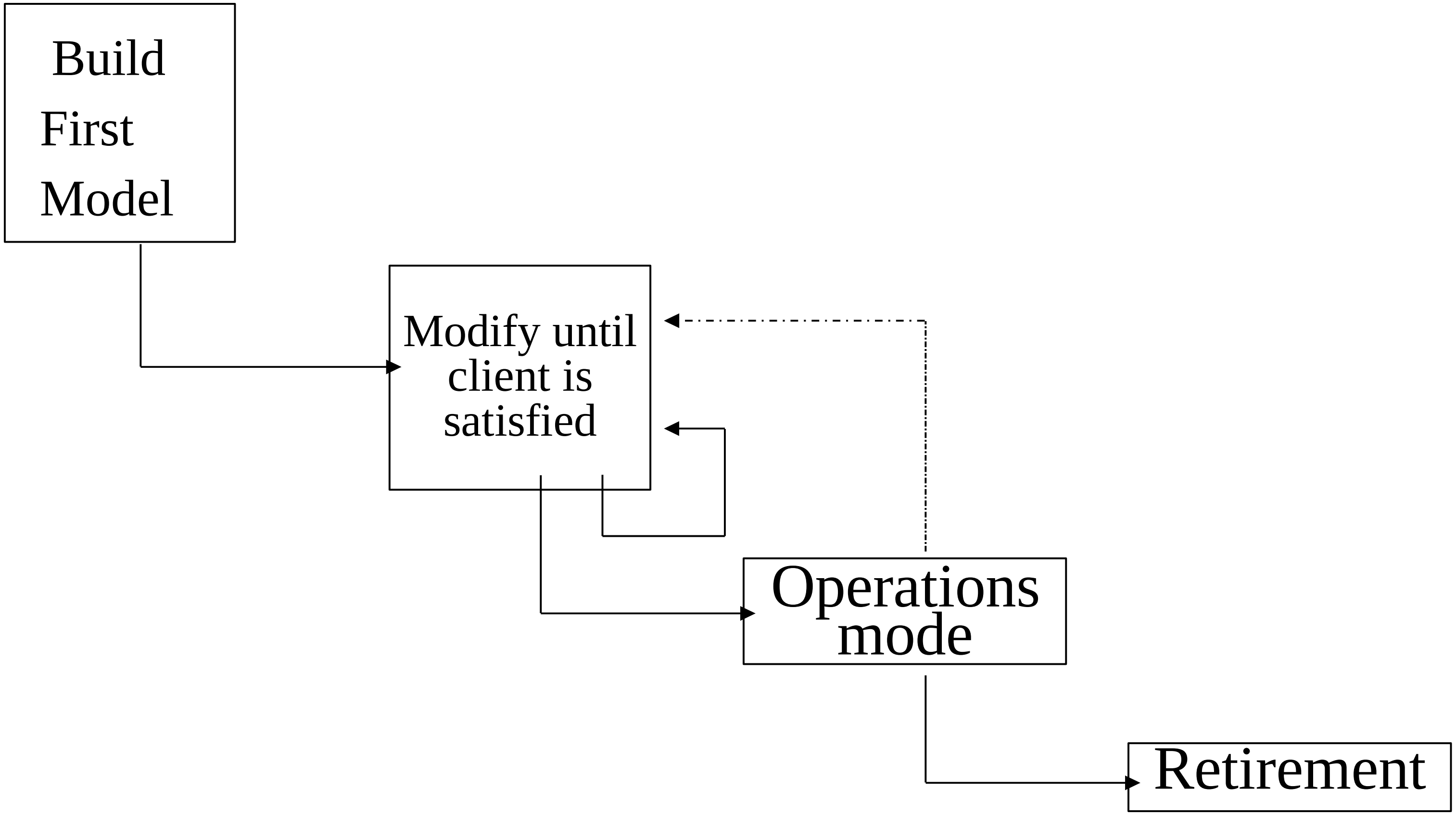
Software Project Management

Lecture-4

Software Life-Cycle Models

Build ad Fix
Water fall
Prototyping
RAD
Incremental
Spiral
RUP

Build and Fix


Build First Model

Modify until client is satisfied

Operations mode

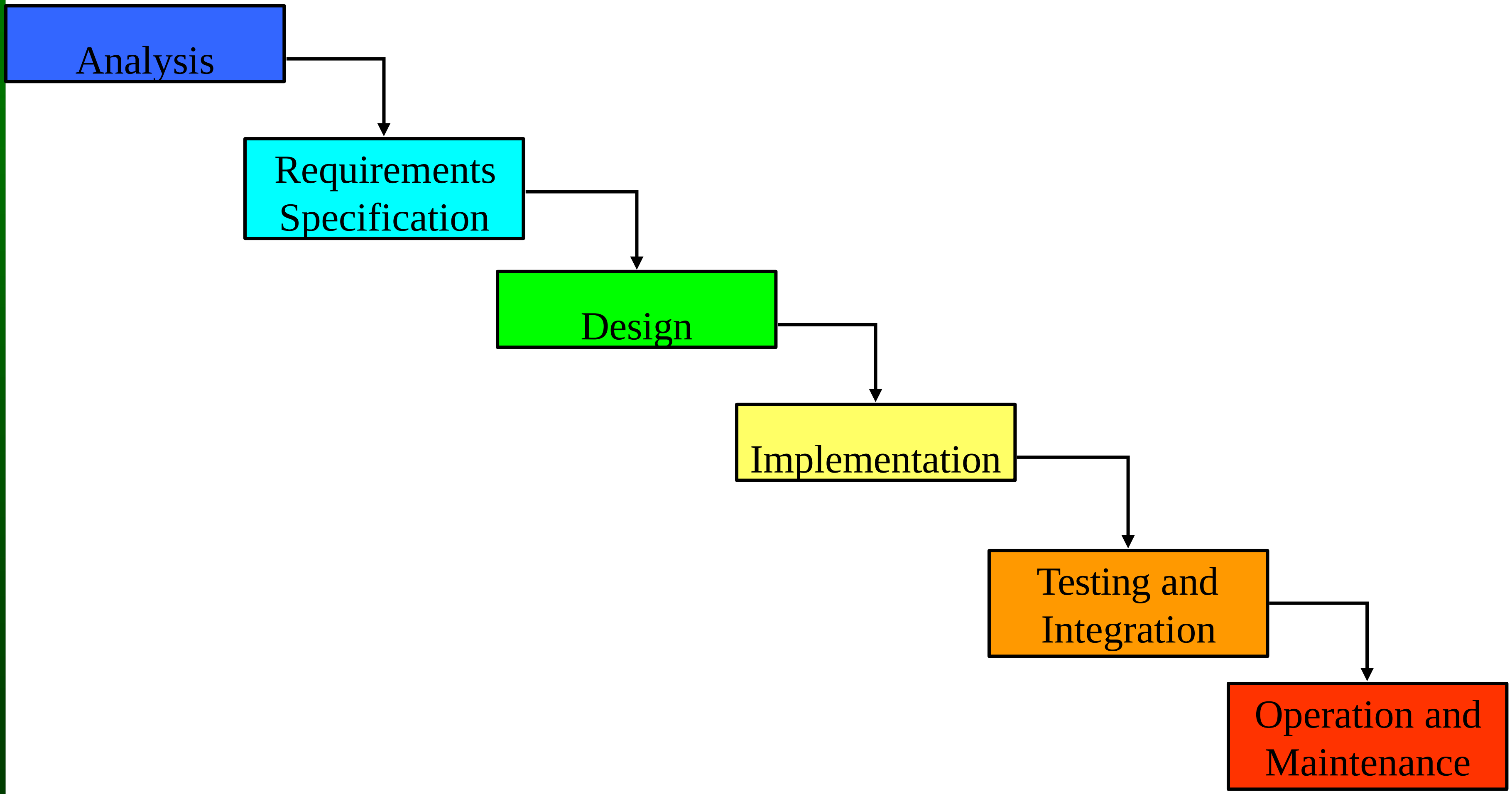
Retirement

Build and Fix

Build it and fix until client is happy

No formal specifications
No formal design
No documentation
Rework is out of control
Loss of client confidence
High cost
Future maintenance difficult/impossible

