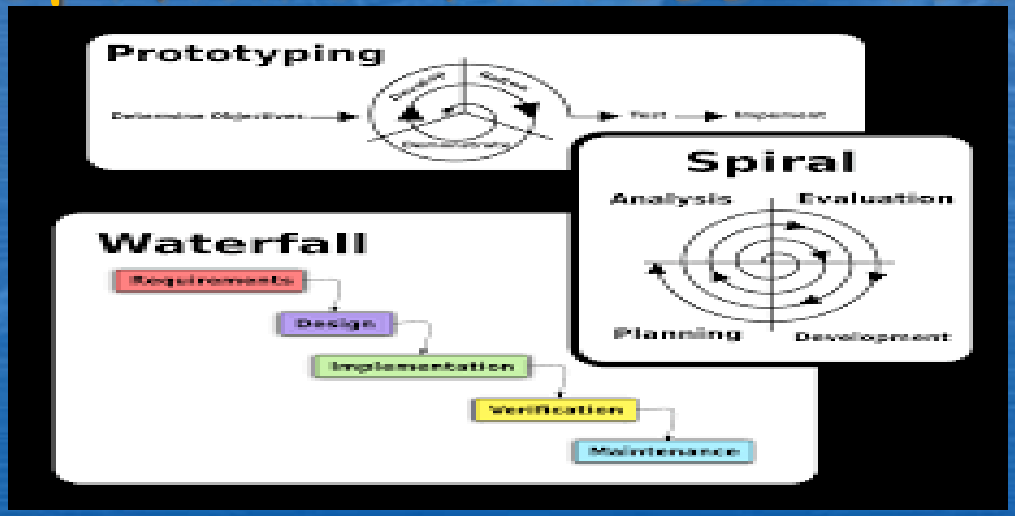
Software Process Models



Software Process Model

- Attempt to organize the software life cycle by
  - ✓ defining activities involved in software production
  - ✓ order of activities and their relationships
- Goals of a software process
  - ✓ standardization, predictability, productivity, high product quality, ability to plan time and budget requirements

Code & Fix

The earliest approach

- Write code
- Fix it to eliminate any errors that have been detected, to enhance existing functionality, or to add new features
- Source of difficulties and deficiencies
  - impossible to predict
  - difficult to manage

Why Models are needed?

- Symptoms of inadequacy: the software crisis
  - scheduled time and cost exceeded
  - user expectations not met
  - poor quality
- The size and economic value of software applications required appropriate "process models"

Goals of Process Model

Determine the order of stages involved in software development and evolution, and to establish the transition criteria for progressing from one stage to the next.

These include completion criteria for the current stage plus choice criteria and entry criteria for the next stage.

Thus a process model addresses the following software project questions:

What shall we do next?
How long shall we continue to do it?

SDLC Model

A framework that describes the activities performed at each stage of a software development project.

Stages of Software Development

Communication
Project initiation
Requirements gathering

Planning
Estimating
Scheduling
Tracking

Construction
Code
Test

Modelling
Analysis
Design

Deployment
Delivery
Support
Feedback

Miscommunication

Memo from CEO to Manager:
Today at 11 o'clock there will be a total eclipse of the sun. This is when the sun disappears behind the moon for two minutes. As this is something that cannot be seen every day, time will be allowed for employees to view the eclipse in the parking lot. Staff should meet in the lot at ten to eleven, when I will deliver a short speech introducing the eclipse, and giving some background information. Safety goggles will be made available at a small cost.

Memo from Manager to Department Head:
Today at ten to eleven, all staff should meet in the car park. This will be followed by a total eclipse of the sun, which will appear for two minutes. For a moderate cost, this will be made safe with goggles. The CEO will deliver a short speech beforehand to give us all some information. This is not something that can be seen every day.

Memo from Department Head to Floor Manager:
The CEO will today deliver a short speech to make the sun disappear for two minutes in the form of an eclipse. This is something that cannot be seen every day, so staff will meet in the car park at ten or eleven. This will be safe, if you pay a moderate cost.

Memo from Floor Manager to Supervisor:
Ten or eleven staff are to go to the car park, where the CEO will eclipse the sun for two minutes. This doesn't happen every day. It will be safe, and as usual it will cost you.

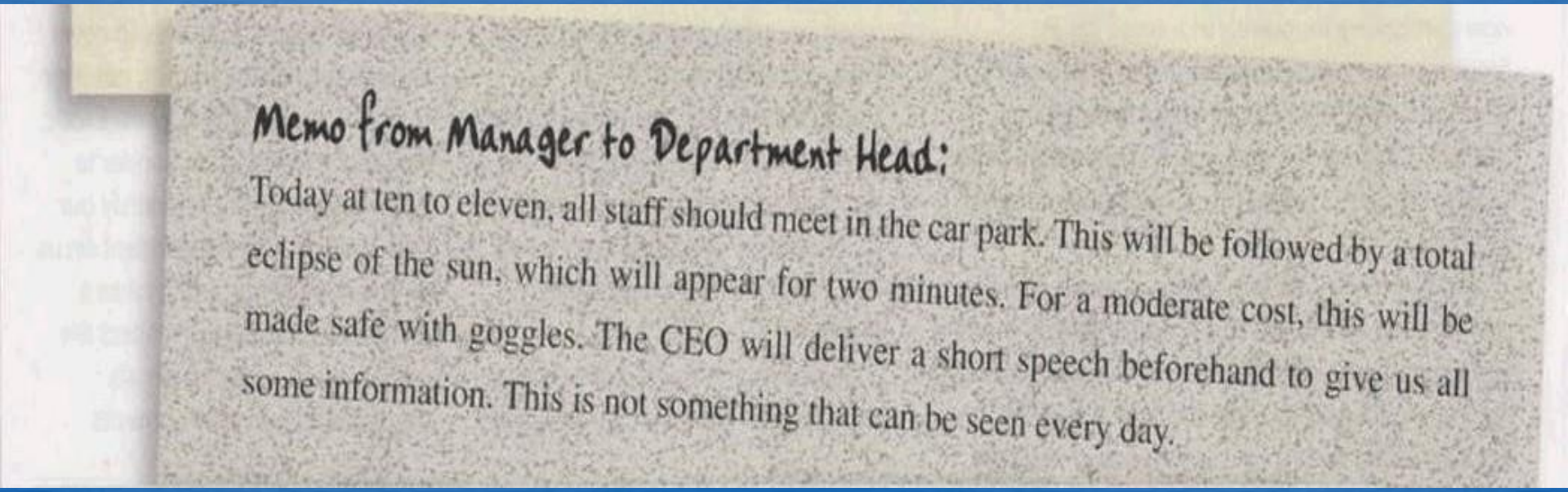
Memo from Supervisor to staff:
Some staff will go to the car park today to see the CEO disappear. It is a pity this doesn't happen everyday.

THELASTPAGE

Miscommunication

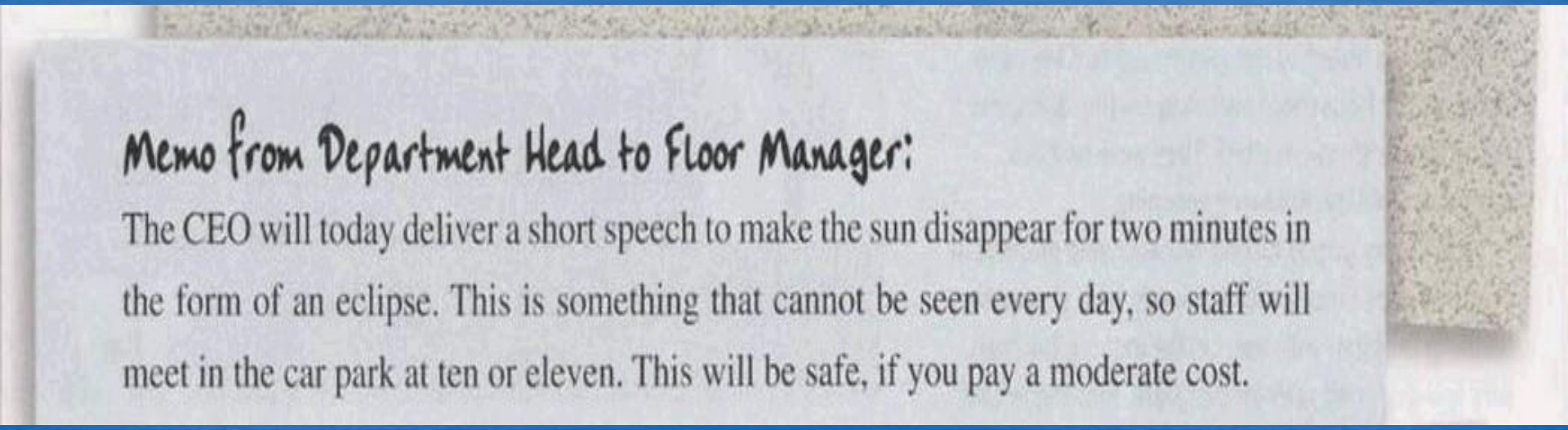
Memo from CEO to Manager:

Today at 11 o'clock there will be a total eclipse of the sun. This is when the sun disappears behind the moon for two minutes. As this is something that cannot be seen every day, time will be allowed for employees to view the eclipse in the parking lot. Staff should meet in the lot at ten to eleven, when I will deliver a short speech introducing the eclipse, and giving some background information. Safety goggles will be made available at a small cost.



