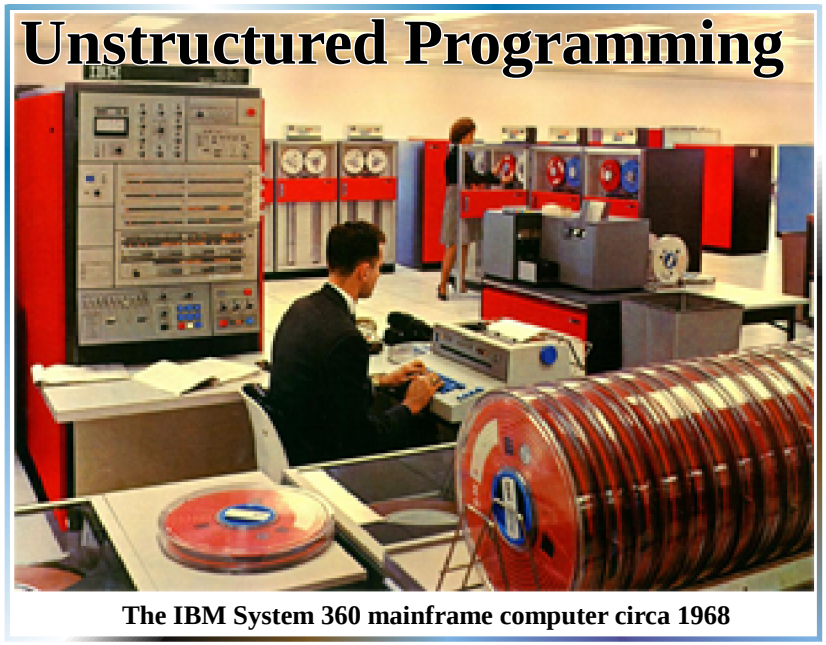
Object Oriented Software Engineering (OOSE*)

* Not the modeling tool developed by Ivar Jacobson of Objectory, Inc. in the early 1990s

Programming Paradigms

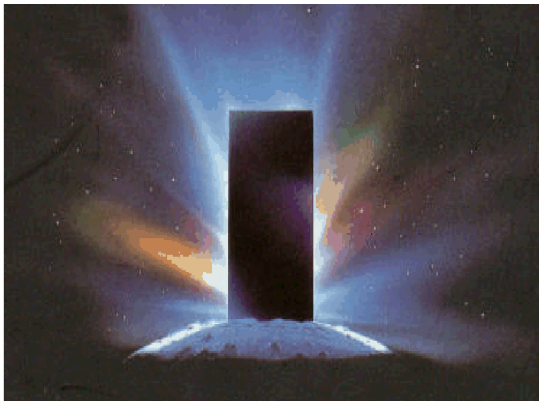
- Unstructured programming
- Procedural programming
- Modular programming
- Object oriented programming

On the way to
Object Oriented Software Engineering
Way Back When
(Like prior to 1975)



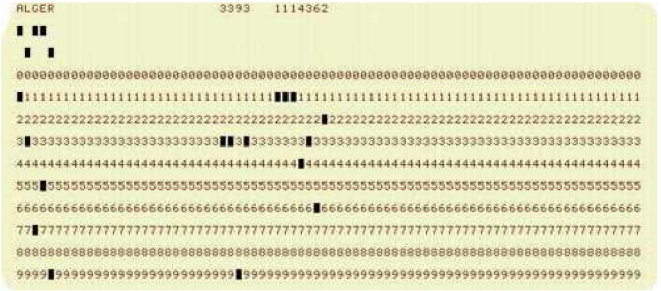
The IBM System 360 mainframe computer circa 1968

A single monolithic block of code.



