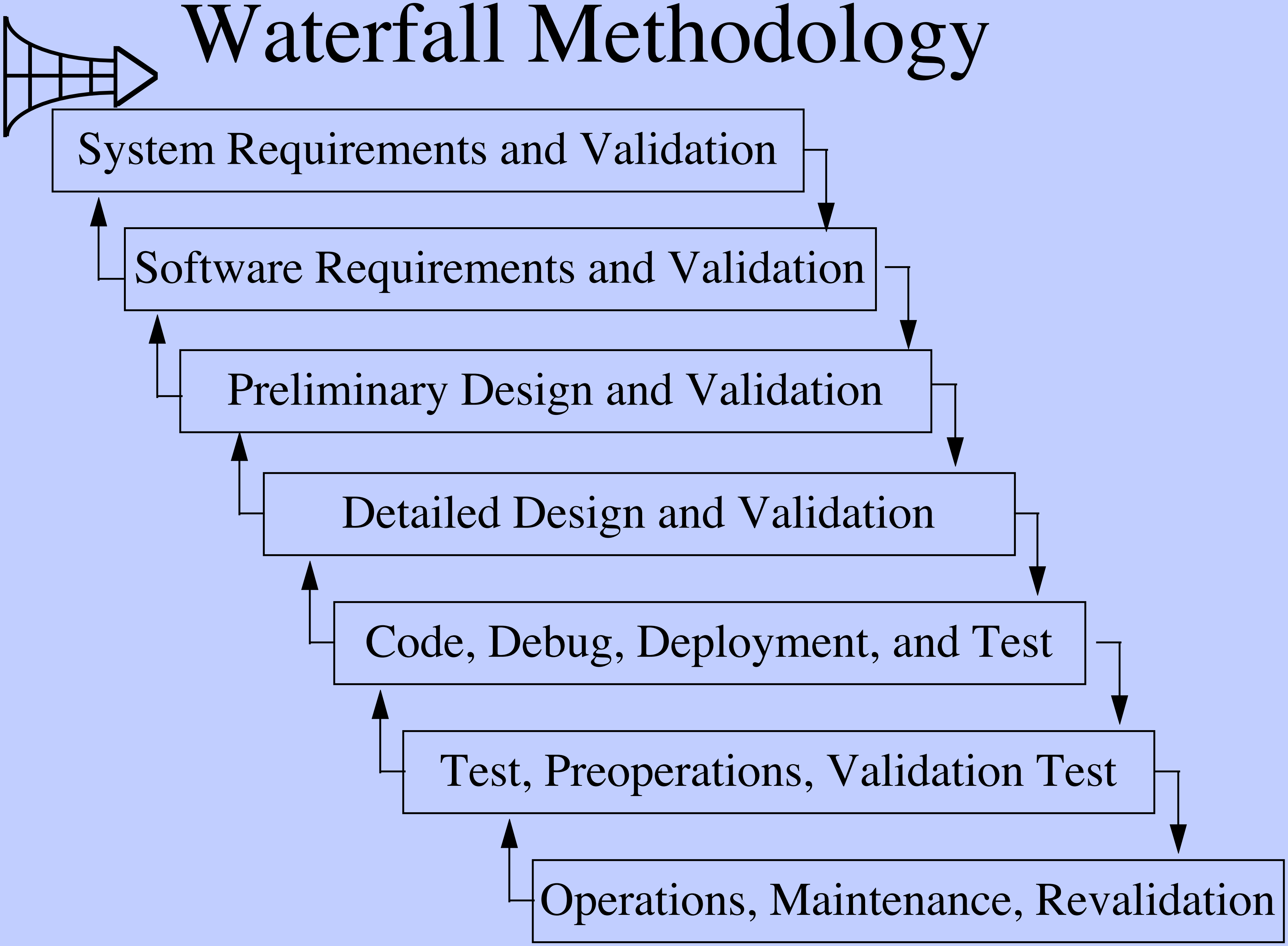
- Methodologies
  - Sequential (Waterfall)
  - Cyclical (Spiral)
  - Best-First (WaterSluice)
- Versions
  - The project may go through several versions.
  - Each version replays the methodology with the previous version used as a starting point for the next version.
  - Some features may be deferred to a later version.