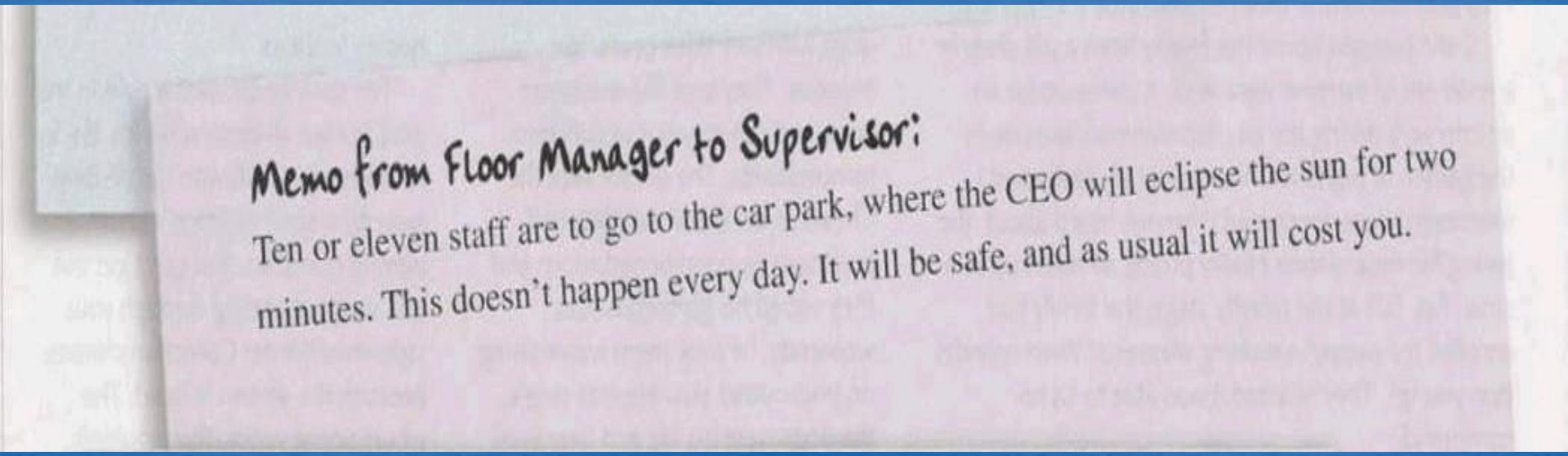
Memo from Manager to Department Head:

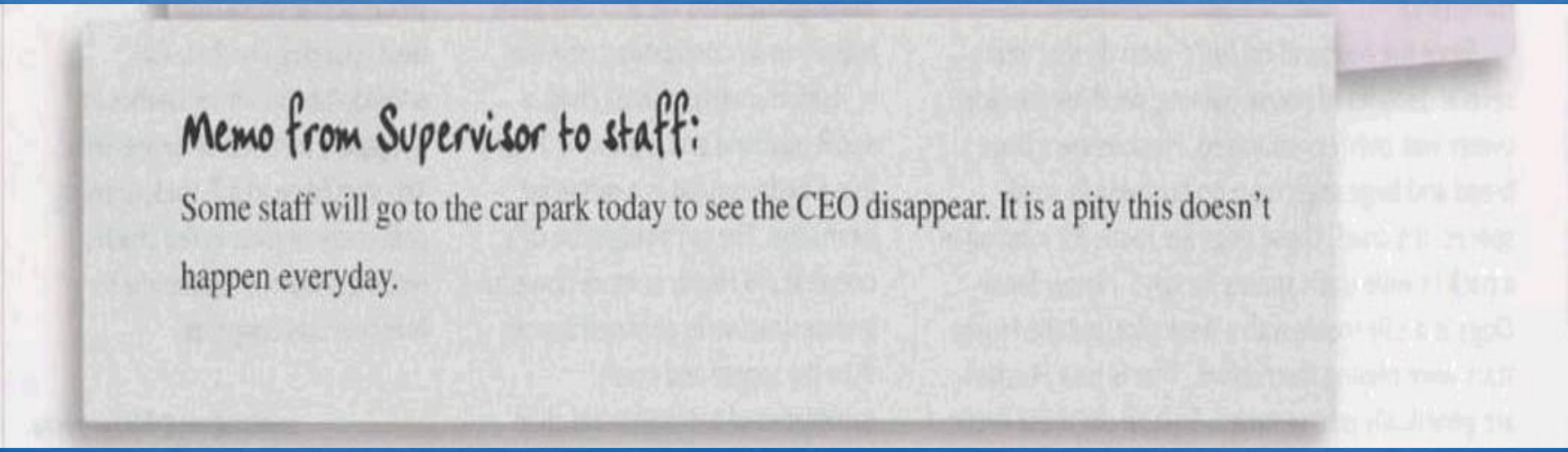
Today at ten to eleven, all staff should meet in the car park. This will be followed by a total eclipse of the sun, which will appear for two minutes. For a moderate cost, this will be made safe with goggles. The CEO will deliver a short speech beforehand to give us all some information. This is not something that can be seen every day.



Memo from Department Head to Floor Manager:

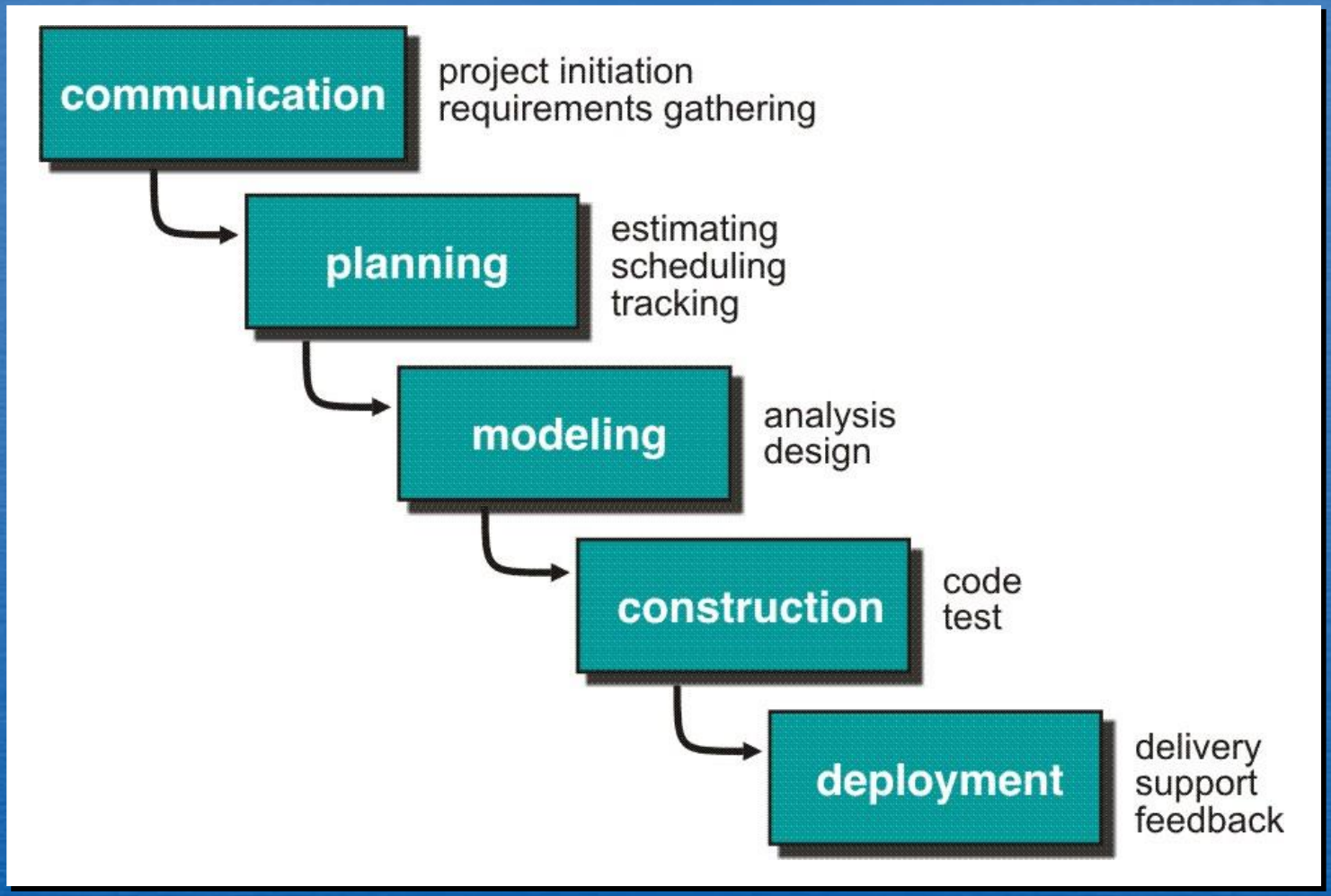
The CEO will today deliver a short speech to make the sun disappear for two minutes in the form of an eclipse. This is something that cannot be seen every day, so staff will meet in the car park at ten or eleven. This will be safe, if you pay a moderate cost.