The Monolith "computer" in *2001: A Space Odyssey* circa 1968

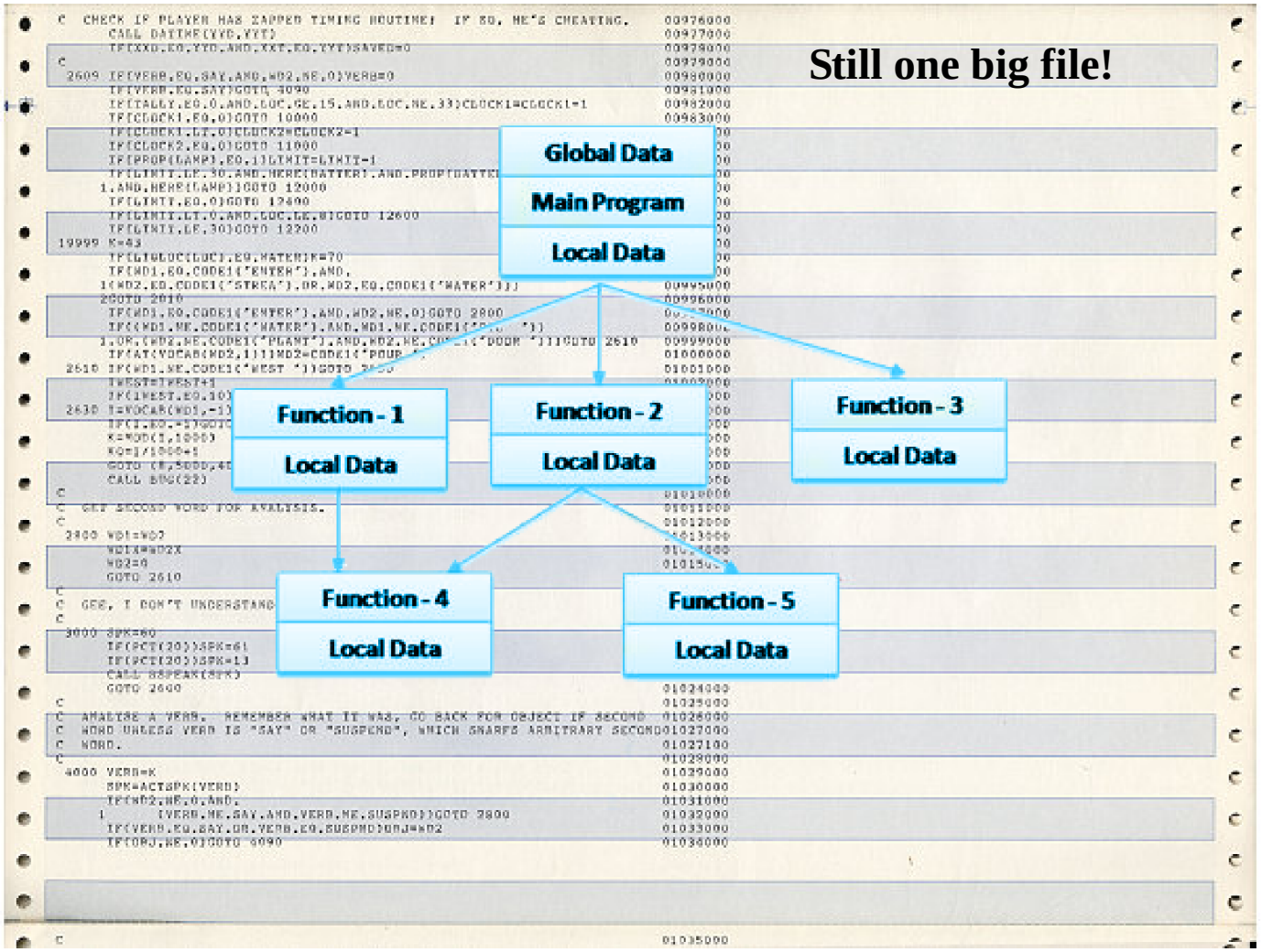
Actually a single monolithic stack of IBM punch cards



On the way to
Object Oriented Software Engineering

Procedural Programming

(Mostly in the 1970s)