Waterfall Model



Analysis

Requirements Specification

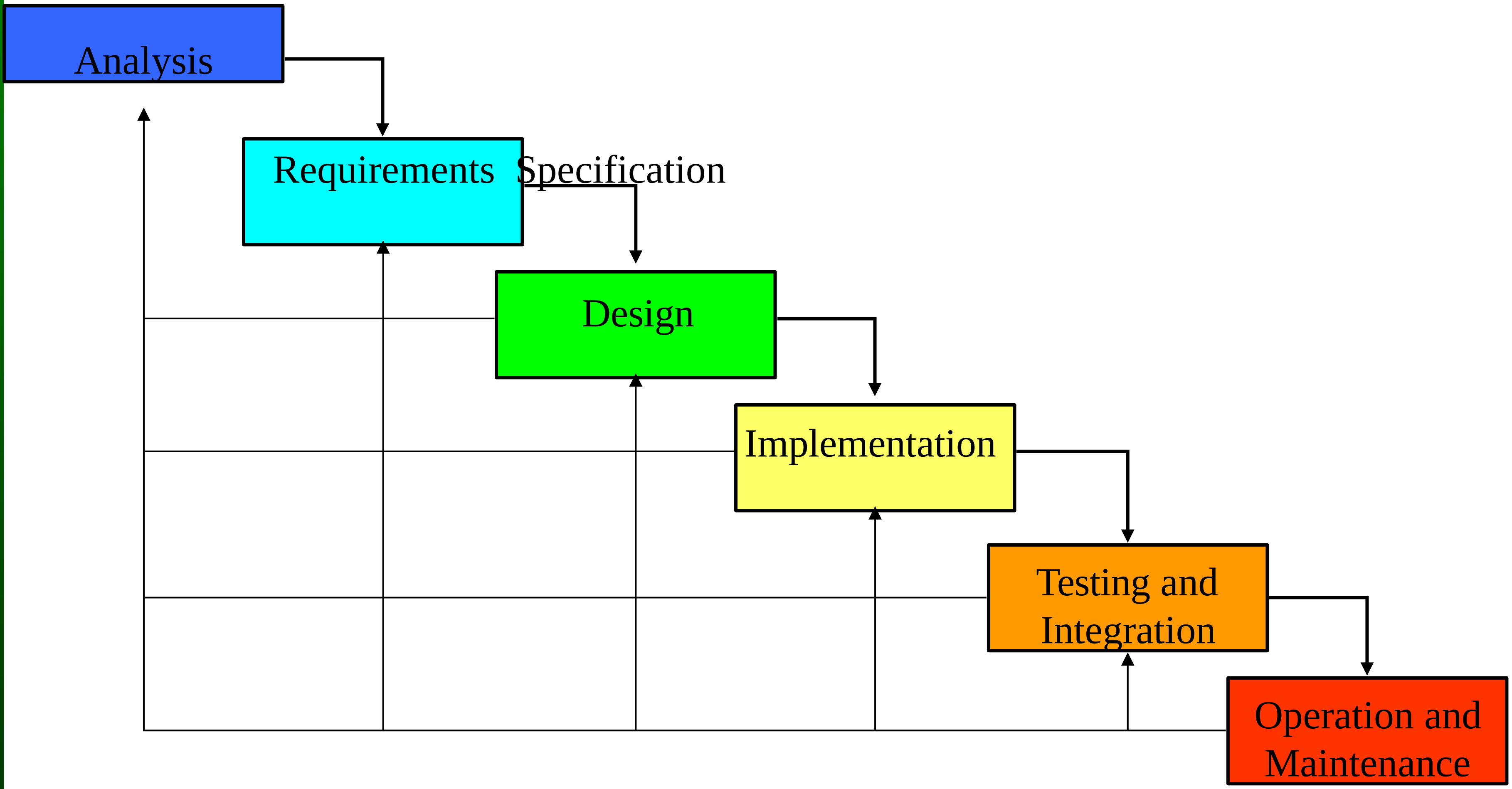
Design

Implementation

Testing and Integration

Operation and Maintenance

Waterfall Reality



Waterfall Model

Advantages:

It is easy to understand

Built-in documentation and quality assurance at each stage

Standard procedure and techniques

It is easy to track the progress of the project using a timeline

Better control over development

Waterfall Model

Disadvantages

????

Prototyping

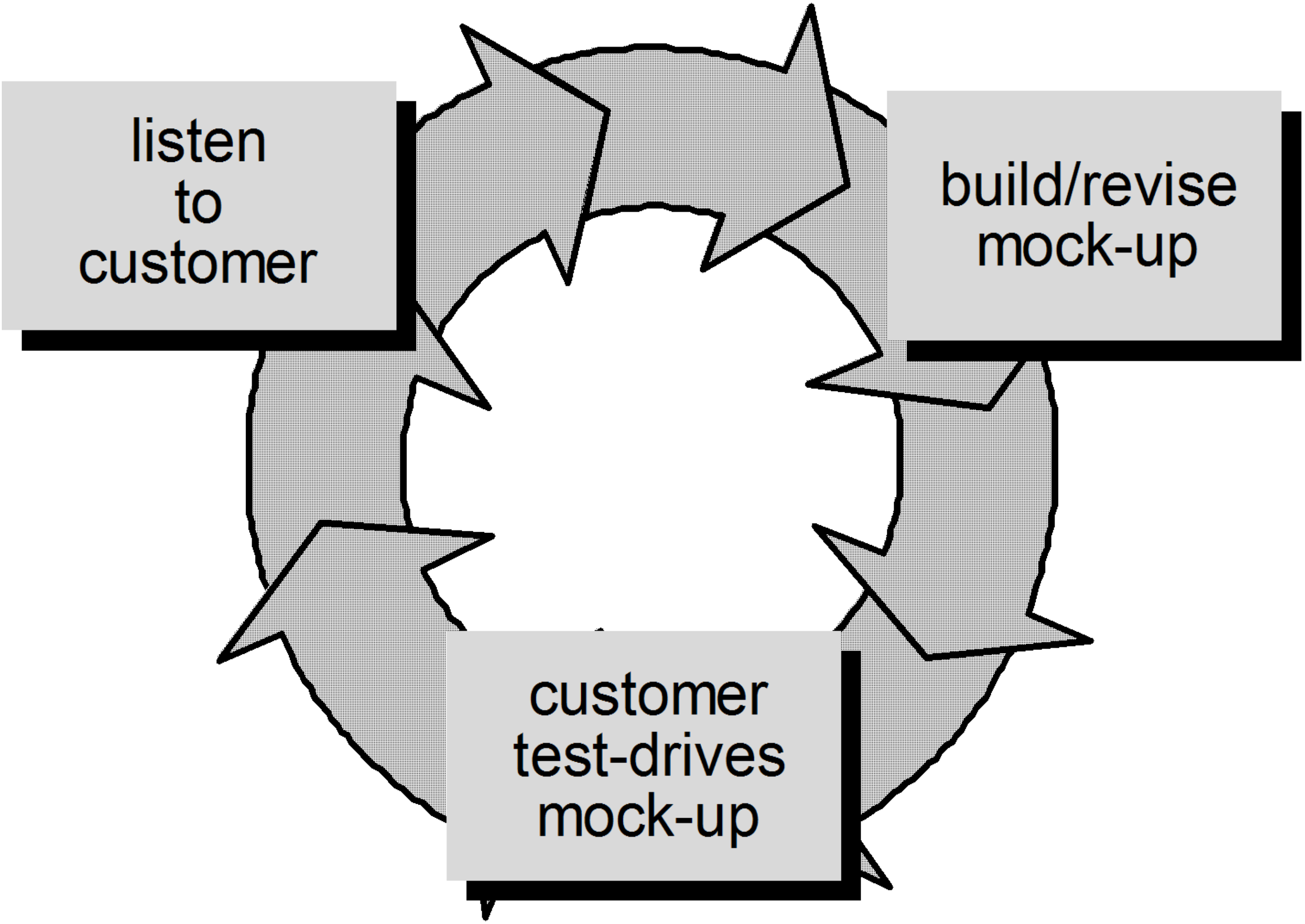


Figure © Pressman Associates

Prototyping

Advantages

Users/Clients “see and feel” the proposed product before it is fully implemented

Users get an earlier sense of something happening

Developers get a better feel for the design

Developers work closer with users in design

Prototyping

Disadvantages

????

The RAD Model

Rapid Application Development - James Martin (1991)

A linear sequential software development process that emphasizes an extremely short development cycle.

Rapid development is achieved by using a component-based construction approach.

Used primarily for information systems applications.

RAD projects are typically staffed with small integrated teams comprised of developers and end users.