Memo from Floor Manager to Supervisor:
Ten or eleven staff are to go to the car park, where the CEO will eclipse the sun for two minutes. This doesn't happen every day. It will be safe, and as usual it will cost you.


Memo from Supervisor to staff:
Some staff will go to the car park today to see the CEO disappear. It is a pity this doesn't happen everyday.


Just for Laugh
Wow, a different error message... Finally some progress!

The Waterfall Model



Waterfall Strengths

- Easy to understand, easy to use
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)
- Works well when quality is more important than cost or schedule

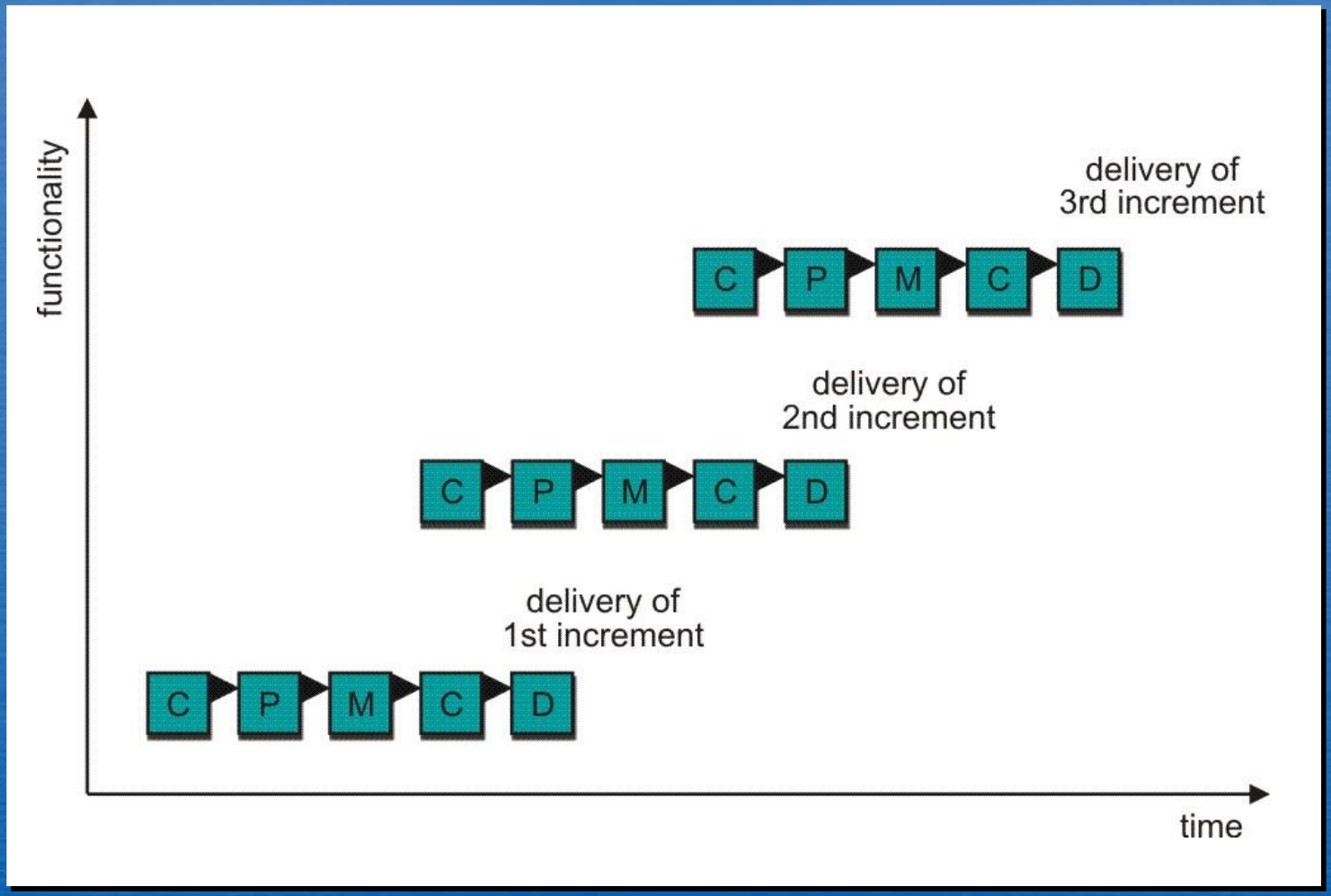
Waterfall Deficiencies

- All requirements must be known upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- Does not reflect problem-solving nature of software development – iterations of phases
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (it may be too late)

When to use the Waterfall Model?

- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform.
  - High risk for new systems because of specification and design problems.
  - Low risk for well-understood developments using familiar technology.

Incremental Model



Incremental Model Strengths

- Develop high-risk or major functions first

- Each increment delivers an operational product
- Customer can respond to each increment

- Lowers initial delivery cost
- Initial product delivery is faster

- Customers get important functionality early
- Risk of changing requirements is reduced

Incremental Model Weaknesses

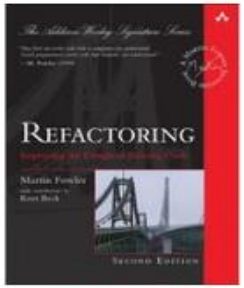
- Requires good planning and design
- Requires early definition of a complete and fully functional system to allow for the definition of increments
- Well-defined module interfaces are required (some will be developed long before others)
- Total cost of the complete system is not lower

When to use the Incremental Model?

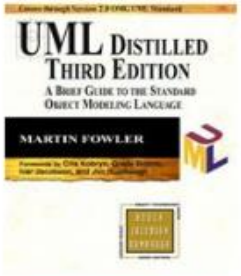
- Risk, funding, schedule, program complexity, or need for early realization of benefits.
- Most of the requirements are known up-front but are expected to evolve over time
- A need to get basic functionality to the market early
- On projects which have lengthy development schedules
- On a project with new technology


When to use iterative development?
You should use iterative development only on projects that you want to succeed.
Martin Fowler
Author and programmer

Books by Martin Fowler

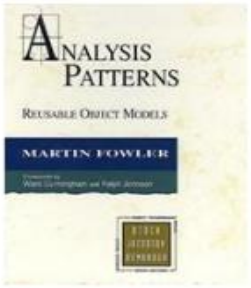


Refactoring: Improving the
Martin Fowler
$7.09 - $52.41

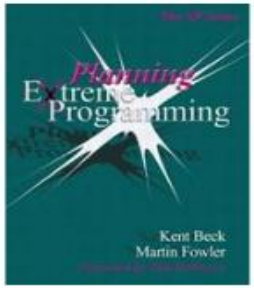


UML Distilled: A Brief Guide to the
Martin Fowler
$3.59 - $5.19

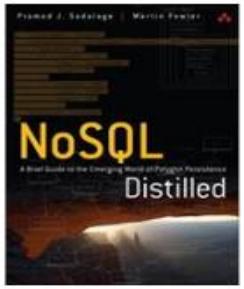
Patterns of Enterprise
Martin Fowler
$46.59 - $65.42



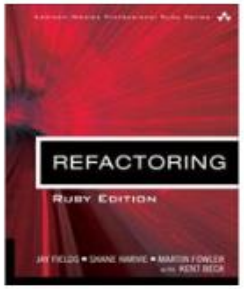
Analysis Patterns: Reusable Object
Martin Fowler
$9.19 - $9.69



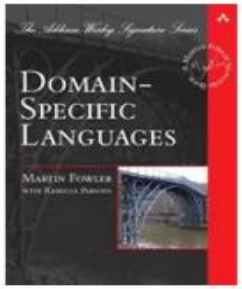
Planning Extreme Programming
Martin Fowler
$4.39 - $4.59