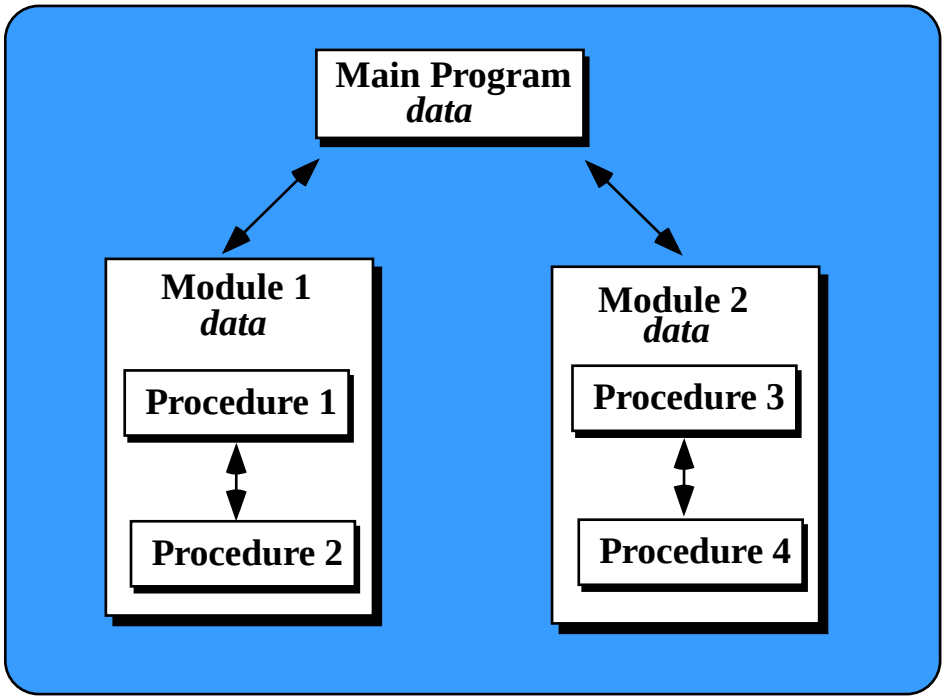


Code is organized into sub-routines, functions, methods, procedures, etc. which become the building blocks of a program used by the main routine.

On the way to
Object Oriented Software Engineering

Modular programming using the Structured or Classical Paradigm
(1975-1985)

- **1. Structured Systems Analysis**
- **2. Structured Programming**
- **3. Structured Testing**



Code is organized into a number of files or sets of files each providing some clearly defined services for the main program.

Some really good programmers could make a modular program look and act like an Object-Oriented program.

On the way to
Object Oriented Software Engineering

Structured or Classical Paradigm
(1975-1985)

1. Structured Systems Analysis*

The three most important techniques used were:

- **Logical Data Modeling**
- **Data Flow Modeling**
- **Entity Behavior Modeling**

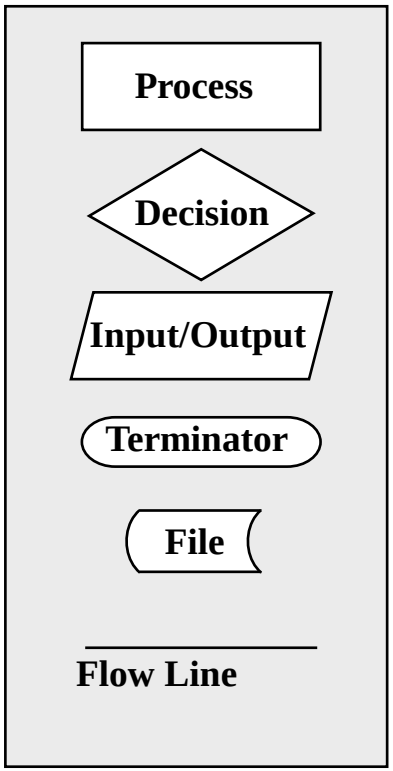
***Originally created by Central Computer and Communications Agency in the United Kingdom.
It was called Structures Systems Analysis and Design Method (SSADM)**