Sequential Methodology

$$\boxed{A} \rightarrow \boxed{D} \rightarrow \boxed{I} \rightarrow \boxed{T}$$

Sequential with Versions

Waterfall Methodology

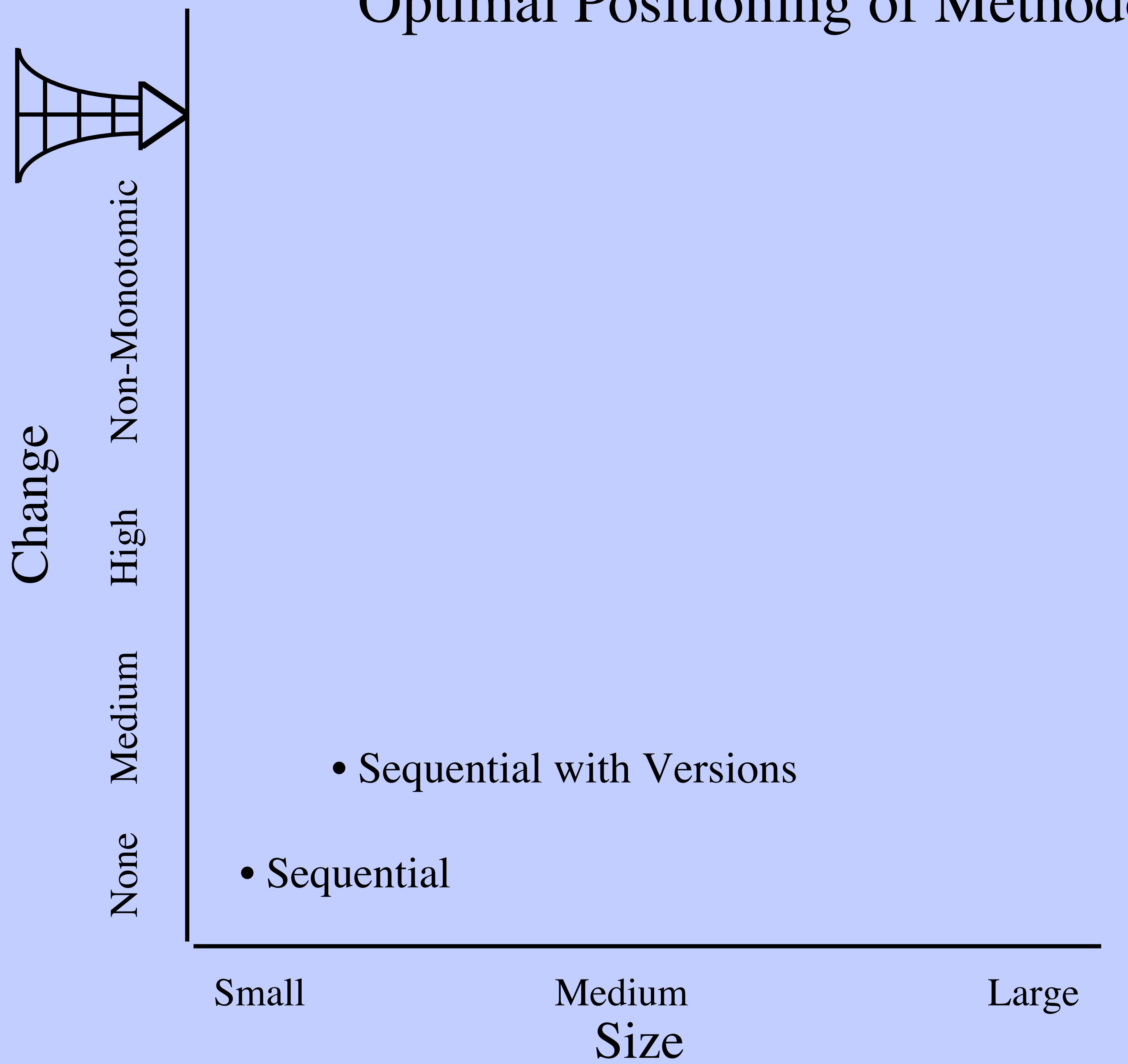


Animation Clip

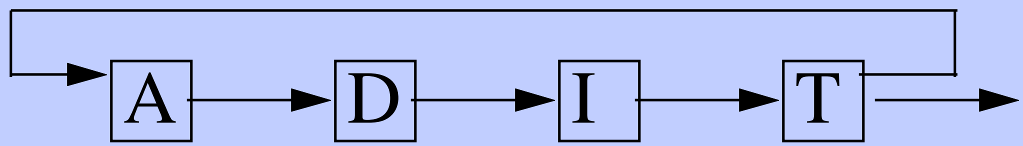
Sequential with Versions Pro and Con

- Pro
  - well established
  - works on qusi-static projects
- Con
  - does not scale to large projects in dynamic environments