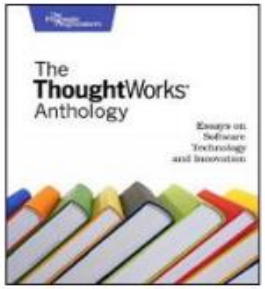
□□NoSQL□15□□□ Distilled□□□□



Refactoring: Ruby Edition: Ruby



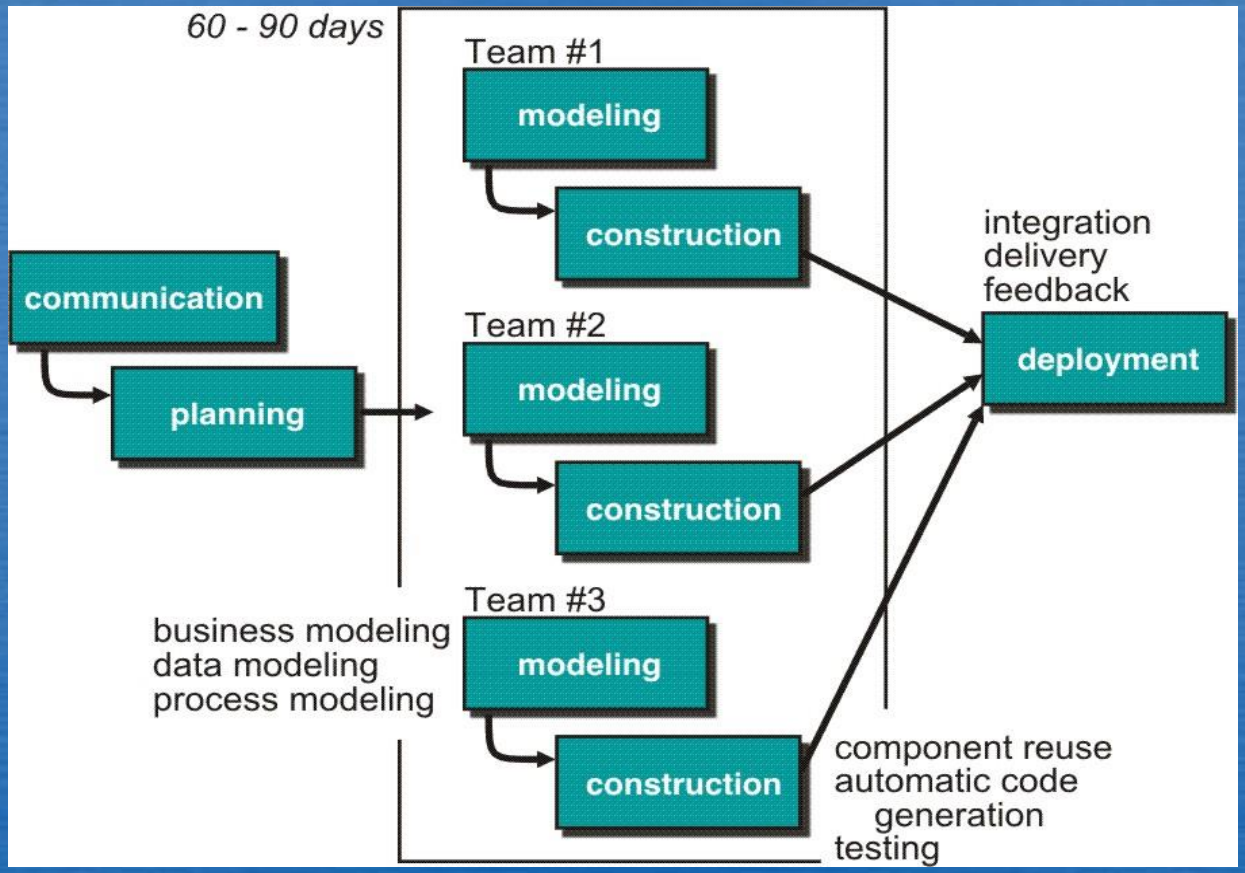
Domain-Specific Languages



The ThoughtWorks Anthology: Essays

Object Oriented Reengineering

RAD Model



Rapid Application Development – RAD

RAD Strengths

- **Reduced cycle time** and improved productivity with fewer people means lower costs
- **Time-box** approach mitigates cost and schedule risk
- **Customer involved throughout** the complete cycle minimizes risk of not achieving customer satisfaction and business needs

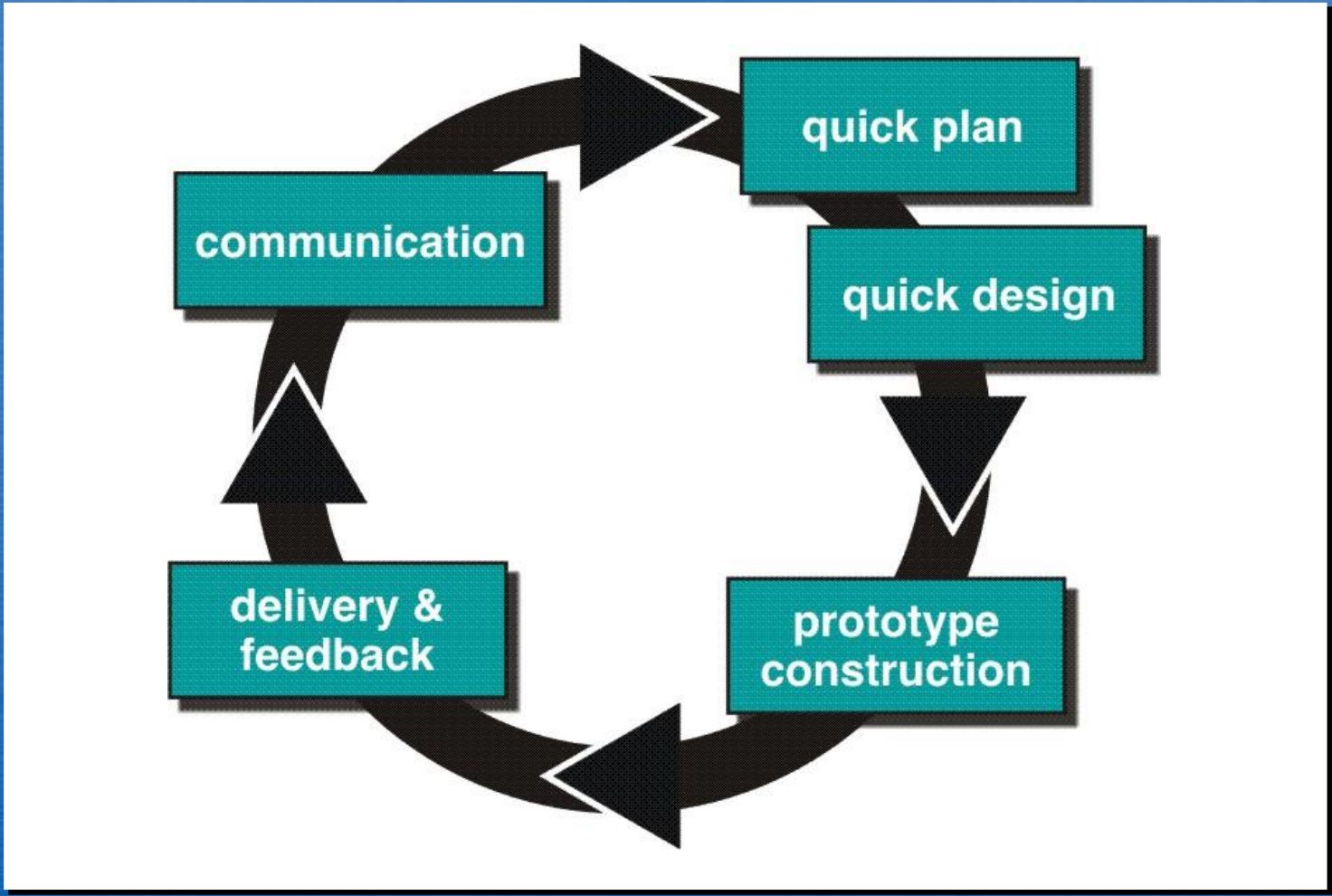
RAD Weaknesses

- Accelerated development process must give quick responses to the user
- Risk of never achieving closure
- Hard to use with legacy systems
- Requires a system that can be modularized
- Developers and customers must be committed to rapid-fire activities in an abbreviated time frame.

When to use RAD?