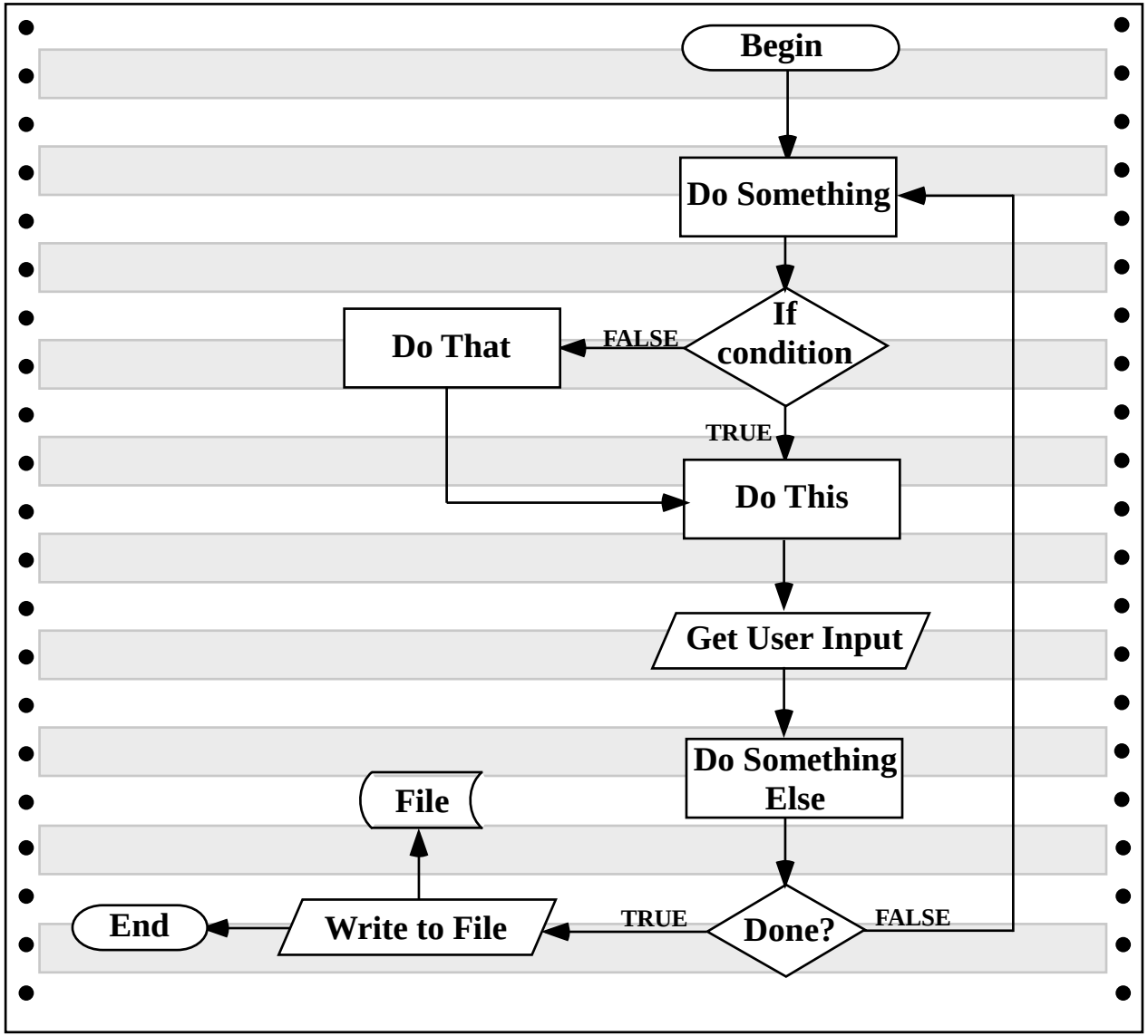
On the way to

Object Oriented Software Engineering

Data Flow Diagrams using Flow Charts

Symbols:





On the way to
Object Oriented Software Engineering

Structured or Classical Paradigm
(1975-1985)

- **2. Structured Programming**

  Following some standard programming in the design.
  Organize code into logical and related units (modules)