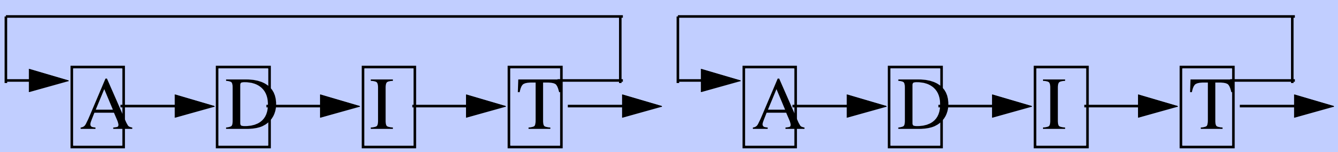
Optimal Positioning of Methodology



Cyclical Methodology

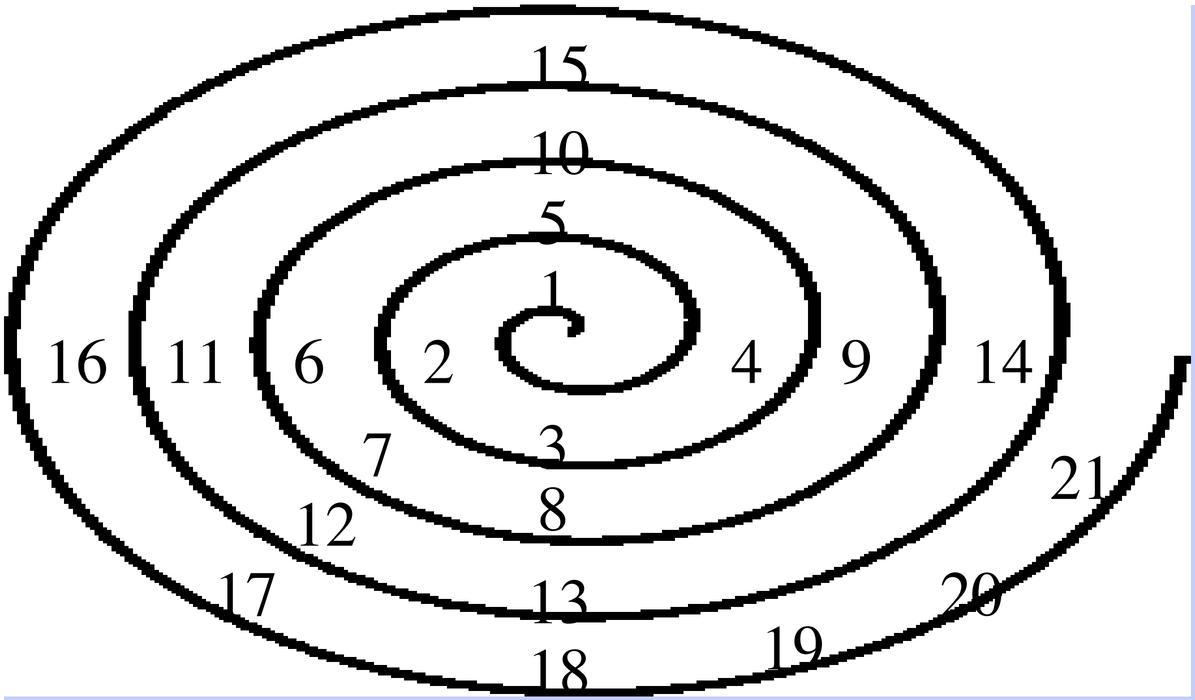


Cyclical Methodology with Version



Version 1 Version 2

Traditional Spiral Methodology



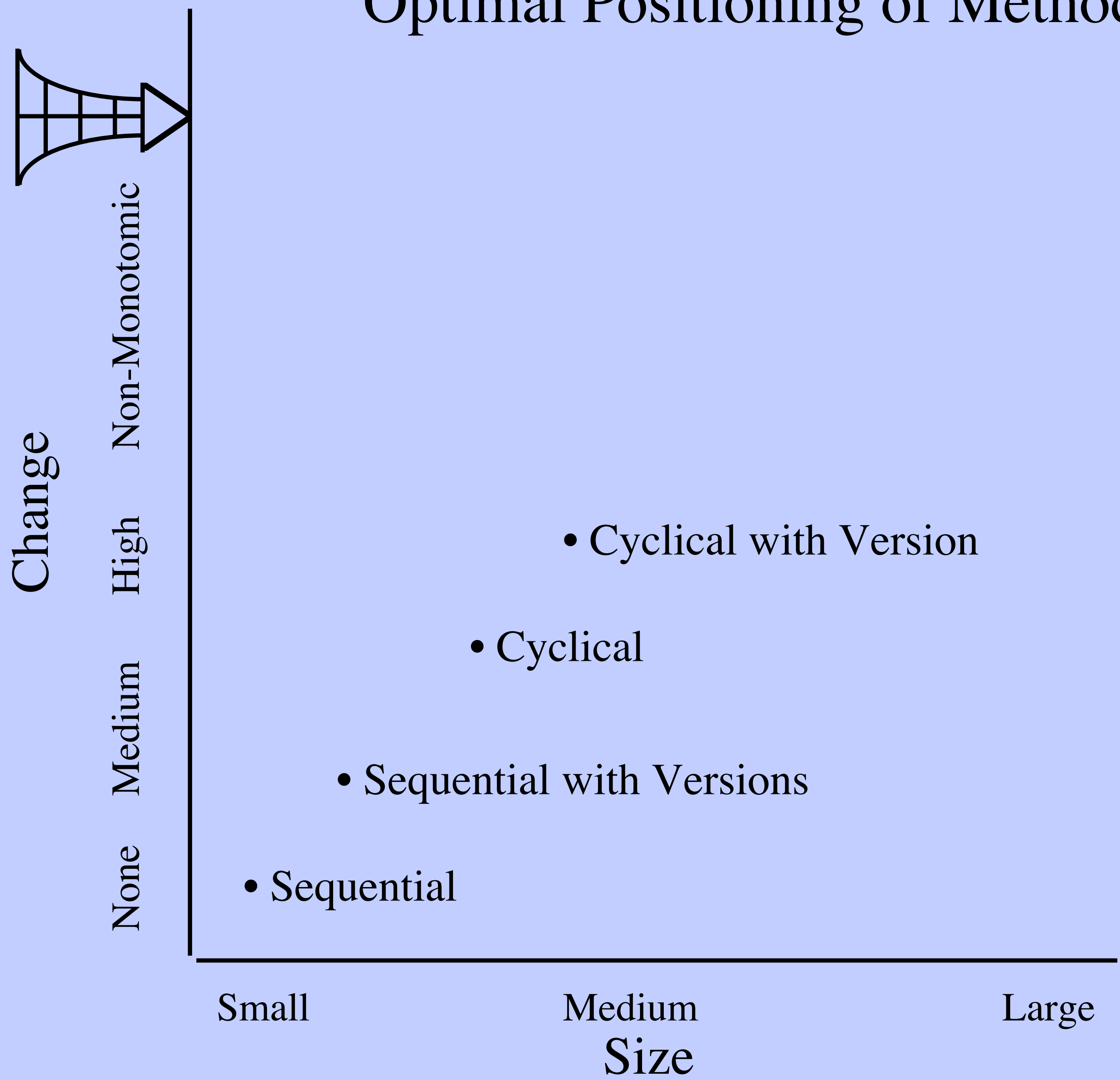
1 Objectives, Alternatives, and Constraints
2 Risk Analysis and Prototype
3 Concept of Operation
4 Requirement and Life-cycle Plan
5 Objectives, Alternatives, and Constraints
6 Risk Analysis and Prototype
7 Simulation, Models, and Benchmarks
8 Software Requirements and Validation
9 Development Plan
10 Objectives, Alternatives, and Constraints
11 Risk Analysis and Prototype
12 Simulation, Models, and Benchmarks
13 Software Product Design, Validation, and Verification
14 Integration and Test Plan
15 Objectives, Alternatives, and Constraints
16 Risk Analysis and Operational Prototype
17 Simulation, Models, and Benchmarks
18 Detailed Design
19 Code
20 Unit, Integration, and Acceptance Testing
21 Implementation (Deployment)