- Reasonably well-known requirements
- User involved throughout the life cycle
- Project can be time-boxed
- Functionality delivered in increments
- Low technical risks
- System can be modularized

Evolutionary Models: Prototyping



Types of Prototyping

Throwaway Prototyping Approach

Evolutionary Prototyping Approach

Prototyping Strengths

- Reduced time and cost
  - Can improve the quality of requirements and specifications provided to developers
    - Early determination of what the user really wants can result in faster and less expensive software
- Improved/increased user involvement
  - User can see and interact with the prototype, allowing them to provide better/more complete feedback and specs
  - Misunderstandings / miscommunications revealed
  - Final product more likely to satisfy their desired look / feel / performance

Disadvantages of prototyping

- Insufficient analysis
- Focus on limited prototype can distract developers from analyzing complete project
- Conversion of limited prototypes into poorly engineered final projects that are hard to maintain
- May not be noticed if developers too focused on building prototype as a model

Disadvantages of prototyping

- User confusion of prototype and finished system
  - Users can think that a prototype (intended to be thrown away) is actually a final system that needs to be polished.
  - Unaware of the scope of programming needed to give prototype robust functionality
  - Users can become attached to features included in prototype for consideration

Disadvantages of prototyping

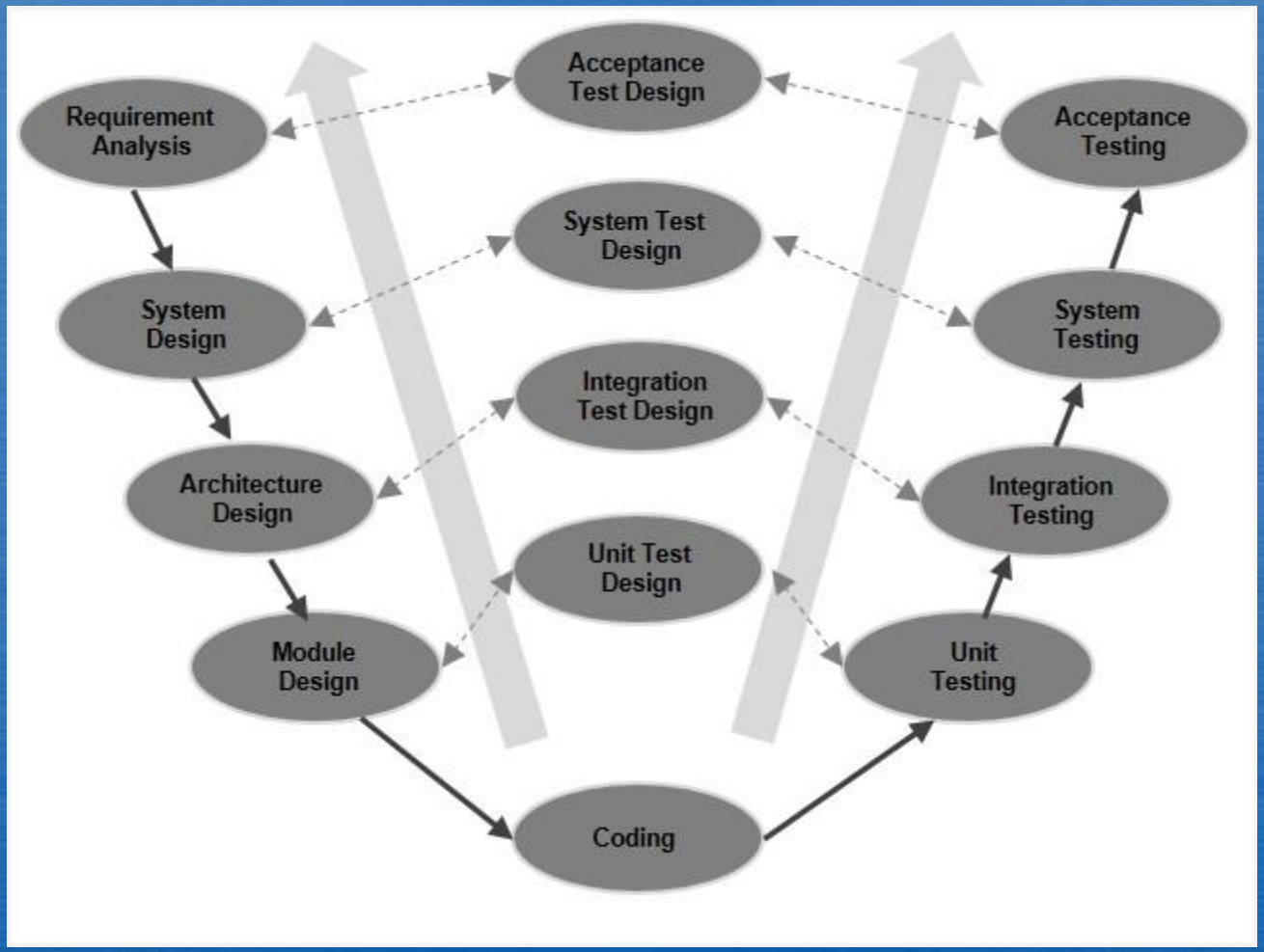
- Developer attachment to prototype
  - If spend a great deal of time/effort to produce, may become attached
  - Might try to attempt to convert a limited prototype into a final system
  - Bad if the prototype does not have an appropriate underlying architecture