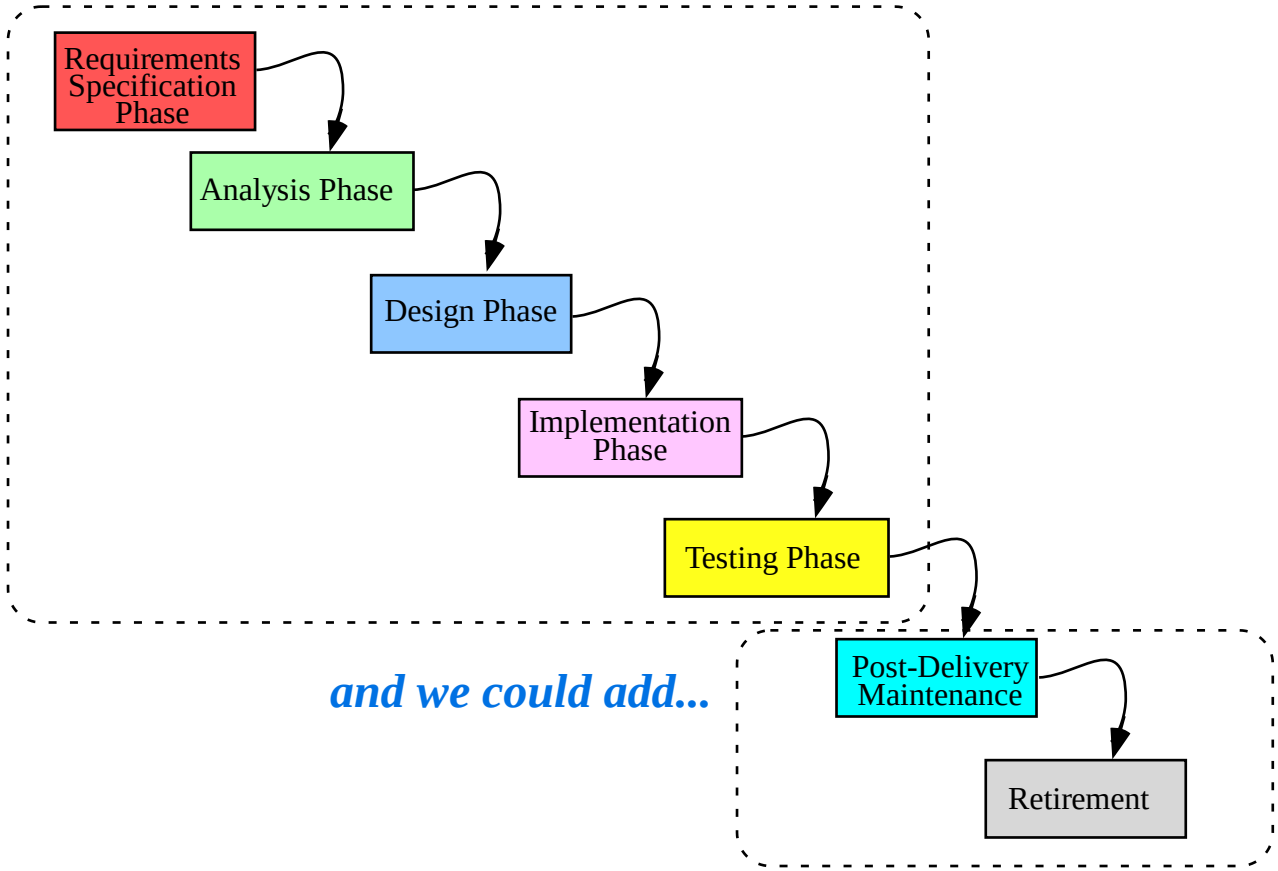
- **3. Structured Testing**

  Do Unit, Integration, and Acceptance Testing following a test plan.
  Use Clear Box testing in every phase.

On the way to
Object Oriented Software Engineering

Phases in Classical Software Development

a.k.a. The Waterfall Method



Object Oriented Software Engineering

Object Oriented Paradigm

Object-oriented programming (OOP) is a programming paradigm using “objects” (data structures consisting of data fields and methods together with their interactions) to design applications and computer programs.

Programming techniques may include features such as data abstraction, encapsulation, messaging, modularity, polymorphism, and inheritance.

Do you really understand what all this means?