Animation Clip

Cyclical with Versions Pro and Con

- Pro
  - feedback path
- Con
  - no governors
  - no priority
  - no conflict management
  - may diverge instead of converge

Optimal Positioning of Methodology



Best-First Methodology

- Borrow the steady progression of the sequential methodology.
- Borrow the iterative nature of the cyclical methodology.
- Add goal focus
- Add priority (cost)
- Add Non-monotonic governor
  - change order control

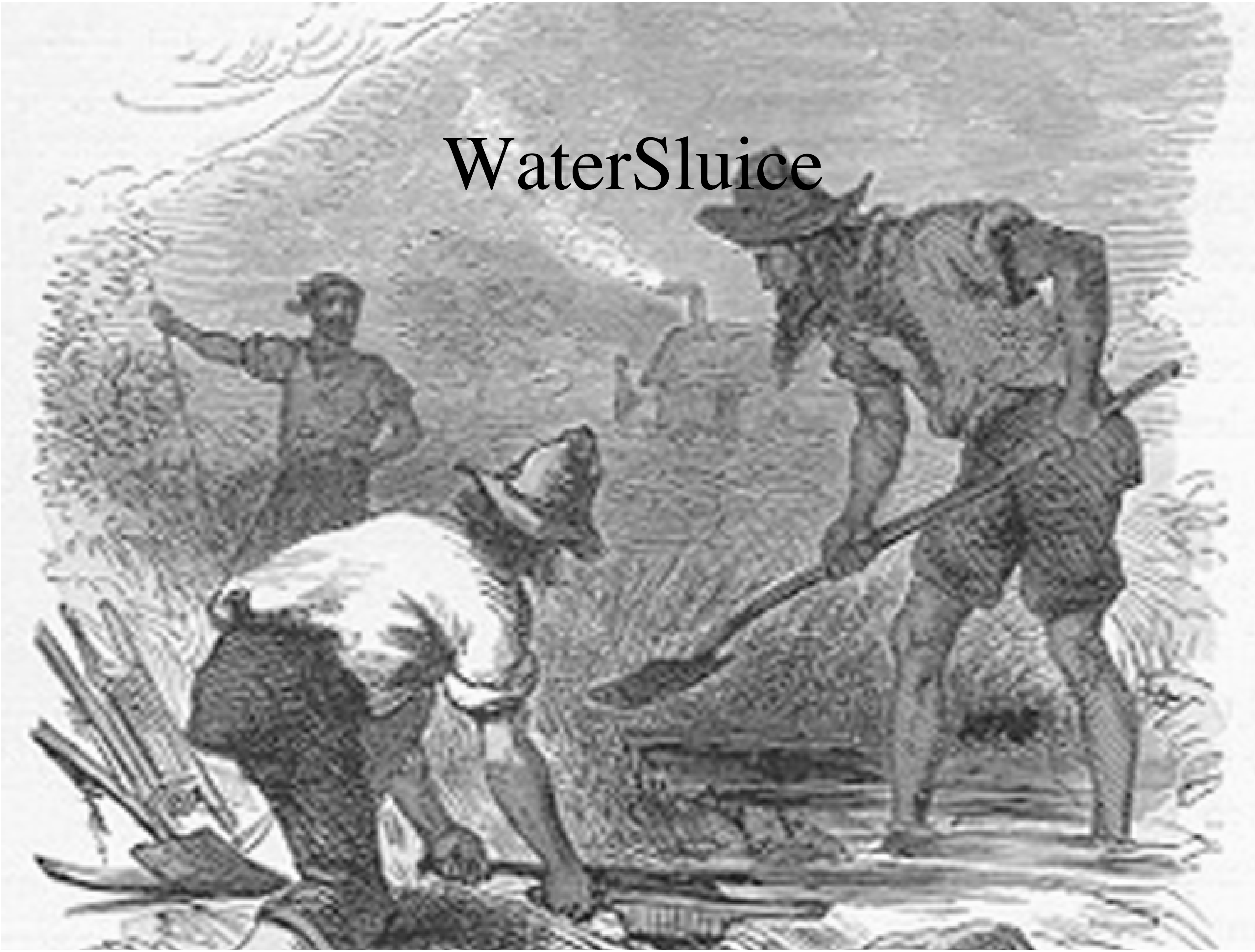
Priority Function

- Each potential task is assigned a priority.
- This priority reflects the benefit to the final goal of accomplishing the task based on what has already been accomplished.
- The highest priority task is accomplished next.

Change Order Control

- Process to manage change.
- Once a component is completed to the satisfaction of the team it is placed under change order control and frozen.
- Only absolutely necessary changes are allowed.
- Changes should be seldom, well justified, and documented.