The RAD Model

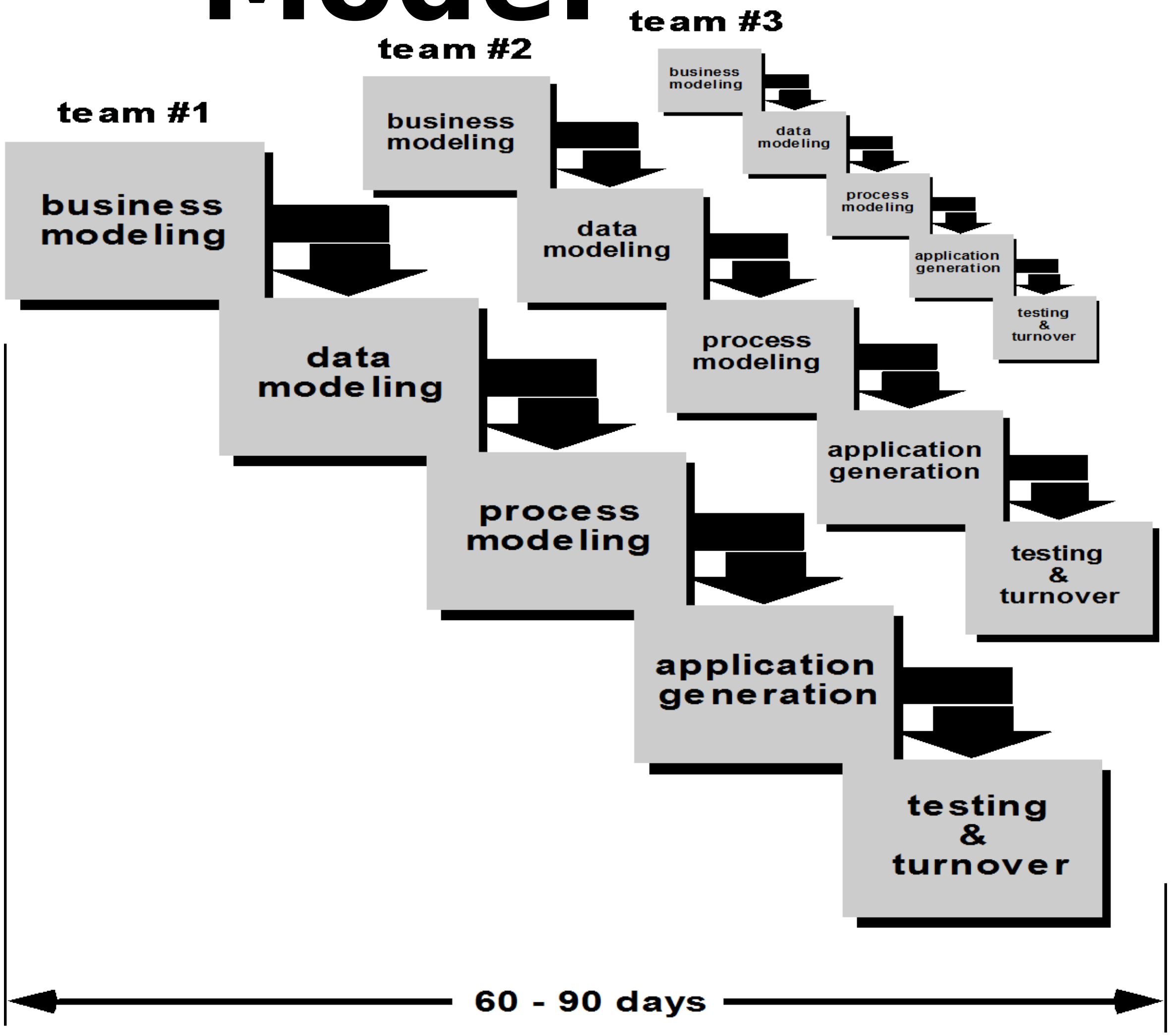


Figure © Pressman Associates

Advantages of the RAD

The model has shown reduced cycle time due to the use of powerful development tools

The end user is involved throughout the life cycle

The system is developed by a project team familiar with the problem domain; thus it continues to build upon this expertise can deliver a full product in a short time period

Disadvantages of the RAD

If the users cannot be involved consistently throughout the life cycle, this can have adverse effects on the final product.

This model requires highly skilled and well-trained developers in the use of the chosen development tools to achieve the rapid turnaround time.

It can require more people

It requires a system that can be properly modularized

It can fail if

Reusable components are not available

Performance is critical

Reliability is critical

The Incremental Model

Linear sequential + Iteration

Each module is delivered incrementally.

For each module

- Detailed design
- Module implementation
- Module integration

The Incremental Model

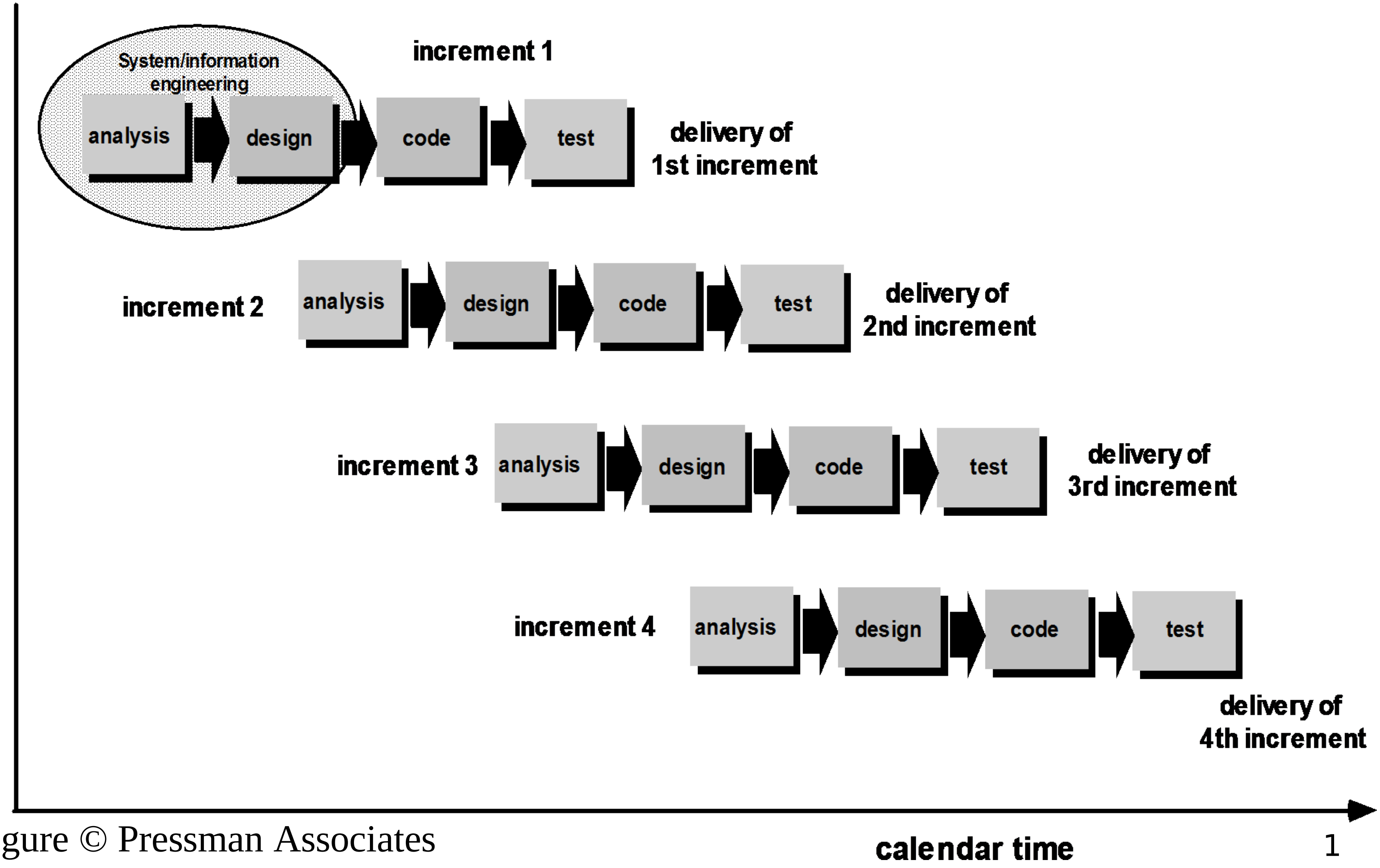


Figure © Pressman Associates

Advantages The Incremental Model

Operational quality portion of product within weeks

Users have earlier use of some components of the system

Changes due to new system are easier to manage piece by piece

Client can make progress payment

Keeps all teams working

Disadvantages the Incremental Model

The definition of a complete, fully functional system must be done early in the life cycle to allow for the definition of the increments

Each build delivery requires total integration testing

System must be incremental in nature

Maintenance on delivered units may affect integration of new units

May turn into Build and Fix

Spiral Model

Boehm (1988)

Evolutionary software process model

Iterative nature of prototype +
controlled and systemic aspect of linear sequential model