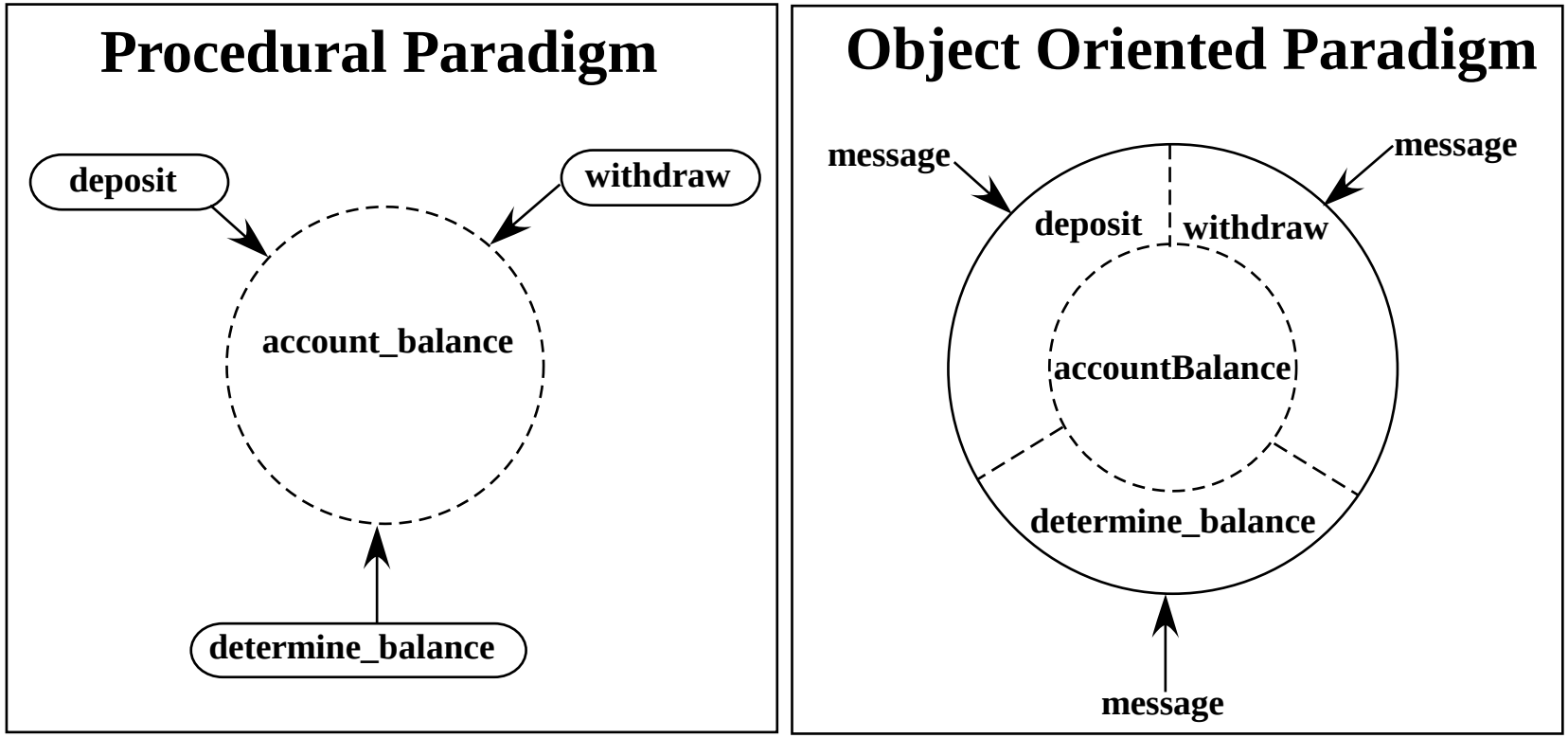
Object Oriented Software Engineering

Important Terms

- **Data Abstraction**
- **Encapsulation**
- **Messaging**
- **Modularity**
- **Polymorphism**
- **Inheritance**

Object Oriented Software Engineering

Think *Object Oriented*



See the difference?

Object Oriented Software Engineering

Change your way of thinking...

BankAccount

- **You are modeling real world objects. Classes represent these real objects so designing is easier.**

To Do List
1. Deposit
2. Withdraw
3. Get Balance

- **Think of objects in your program as independent units with certain assigned responsibilities.**

How'd you do that?

- **Nothing outside the object needs to know HOW it performs its responsibilities. Just tell it to do so.**

- **Post-delivery maintenance is easier because there is less change of a Regression Fault.**

CustomerAccount

- **Well designed objects are easier to reuse in other programs.**

The Unified Process

- The primary methodology used today.
- Originally published in 1999 by Ivar Jacobson, Grady Booch and James Rumbaugh, the developers of UML.
- Takes an iterative, incremental approach to software development.
- It is an adaptable approach, i.e. one size does not fit all.

Object Oriented Software Engineering

The Unified Process

An iterative, incremental, and adaptable approach

How do we move from...

Classical Phases in Software Development

1. Requirements Specification Phase
2. Analysis Phase
3. Design Phase
4. Implementation Phase
5. Testing Phase

...to...