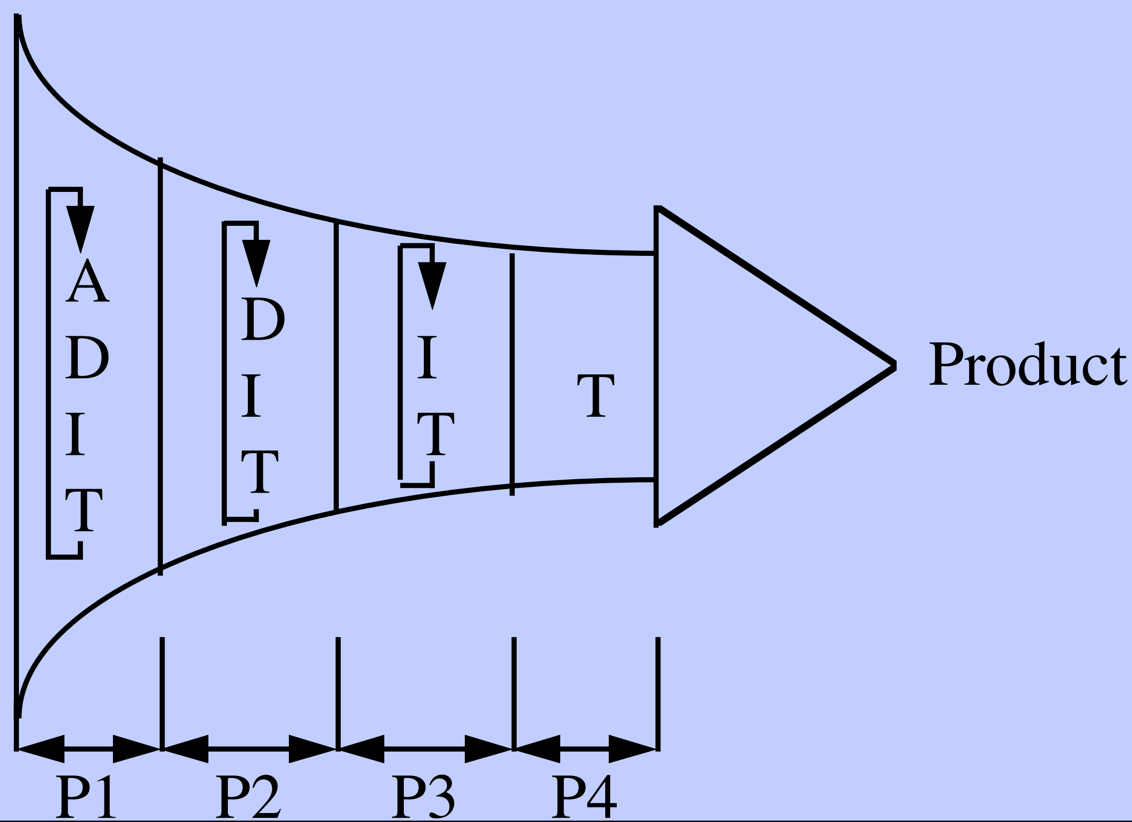
WaterSluice



The WaterSluice Methodology

A: Analysis
D: Design
I: Implementation
T: Testing

P1: Proof of Principle
P2: Prototype
P3: Alpha and Beta
P4: Product

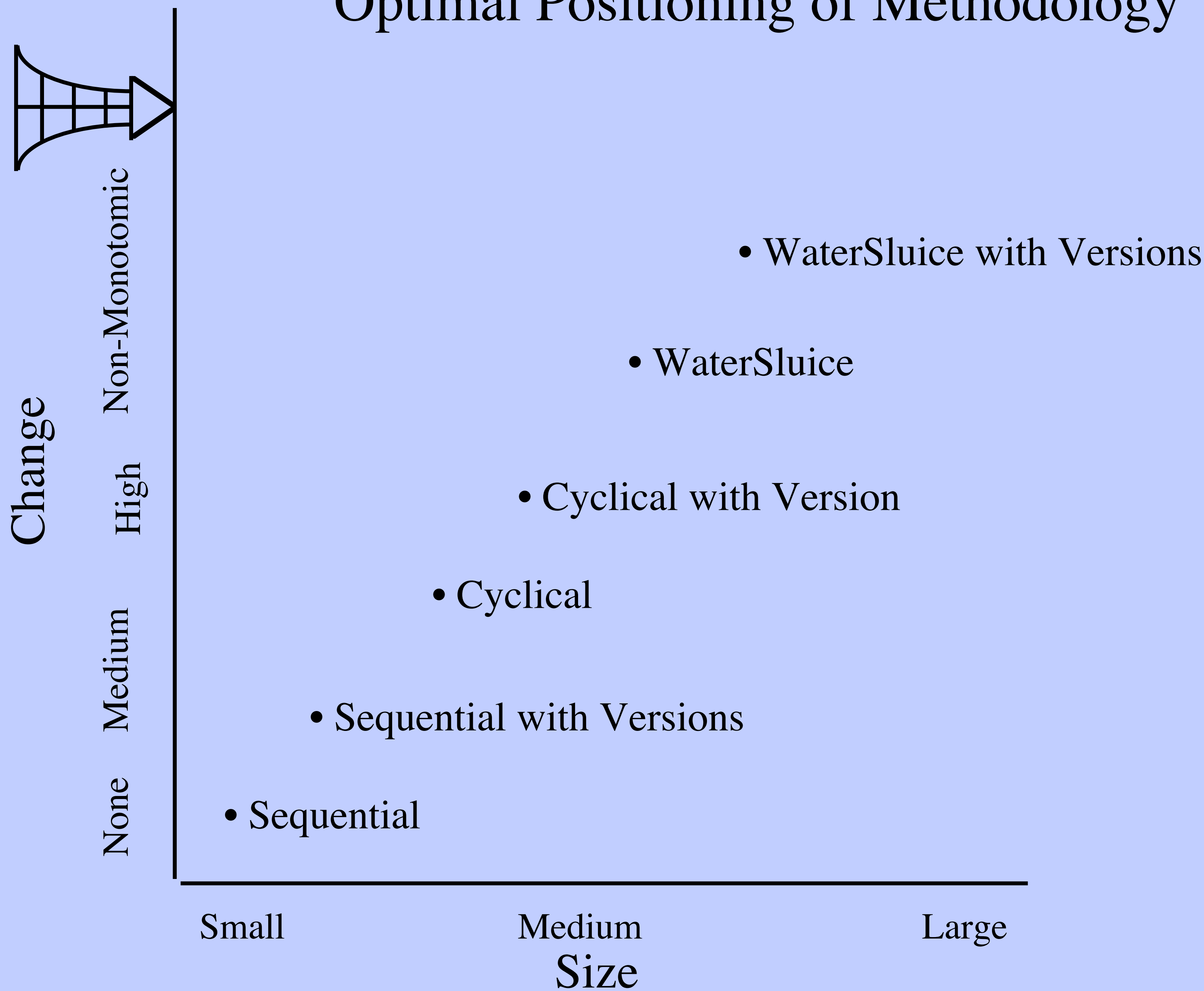


Animation Clip

WaterSluice - Pro and Con

- Pro
  - feedback path
  - governors
  - priority
  - goal directed
  - forces converges
- Con
  - more complex