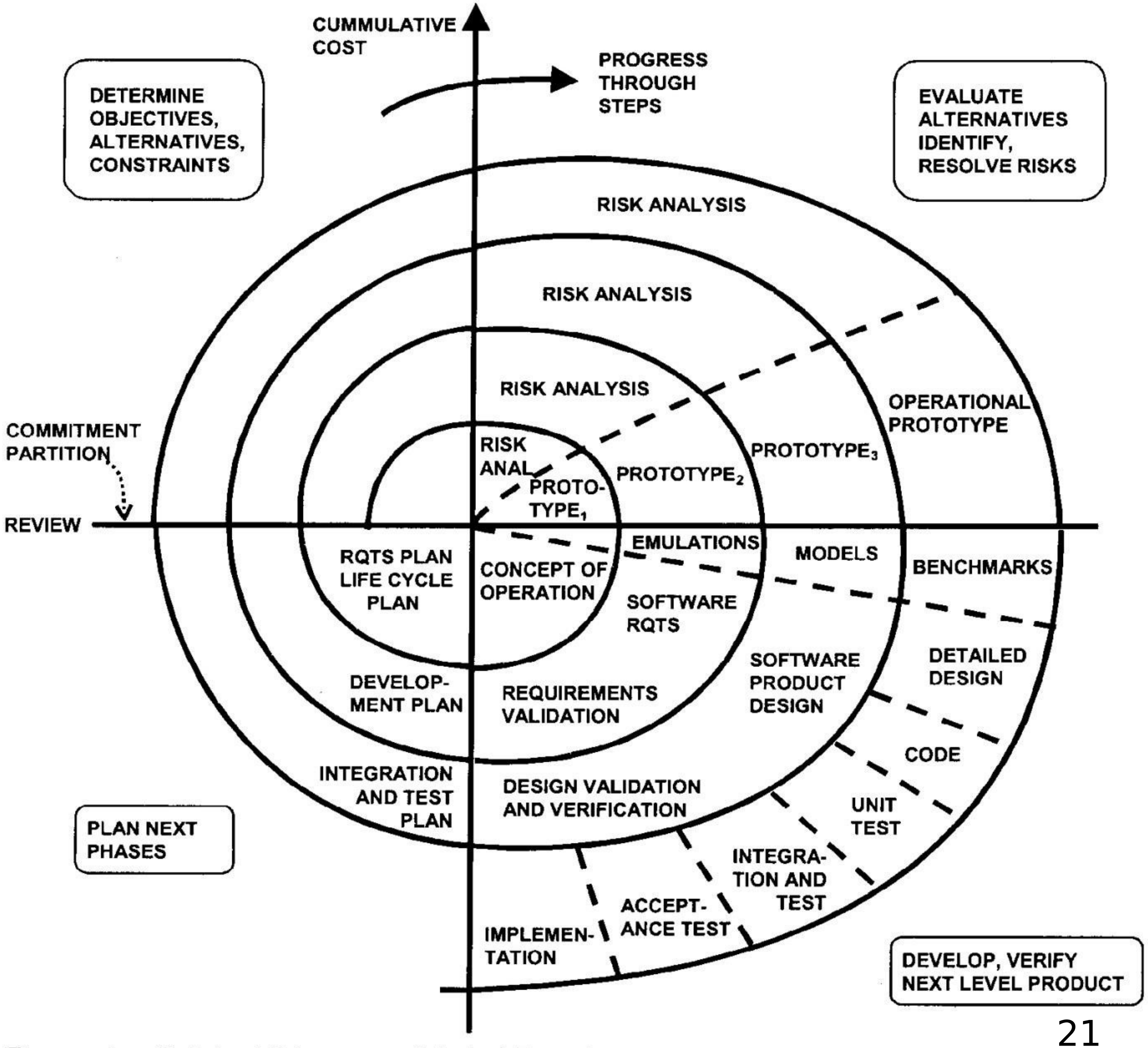
Each stage requires risk analysis before entry

Each stage has verification on completion

Spiral Model

Original Model (1988)

Four sectors



Spiral Model

Development team moves around the spiral beginning at the center.

And began each cycle of the spiral by performing the next level of elaboration of the perspective system’s

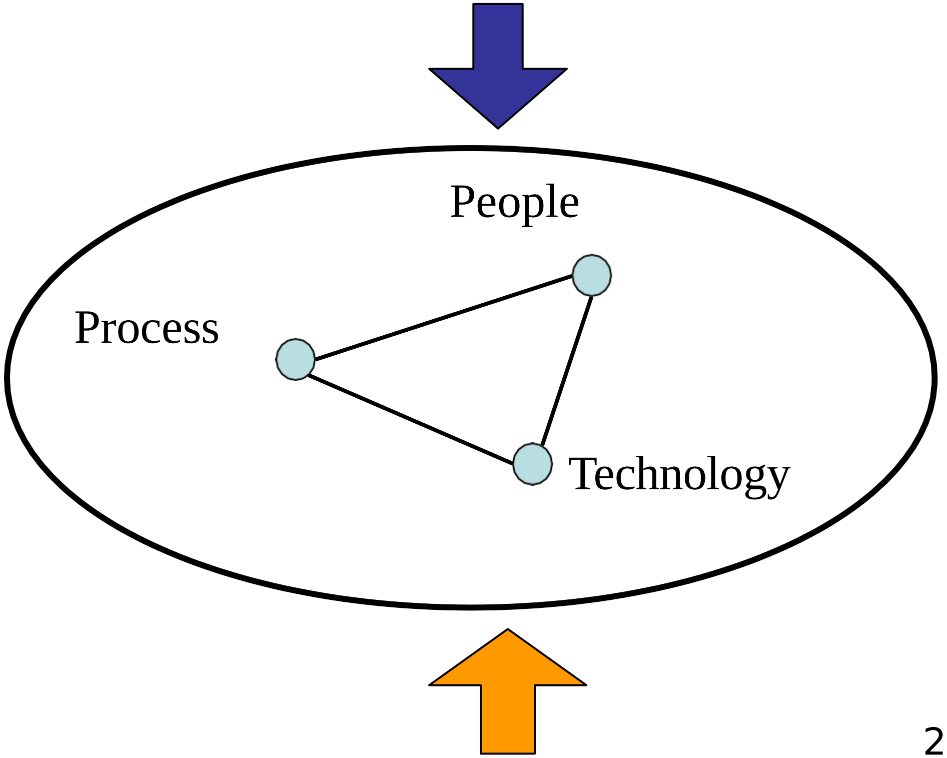
objectives,

constraints, and

alternatives.

RUP

- Rational Software Development Process
- Two Perspective
  - Management
  - Technical