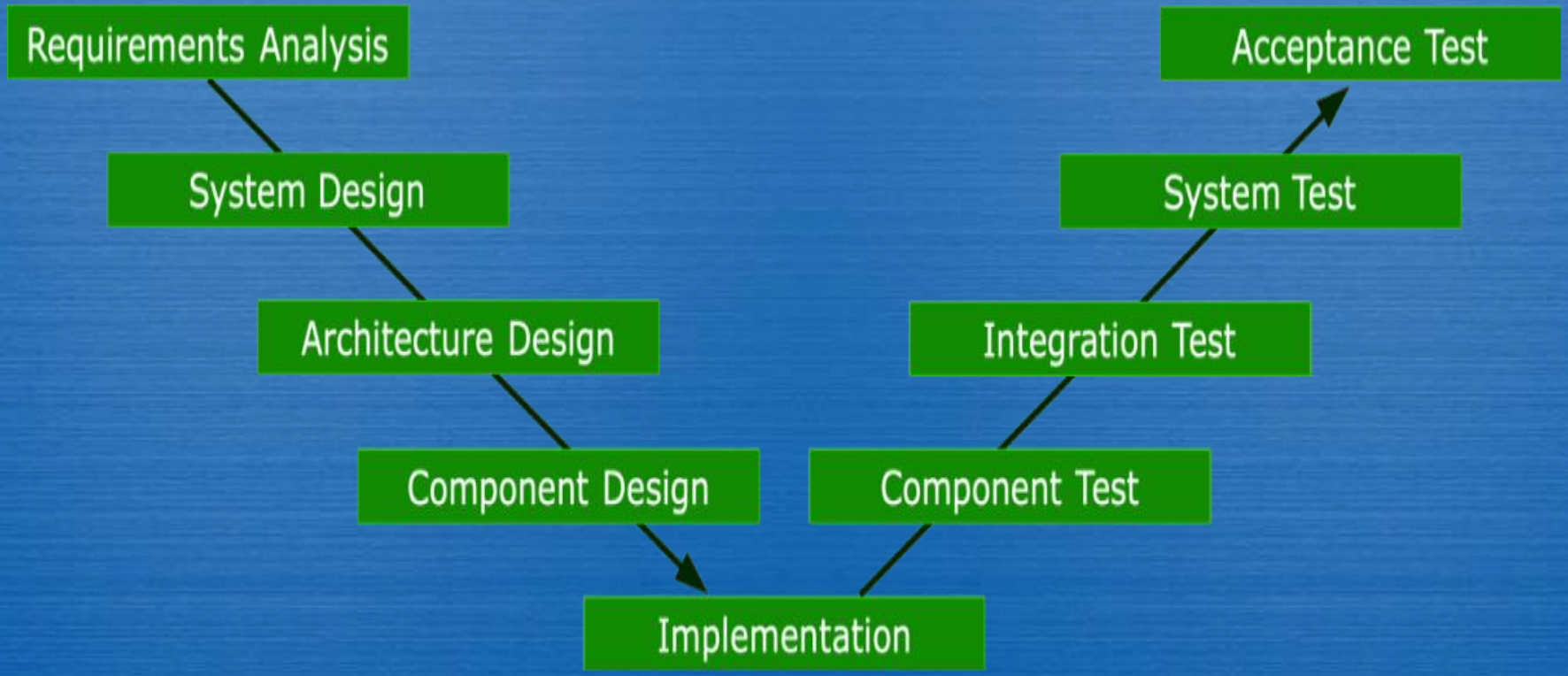
Disadvantages of prototyping

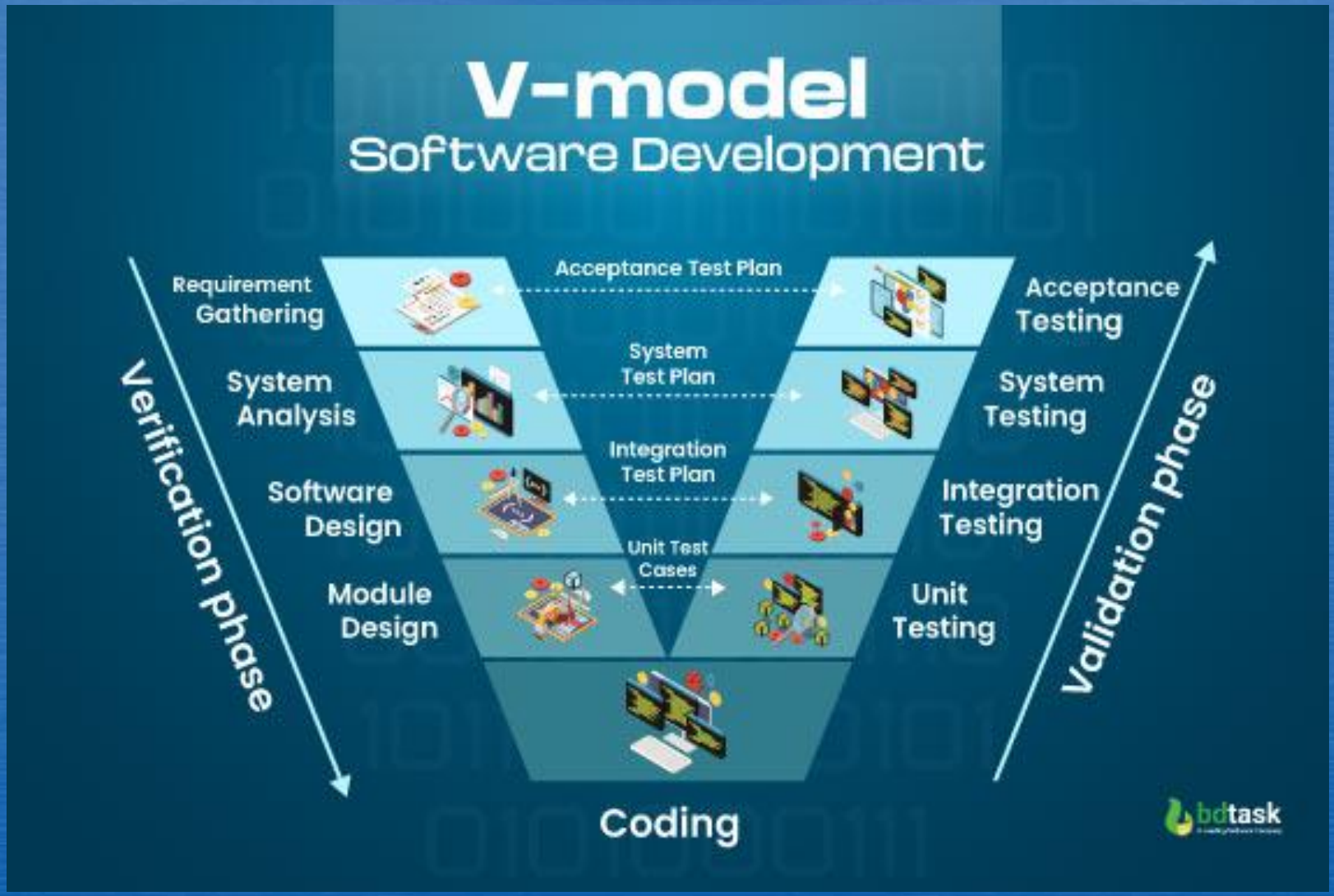
- Excessive development time of the prototype
  - Prototyping supposed to be done quickly.
  - If developers lose sight of this, can try to build a prototype that is too complex
  - For throw away prototypes, the benefits realized from the prototype (precise requirements) may not offset the time spent in developing the prototype – expected productivity reduced
  - Users can be stuck in debates over prototype details and hold up development process

The V Model

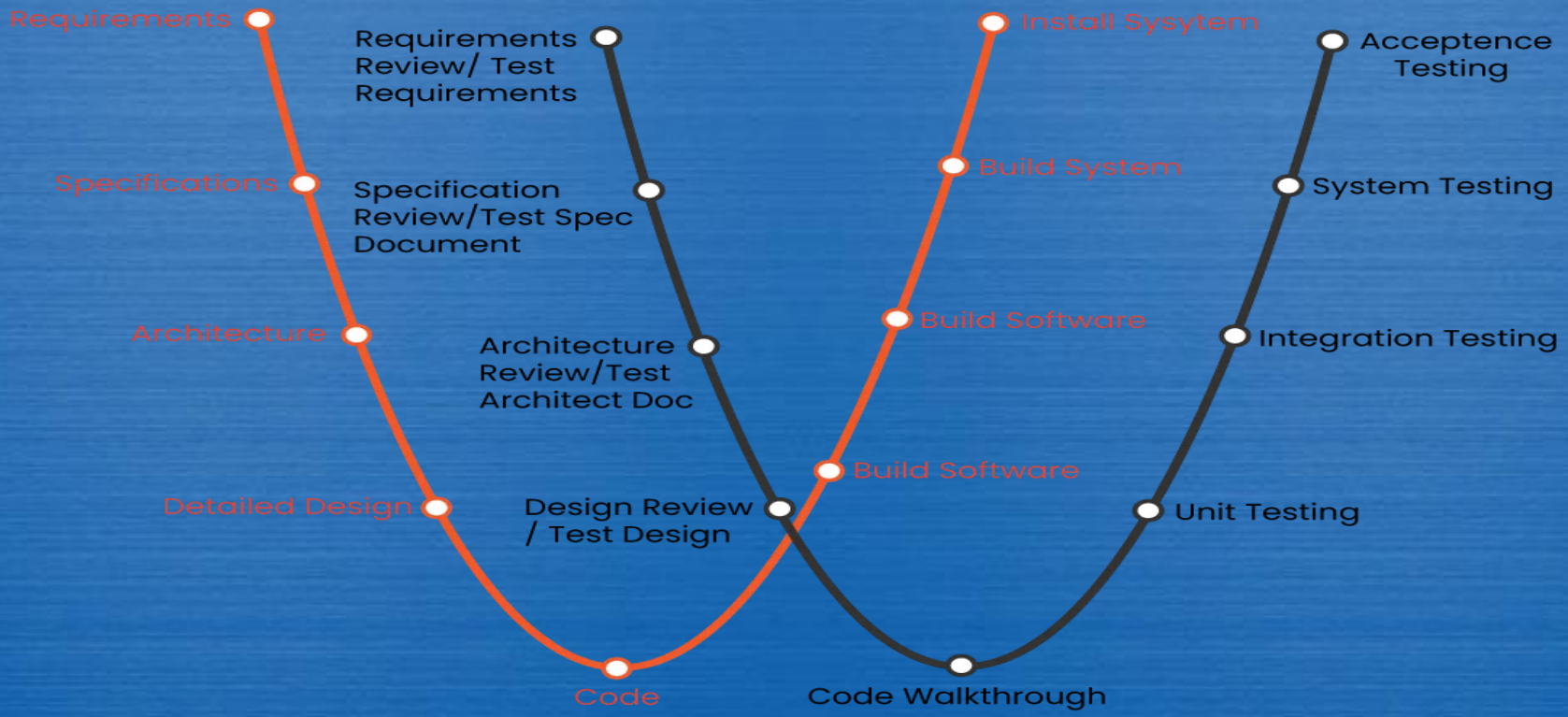
- It is an **extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage.**
- **A variant of the Waterfall that emphasizes the verification and validation of the product.**
- This means that for every single phase in the development cycle, there is a directly associated testing phase.
- Testing of the product is planned & designed in parallel with a corresponding phase of development
- This is a highly-disciplined model and the next phase starts only after completion of the previous phase.


Requirement Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding


Requirements Analysis
System Design
Architecture Design
Component Design
Implementation
Component Test
Integration Test
System Test
Acceptance Test


V-model
Software Development
Requirement Gathering
Acceptance Test Plan
Acceptance Testing
System Analysis
System Test Plan
System Testing
Software Design
Integration Test Plan
Integration Testing
Module Design
Unit Test Cases
Unit Testing
Coding
Verification phase
Validation phase
bdtask

W – Model


Requirements
Requirements Review/ Test Requirements
Specifications
Specification Review/Test Spec Document
Architecture
Architecture Review/Test Architect Doc
Detailed Design
Design Review / Test Design
Code
Code Walkthrough
Build Software
Unit Testing
Build System
System Testing
Build Software
Integration Testing
Install Sysytem
Acceptence Testing

V-Model Strengths

- Emphasize planning for verification and validation of the product in early stages of product development
- Each deliverable must be testable
- Project management can track progress by milestones
- Easy to use

V-Model Weaknesses

- Does not easily handle concurrent events
- Does not handle iterations or phases
- Does not easily handle dynamic changes in requirements
- Does not contain risk analysis activities

When to use the V-Model?