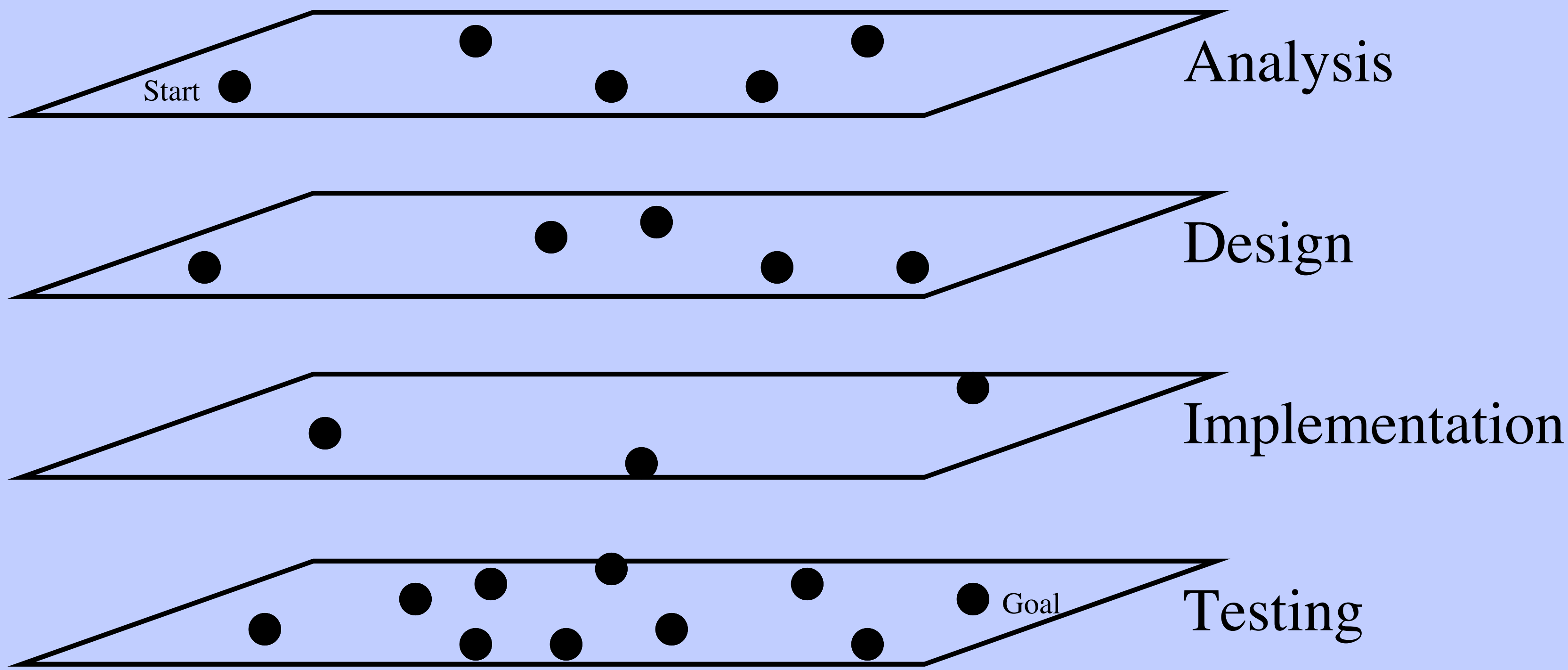
Optimal Positioning of Methodology



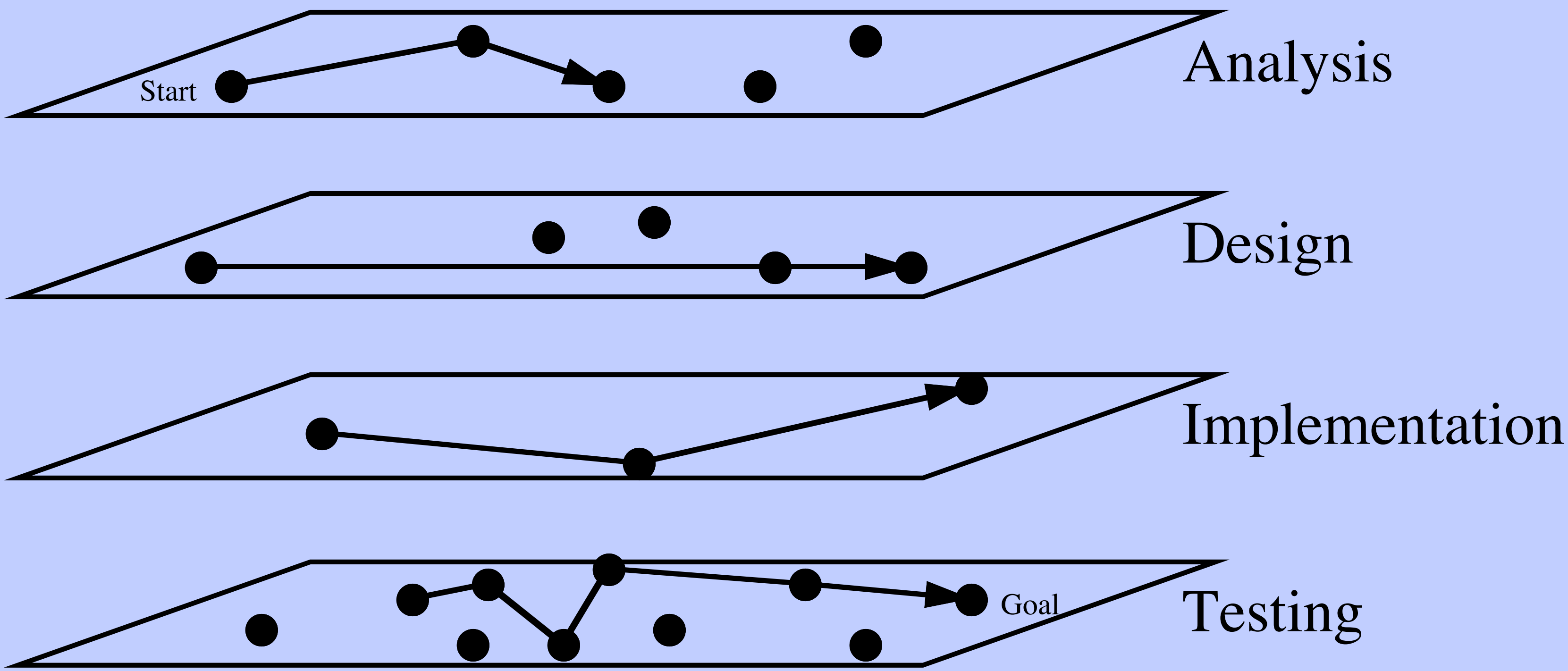
Theory

- Map waterfall to breadth-first search
- Map spiral to depth-first search
- Map WaterSluice to best-first search