RUP

A management perspective

the financial,

strategic,

commercial, and

human aspects

A technical perspective

dealing with quality

engineering and

design method aspects

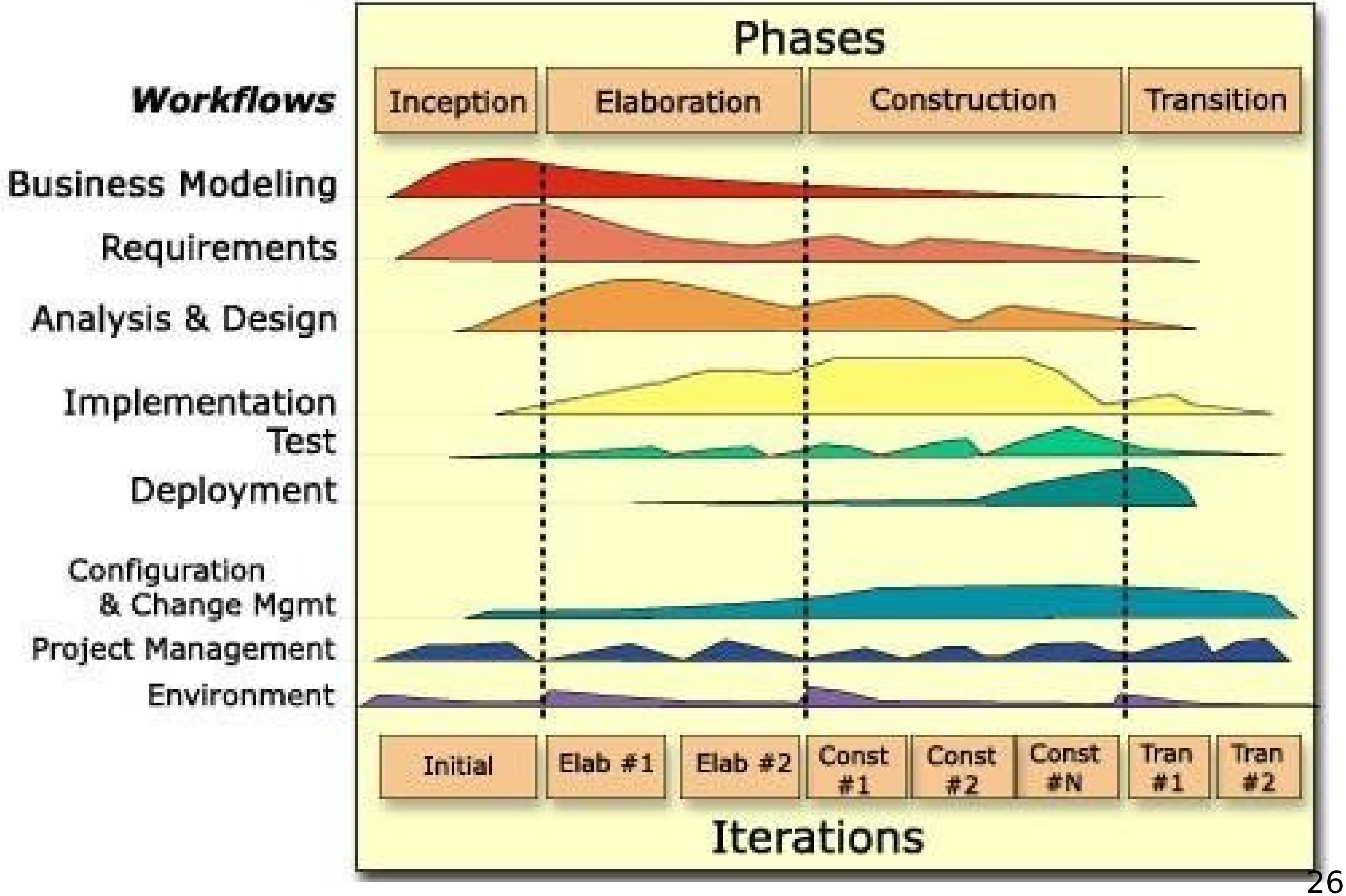
RUP

The software lifecycle is divided into cycles, each cycle working on new generation of the product

One development cycle is divided in four consecutive phases

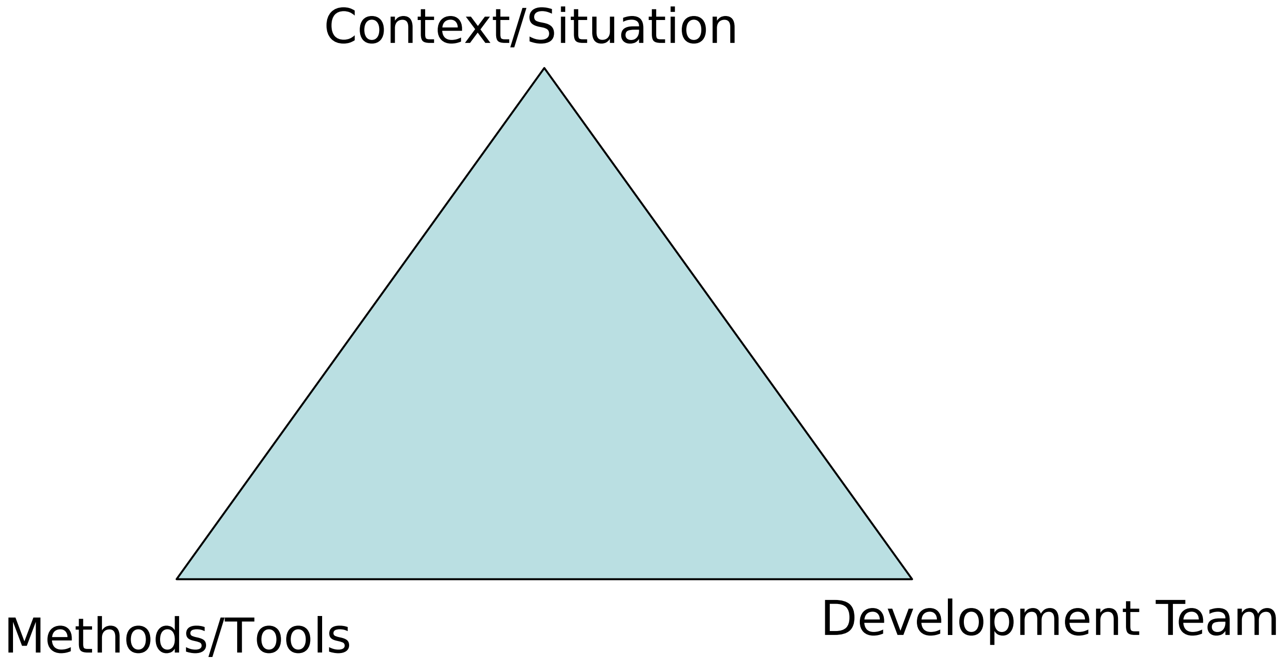
- Inception
- Elaboration
- Construction
- Transition

RUP Phases and Workflows

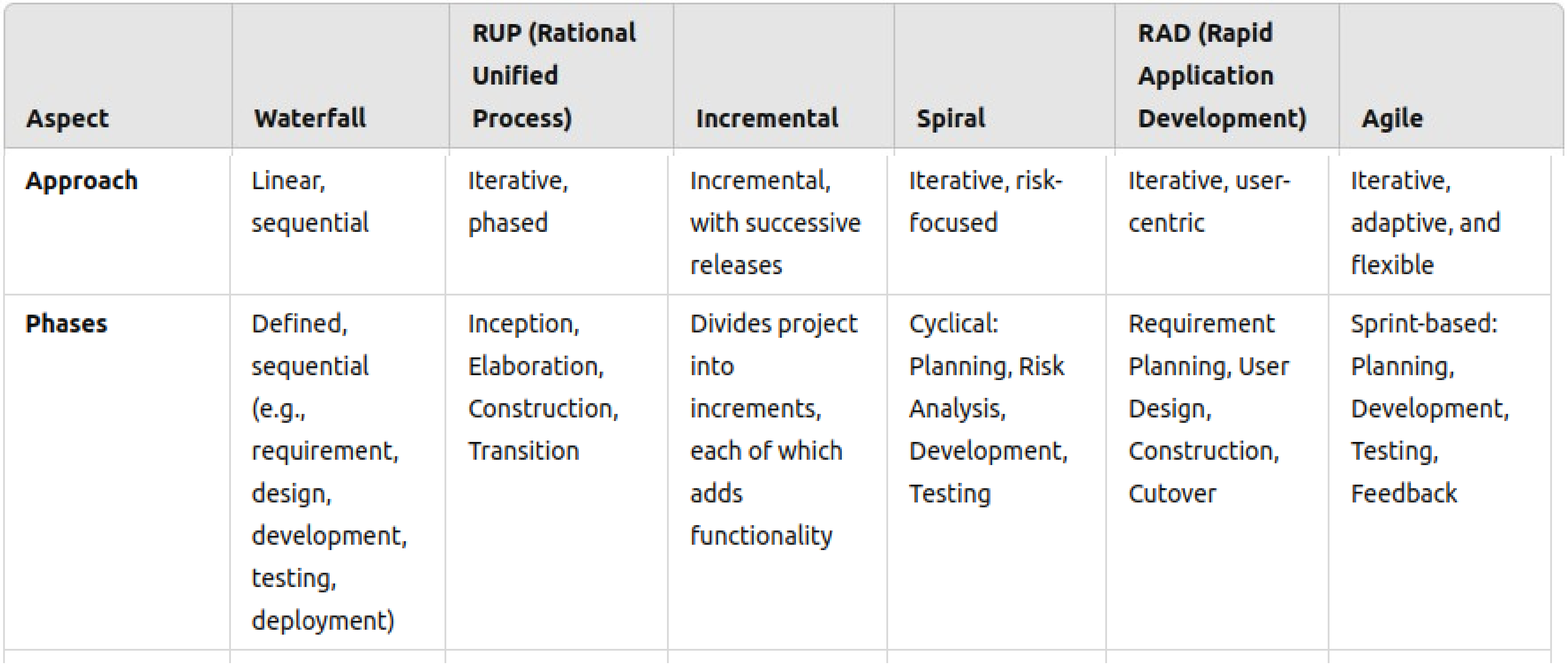


Which Model?