- **Excellent choice for systems requiring high reliability – hospital patient control applications.**
- All requirements are known up-front
- When it can be modified to handle changing requirements beyond analysis phase
- Solution and technology are known

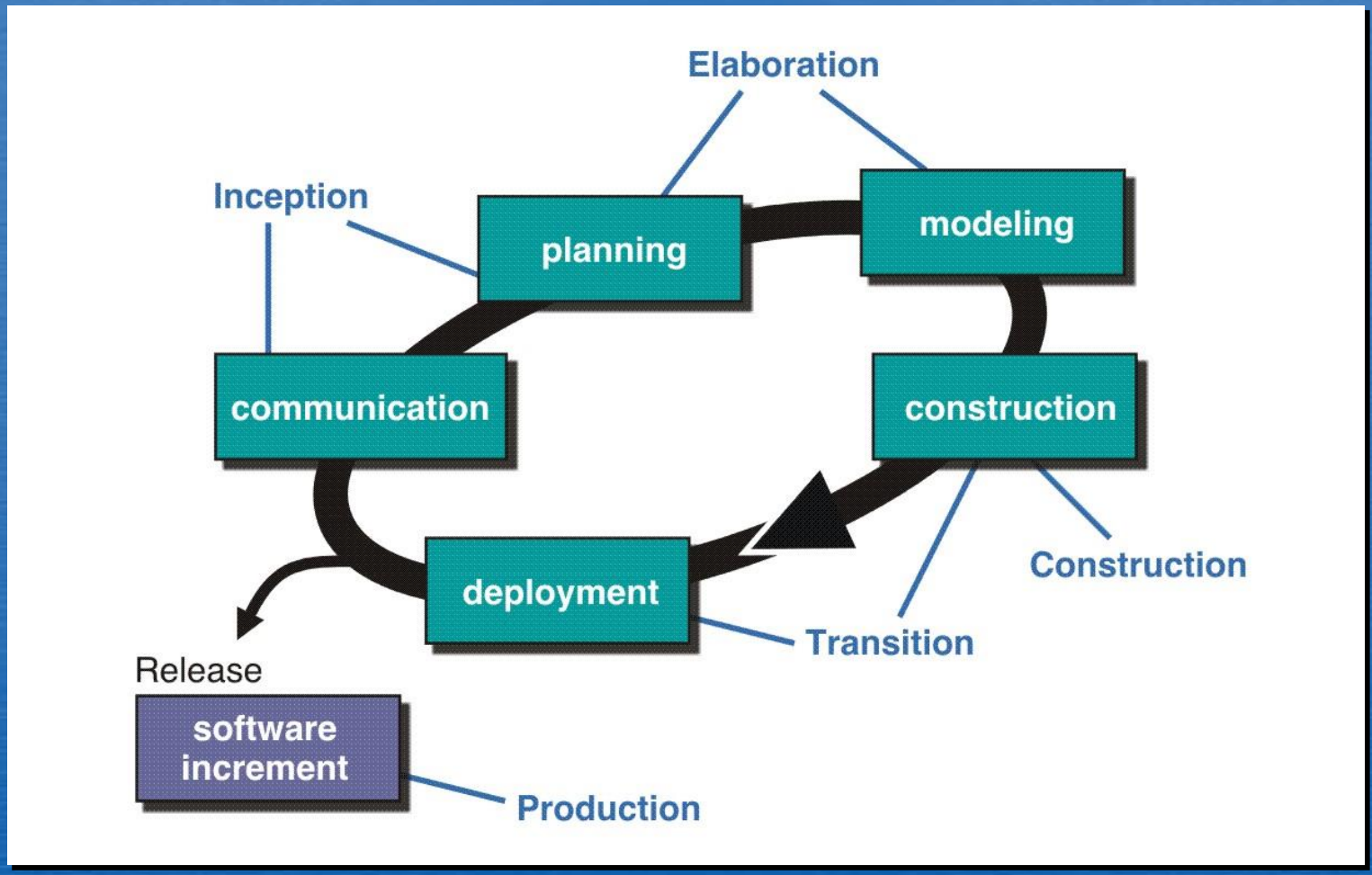
Unified Process Model

A software process that is:

- use-case driven
- architecture-centric
- iterative and incremental

Closely aligned with the Unified Modeling Language (UML)

The Unified Process (UP)



UP Work Products

Inception Phase

- Vision document
- Initial use-case model
- Initial project glossary
- Initial business case
- Initial risk assessment
- Project plan phases and iterations
- Business model if necessary
- One or more prototypes

Elaboration Phase

- Use-case model
- Supplementary requirements including non-functional
- Analysis model
- Software architecture description
- Executable architectural prototype
- Preliminary design model
- Revised risk list
- Project plan including
  - iteration plan
  - adapted workflows
  - milestones
  - technical work products
- Preliminary user manual

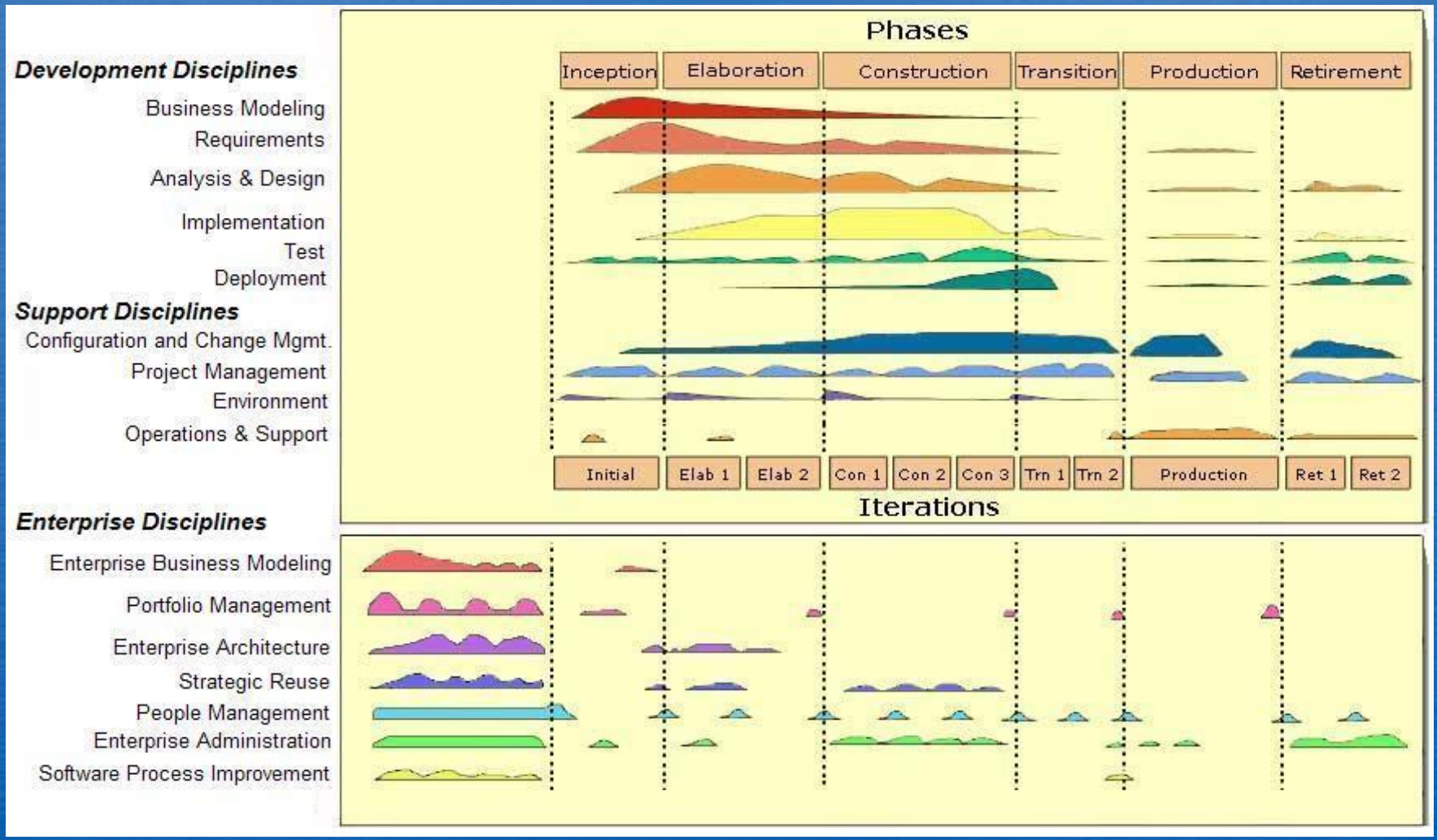
Construction Phase

- Design model
- Software components
- Integrated software increment
- Test plan and procedure
- Test cases
- Support documentation
  - user manuals
  - installation manuals
  - description of current increment