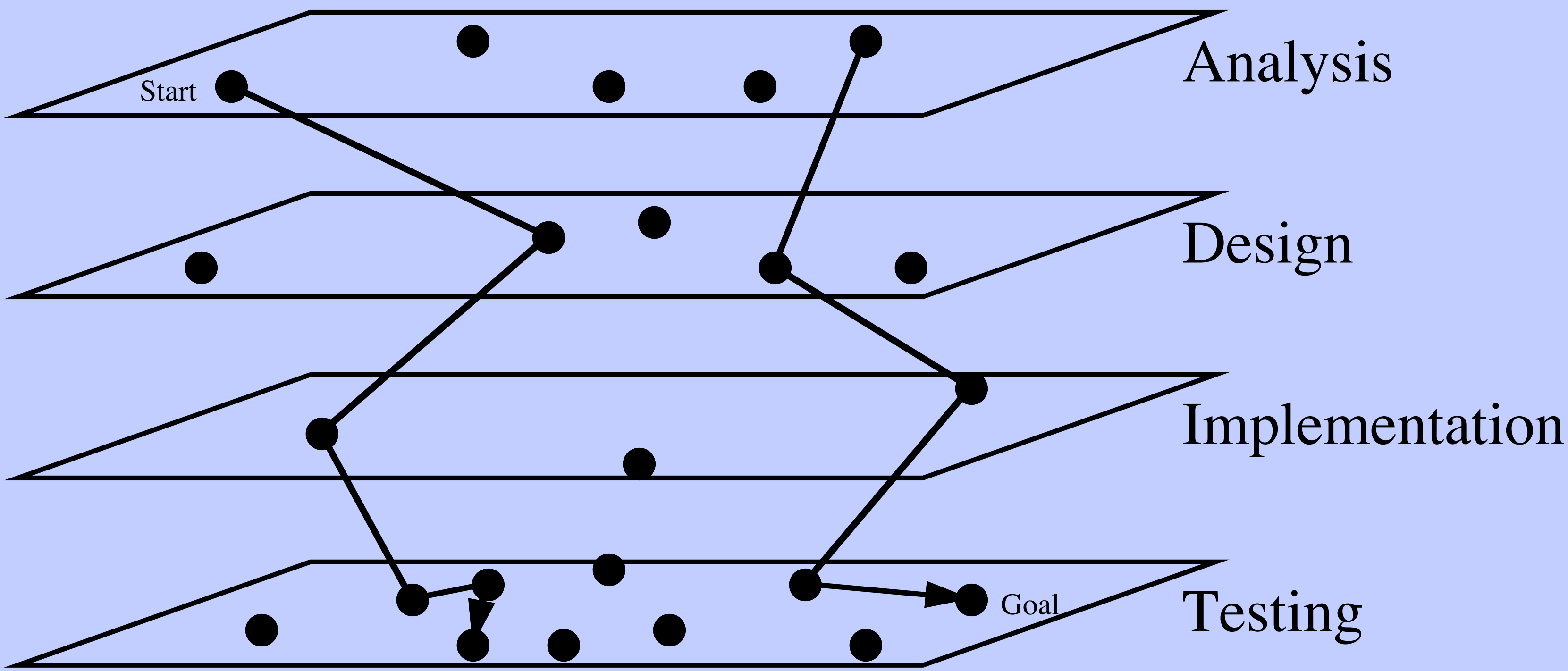
Search Space



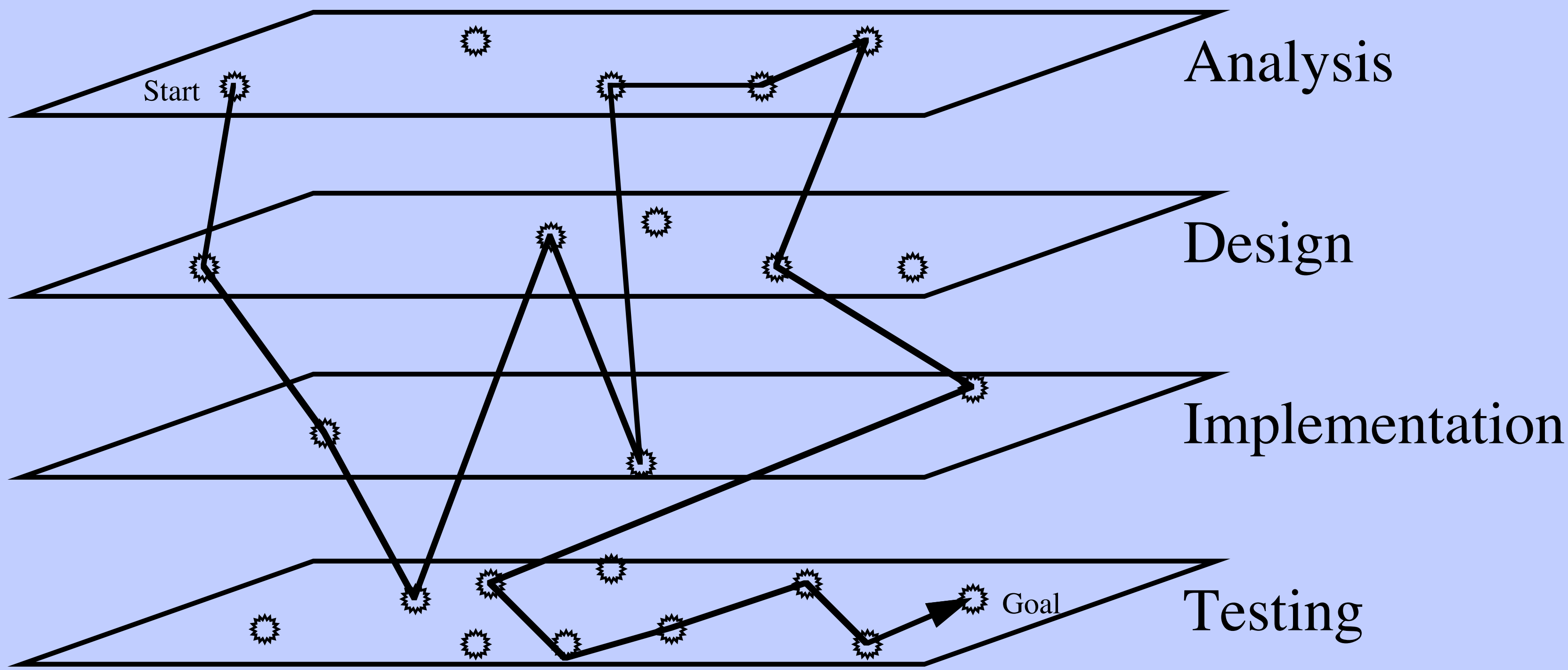
Waterfall: Breadth-First Search



Spiral: Depth-First Search



WaterSluice: Best-First Search



Summary of Theorems

Methodology	Static Complete	Dynamic Complete	Dynamic Optimum	Performance
waterfall	yes	no	no	good
spiral	yes	yes	no	good->better
WaterSluice	yes	yes	yes	good->best

Other Work

- DADL - Distributed Architecture Definition Language
- Noema - Engineering paradigm
- CHAIMS - Component engineering
- Distributed Computer Environments

The Thesis

- Software Engineering
- Methodologies
- Requirements
  - two examples
- Implementation and Testing
  - C++ container classes
- Decision Making

Conclusion

The WaterSluice methodology borrows the iterative nature of the cyclical methodologies and the steady progression of the sequential methodologies and then adds priority and change order control.

The WaterSluice methodology will work best in a very dynamic environments as compared to the sequential or cyclical methodologies.