Selecting SLC Model

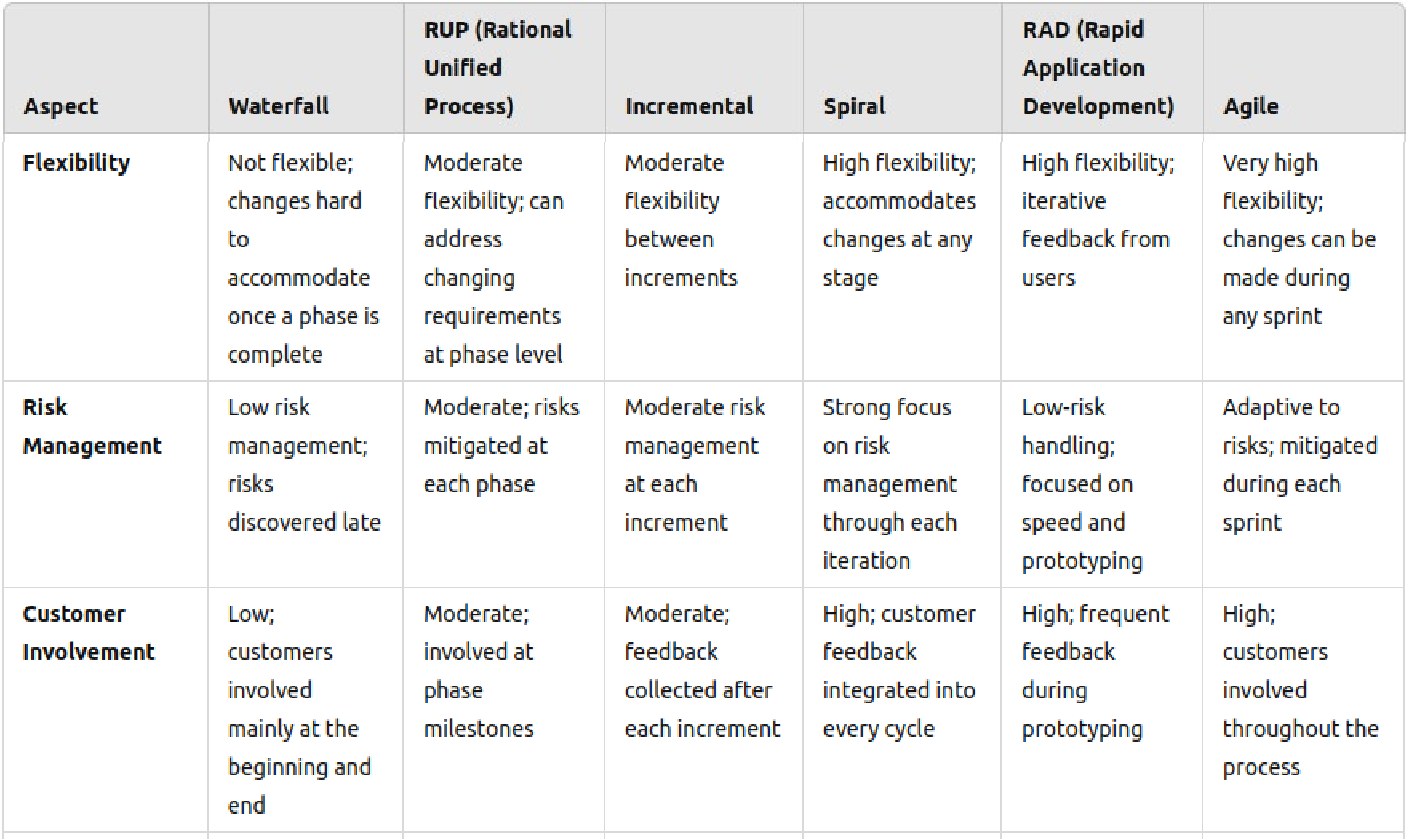


SLC Model Comparison



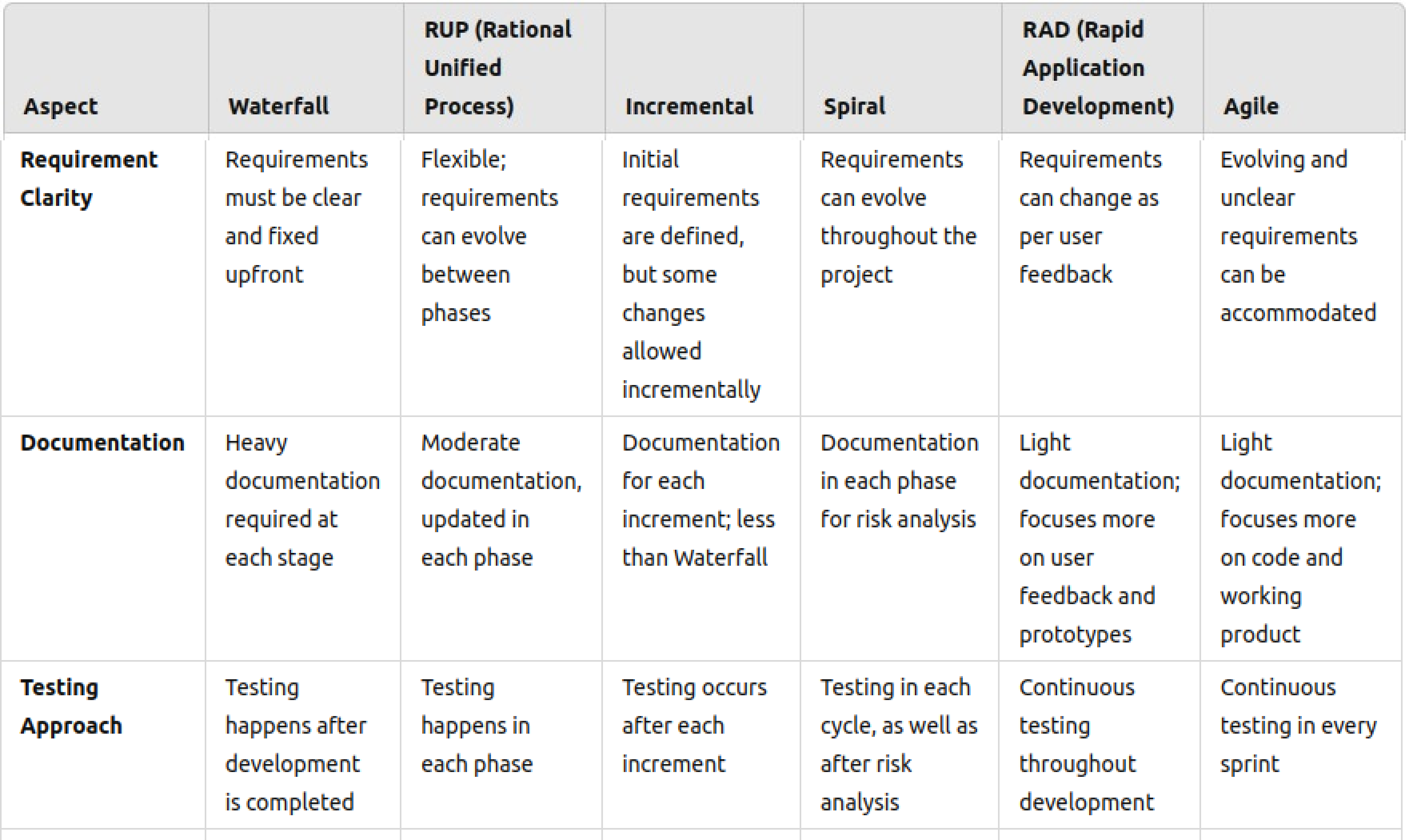
Aspect	Waterfall	RUP (Rational Unified Process)	Incremental	Spiral	RAD (Rapid Application Development)	Agile
Approach	Linear, sequential	Iterative, phased	Incremental, with successive releases	Iterative, risk-focused	Iterative, user-centric	Iterative, adaptive, and flexible
Phases	Defined, sequential (e.g., requirement, design, development, testing, deployment)	Inception, Elaboration, Construction, Transition	Divides project into increments, each of which adds functionality	Cyclical: Planning, Risk Analysis, Development, Testing	Requirement Planning, User Design, Construction, Cutover	Sprint-based: Planning, Development, Testing, Feedback