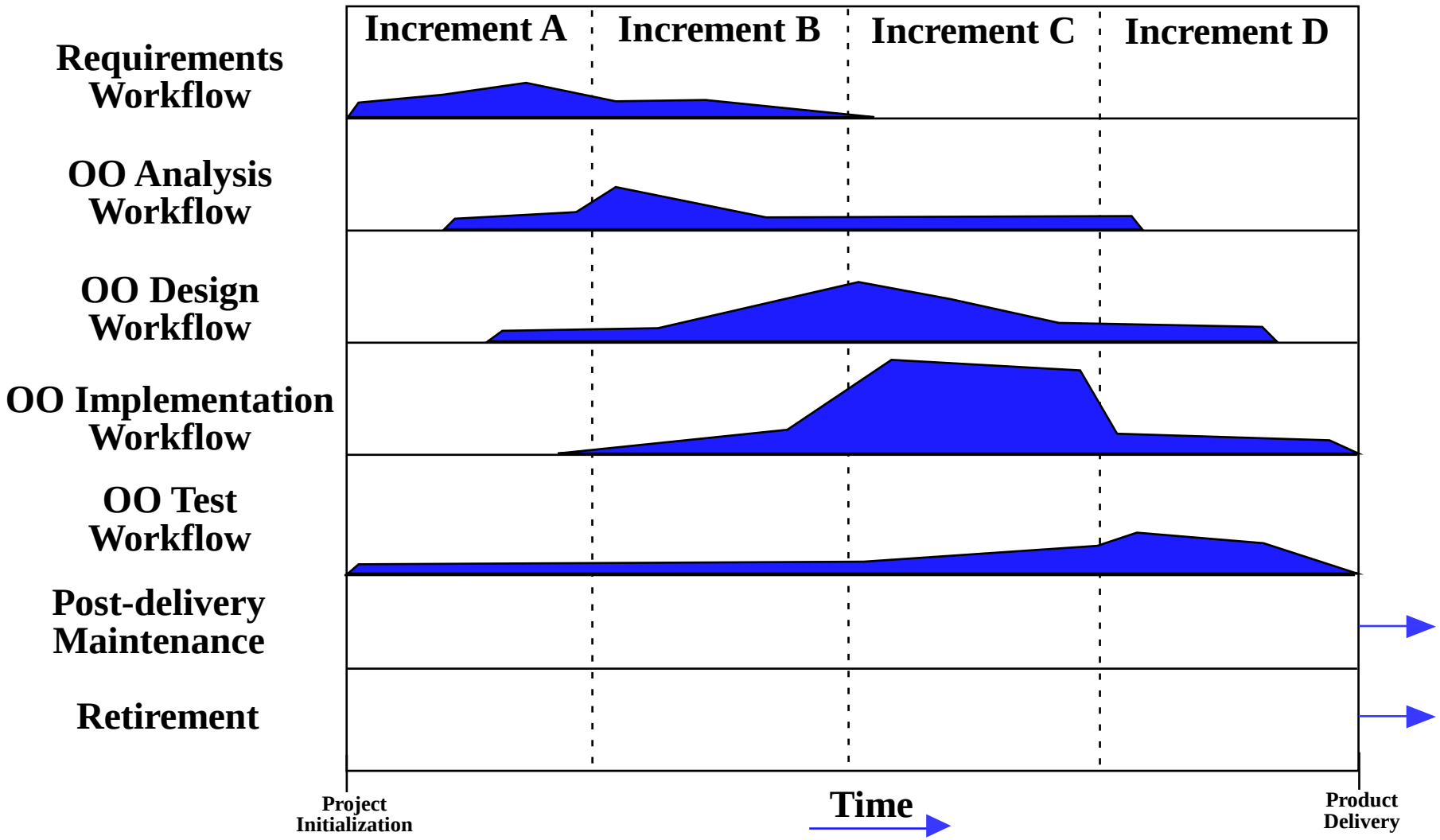
Workflows in the Unified Process

1. Requirements Workflow
2. Analysis Workflow
3. Design Workflow
4. Implementation Workflow
5. Test Workflow
6. Post-delivery Maintenance
7. Retirement

Object Oriented Software Engineering

The Unified Process

An iterative, incremental, and adaptable approach



The Unified Process Requirements Workflow

Primary aim is to determine the customer's needs.

Understand the Application Domain

Subject matter and specific environment.

Carry out a Concept Exploration

Meet with the customer to discuss needs.

Define the Constraints

Things like deadline, cost, reliability, size, etc.

During the Requirements Workflow you write the Requirements Definition Document. Work is begun on the Software Test Plan.

The Unified Process
Analysis Workflow

Primary aim is to analyze and refine the requirements.

Activities

1. *List all the requirements with technical details.*
2. *Determine what the deliverables will be.*
3. *List the major milestones.*
4. *Determine what the budget will be.*
5. *Prepare an Architectural Design (determine modules, units, and packages).*

BTW

Stating Requirements Precisely:

A part record and a plant record are read from the database. If it contains the letter A directly followed by the letter Q, then calculate the cost of transporting that part to that plant.

What does it refer to?

During the Analysis Workflow you write the Software Development Plan and the Requirements Specification Document and/or a set of UML diagrams. Work continues on the Software Test Plan.

The Unified Process
Design Workflow

Primary aim is to refine the products of the Analysis Workflow until they are in a form that can be implemented.

Activities

1. *Plan the Detailed Design (member variables, member functions, algorithms) from the Architectural Design.*
2. *Keep meticulous records of all design decisions:*
   a. *If a problem is encountered.*
   b. *If future enhancements are made.*
   c. *When the product must be completely redesigned.*

During the Design Workflow you write the Software Design Plan. Work continues on the Software Test Plan.

The Unified Process Implementation Workflow

Primary aim is to implement the design in the chosen language.

Activities

1. *Assign modules to team members.*
2. *Write code.*
3. *Integrate code from all team members.*
4. *Revise code as required while testing.*

Final work is done on the Software Test Plan.

The Unified Process
Test Workflow

Primary aim is to verify and validate that all parts of the product function correctly.

Activities

TEST, TEST, TEST

The Unified Process
Post-delivery Maintenance

Primary aim is to revise code to correct errors found after delivery and to add upgrades and enhancements.

Activities

Document changes.
Perform Regression Testing.