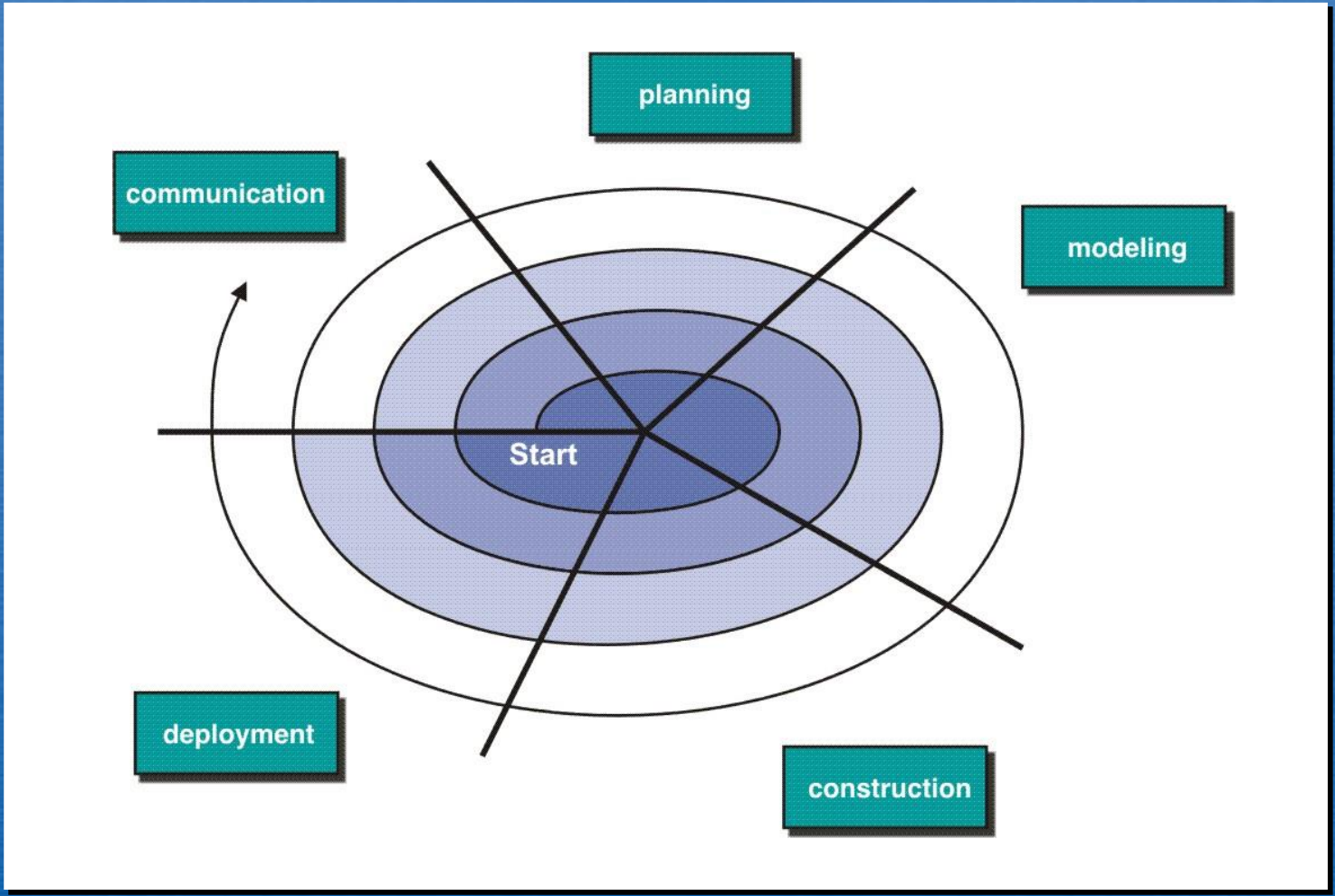
Transition Phase

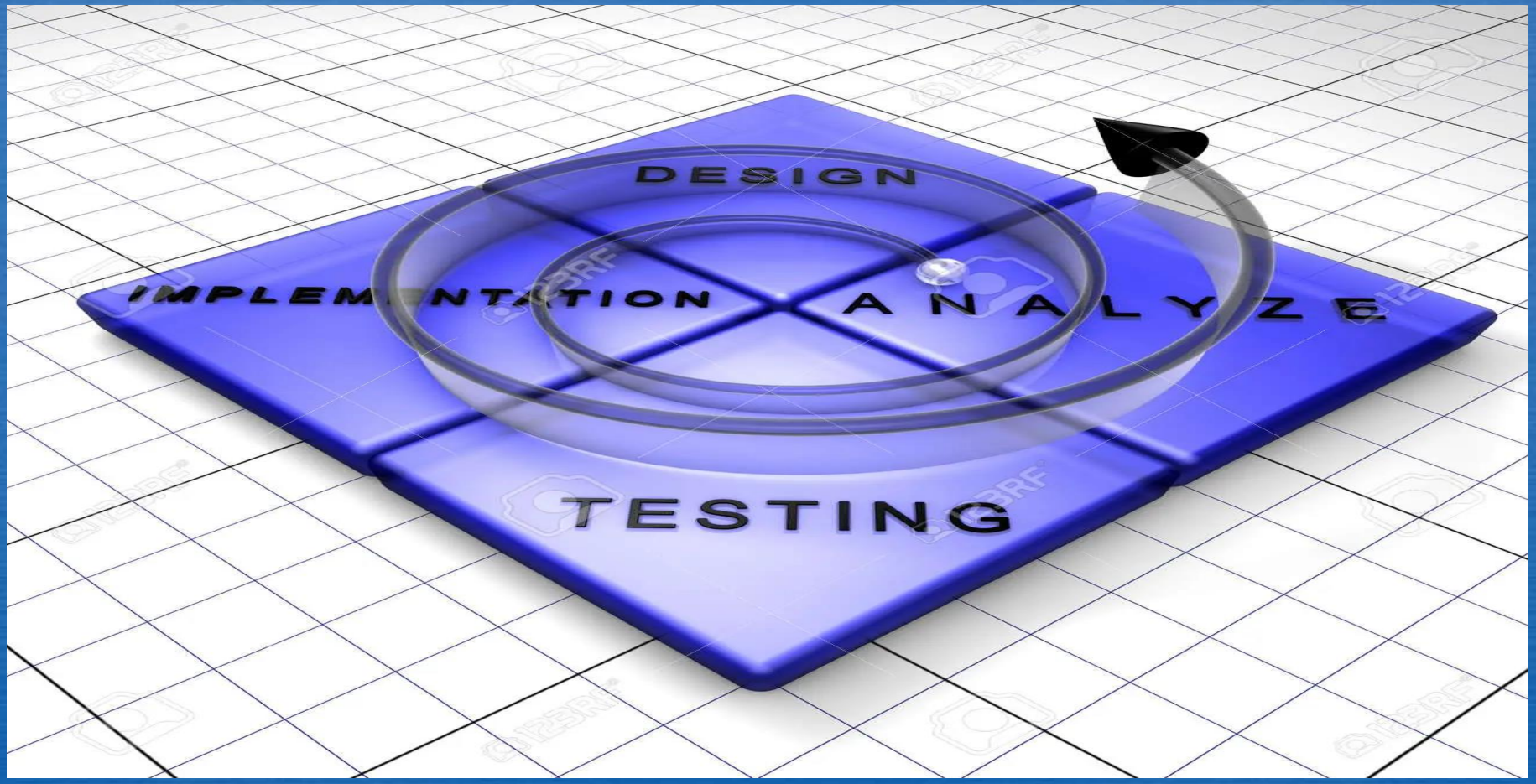
- Delivered software increment
- Beta test reports
- General user feedback

Lifecycle for Enterprise Unified Process



Evolutionary Models: The Spiral




DESIGN
ANALYZE
TESTING

Spiral Model

- Simplified form
  - Waterfall model plus risk analysis
- Precede each phase by
  - Alternatives
  - Risk analysis
- Follow each phase by
  - Evaluation
  - Planning of next phase

Analysis of Spiral Model

- Strengths
  - Easy to judge how much to test
  - No distinction between development, maintenance
- Weaknesses
  - For large-scale software only
  - For internal (in-house) software only

Conclusion

- Different life-cycle models
- Each with own strengths
- Each with own weaknesses
- Criteria for deciding on a model include
  - The organization
  - Its management
  - Skills of the employees
  - The nature of the product
- Best suggestion
  - “Mix-and-match” life-cycle model