SLC Model Comparison



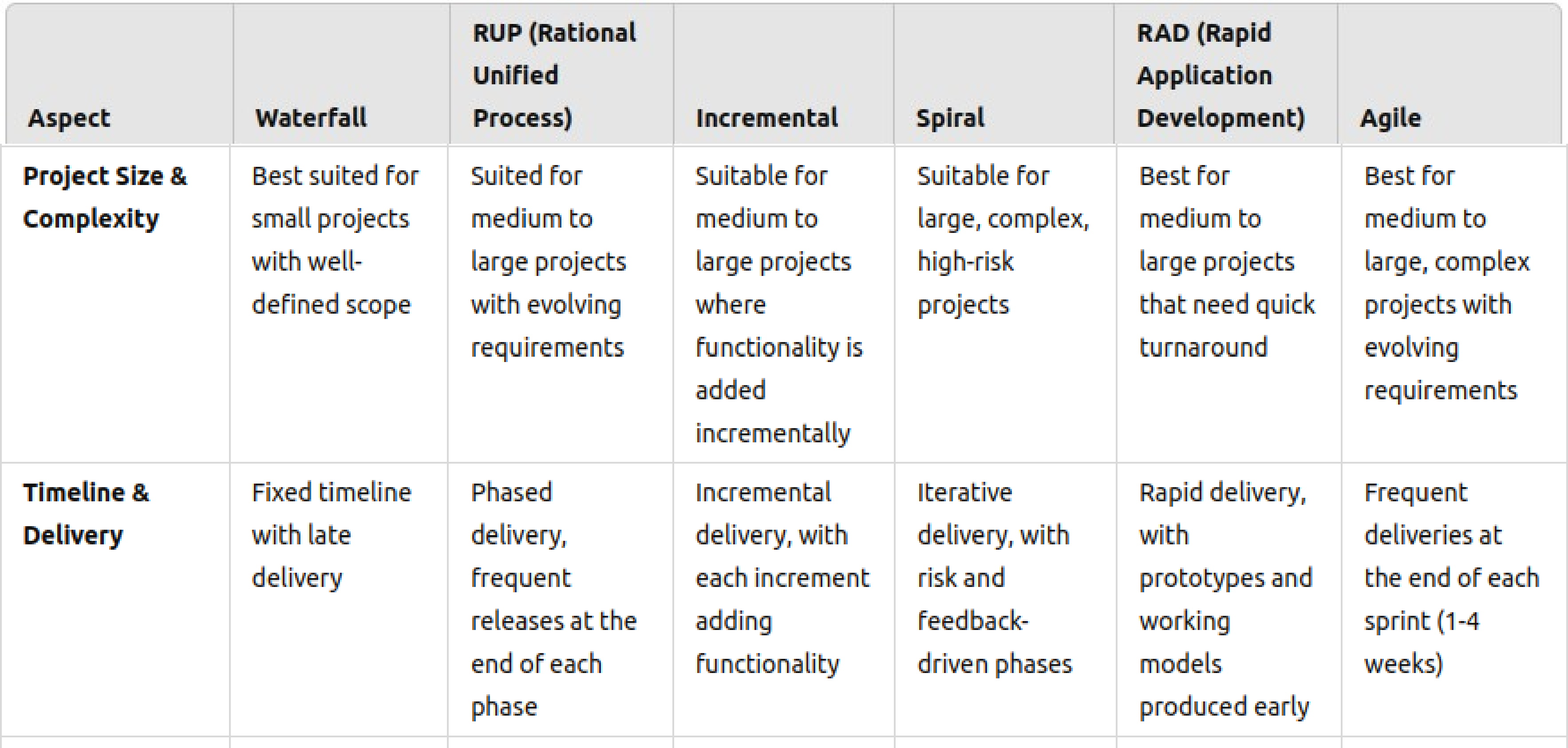
Aspect	Waterfall	RUP (Rational Unified Process)	Incremental	Spiral	RAD (Rapid Application Development)	Agile
Flexibility	Not flexible; changes hard to accommodate once a phase is complete	Moderate flexibility; can address changing requirements at phase level	Moderate flexibility between increments	High flexibility; accommodates changes at any stage	High flexibility; iterative feedback from users	Very high flexibility; changes can be made during any sprint
Risk Management	Low risk management; risks discovered late	Moderate; risks mitigated at each phase	Moderate risk management at each increment	Strong focus on risk management through each iteration	Low-risk handling; focused on speed and prototyping	Adaptive to risks; mitigated during each sprint
Customer Involvement	Low; customers involved mainly at the beginning and end	Moderate; involved at phase milestones	Moderate; feedback collected after each increment	High; customer feedback integrated into every cycle	High; frequent feedback during prototyping	High; customers involved throughout the process

SLC Model Comparison



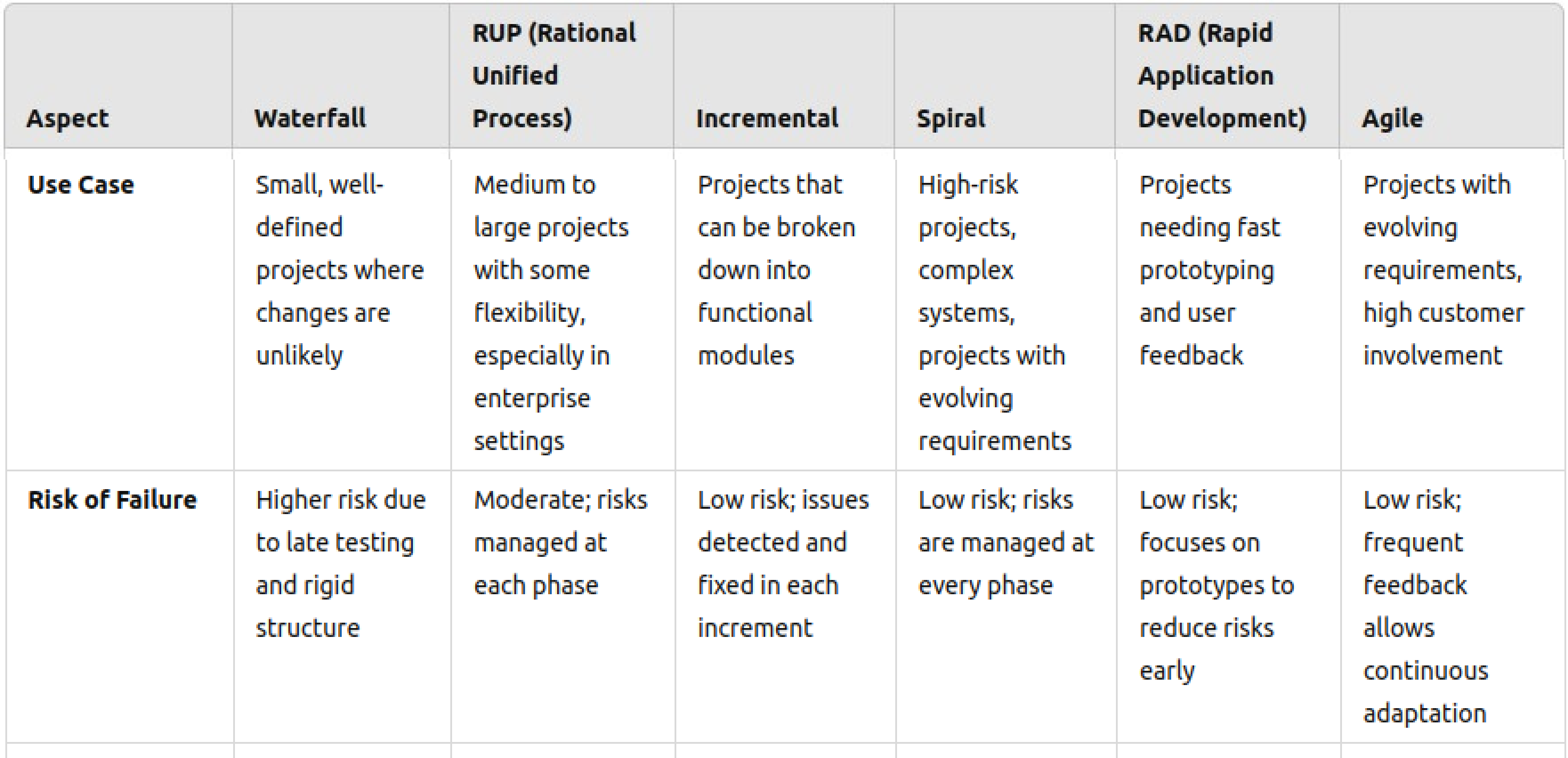
Aspect	Waterfall	RUP (Rational Unified Process)	Incremental	Spiral	RAD (Rapid Application Development)	Agile
Requirement Clarity	Requirements must be clear and fixed upfront	Flexible; requirements can evolve between phases	Initial requirements are defined, but some changes allowed incrementally	Requirements can evolve throughout the project	Requirements can change as per user feedback	Evolving and unclear requirements can be accommodated
Documentation	Heavy documentation required at each stage	Moderate documentation, updated in each phase	Documentation for each increment; less than Waterfall	Documentation in each phase for risk analysis	Light documentation; focuses more on user feedback and prototypes	Light documentation; focuses more on code and working product
Testing Approach	Testing happens after development is completed	Testing happens in each phase	Testing occurs after each increment	Testing in each cycle, as well as after risk analysis	Continuous testing throughout development	Continuous testing in every sprint

SLC Model Comparison



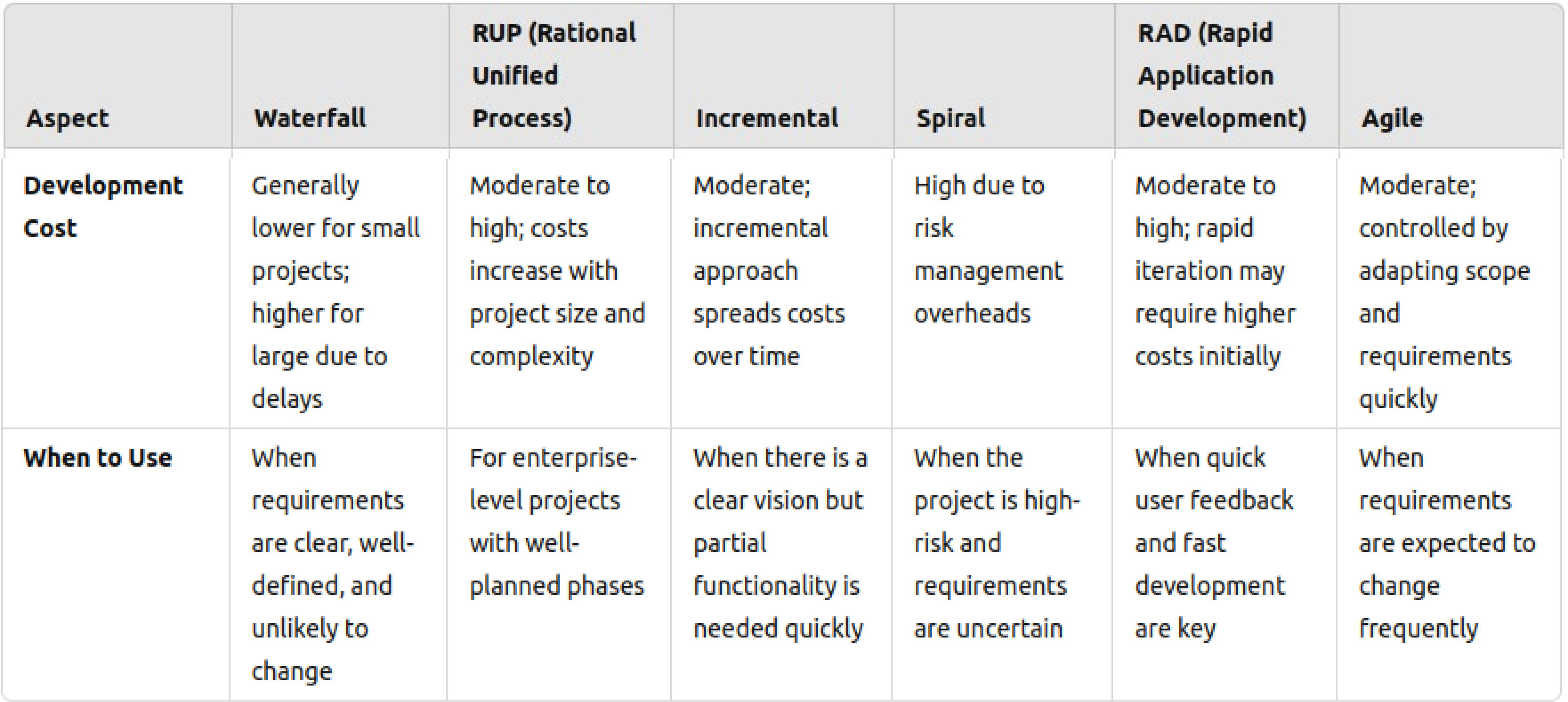
Aspect	Waterfall	RUP (Rational Unified Process)	Incremental	Spiral	RAD (Rapid Application Development)	Agile
Project Size & Complexity	Best suited for small projects with well-defined scope	Suited for medium to large projects with evolving requirements	Suitable for medium to large projects where functionality is added incrementally	Suitable for large, complex, high-risk projects	Best for medium to large projects that need quick turnaround	Best for medium to large, complex projects with evolving requirements
Timeline & Delivery	Fixed timeline with late delivery	Phased delivery, frequent releases at the end of each phase	Incremental delivery, with each increment adding functionality	Iterative delivery, with risk and feedback-driven phases	Rapid delivery, with prototypes and working models produced early	Frequent deliveries at the end of each sprint (1-4 weeks)

SLC Model Comparison



Aspect	Waterfall	RUP (Rational Unified Process)	Incremental	Spiral	RAD (Rapid Application Development)	Agile
Use Case	Small, well-defined projects where changes are unlikely	Medium to large projects with some flexibility, especially in enterprise settings	Projects that can be broken down into functional modules	High-risk projects, complex systems, projects with evolving requirements	Projects needing fast prototyping and user feedback	Projects with evolving requirements, high customer involvement
Risk of Failure	Higher risk due to late testing and rigid structure	Moderate; risks managed at each phase	Low risk; issues detected and fixed in each increment	Low risk; risks are managed at every phase	Low risk; focuses on prototypes to reduce risks early	Low risk; frequent feedback allows continuous adaptation

SLC Model Comparison



Aspect	Waterfall	RUP (Rational Unified Process)	Incremental	Spiral	RAD (Rapid Application Development)	Agile
Development Cost	Generally lower for small projects; higher for large due to delays	Moderate to high; costs increase with project size and complexity	Moderate; incremental approach spreads costs over time	High due to risk management overheads	Moderate to high; rapid iteration may require higher costs initially	Moderate; controlled by adapting scope and requirements quickly
When to Use	When requirements are clear, well-defined, and unlikely to change	For enterprise-level projects with well-planned phases	When there is a clear vision but partial functionality is needed quickly	When the project is high-risk and requirements are uncertain	When quick user feedback and fast development are key	When requirements are expected to change frequently

Selecting SLC Model

1-Project Requirements and Clarity

	W	S	I	RUP	RAD	Agile
Are the project requirements clearly defined and fixed?	Y	N	N	Y	N	N
Do you expect the project requirements to evolve frequently during development?	N	Y	N	N	Y	Y
How complex are the requirements, and are they subject to significant changes?	N	Y	N	N	N	Y

Selecting SLC Model

2-Customer and Stakeholder Involvement

	W	S	I	RUP	RAD	Agile
Will the customer or stakeholders be highly involved throughout the development process?	N	Y	N	N	Y	Y
Do stakeholders require frequent delivery of working software for review and feedback?	N	-	Y	N	Y	Y

Selecting SLC Model

3-Risk Management and Project Uncertainty

	W	S	I	RUP	RAD	Agile
Is the project highly risky, or does it involve significant technical uncertainties?	N	Y	N	Y	N	-
Do you need a development model that focuses on identifying and mitigating risks throughout the process?	N	Y	N	Y	N	-
Are the potential risks in this project mostly related to unclear requirements, technology, or stakeholders' expectations?	N	Y	N	N	-	Y

Selecting SLC Model

4-Flexibility and Changes

	W	S	I	RUP	RAD	Agile
How much flexibility do you need to make changes during the development process?	LF	HF	-	LF	HF	HF
Do you anticipate needing to adjust scope or features based on customer feedback during development?	N	Y	-	-	Y	Y
Is the development process expected to be highly adaptive and iterative?	N	Y	-	N	-	Y

Selecting SLC Model

5-Delivery Timeliness and Schedule

	W	S	I	RUP	RAD	Agile
Is there a strict deadline or a need for phased delivery?	Y	N	Y	Y	-	N
Do you require regular, small releases of working software?	N	-	Y	N	Y	Y
Are you comfortable with developing a prototype first and then refining it in later stages?	N	Y	-	N	Y	Y

Selecting SLC Model

6-Project Complexity and Size

	W	S	I	RUP	RAD	Agile
Is the project small and straightforward with minimal need for feedback or iteration?	Y	N	Y	N	Y	N
Is the project large, complex, or involves multiple teams?	N	Y	N	Y	N	Y

Selecting SLC Model

7-Testing and Quality Assurance

	W	S	I	RUP	RAD	Agile
Do you need to perform extensive testing at the end of the project (after development is complete)?	Y	N	N	Y	N	N
Is continuous testing and feedback integration important throughout the development lifecycle?	N	Y	Y	N	Y	Y

Selecting SLC Model

8-Budget and Resource Considerations

	W	S	I	RUP	RAD	Agile
Do you have a fixed budget with limited scope for changes?	Y	N		Y	N	N
Can the project’s budget and resource allocation adjust based on evolving requirements or risk mitigation?	N	Y		N		Y
Do you need to minimize development costs and time through rapid prototyping?	N		Y	N	Y	

Selecting SLC Model

9-Team Structure and Collaboration

	W	S	I	RUP	RAD	Agile
Is the project team small, and can they work in close collaboration with customers?	N	-	N	N	Y	Y
Do you need a structured, phased approach for large teams working on different parts of the project?	-	Y	N	Y	N	N
Does the development team have experience working in iterative, collaborative environments?	N	Y	N	N	Y	Y