The Unified Process
Retirement

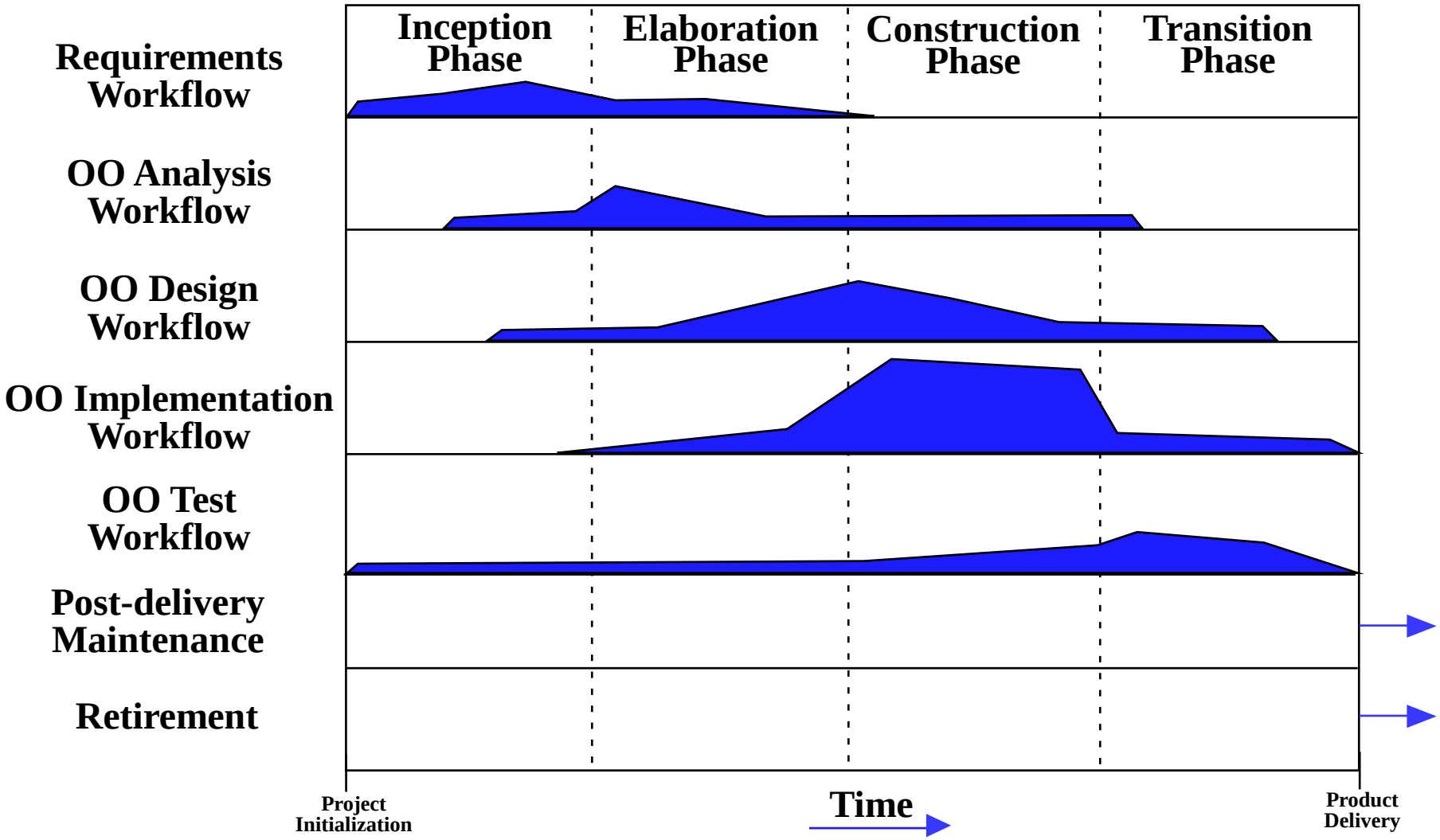
Eventually a time is reached when post-delivery maintenance is no longer cost effective.

Main Reasons for Retirement

1. *Proposed changes require a complete redesign.*
2. *Even a small change requires major work.*
3. *Documentation has not been kept updated.*
4. *The original hardware or operating system has been replaced.*

The Unified Process

The 4 Phases



The Unified Process
The Four Phases

Inception Phase

Primary aim is to is to determine if it is worthwhile to develop the target software product.

Elaboration Phase

Primary aim is to refine the initial requirements, refine the architecture, monitor the risks and refine their priorities, redefine the business case, produce the Software Management Plan.

Construction Phase

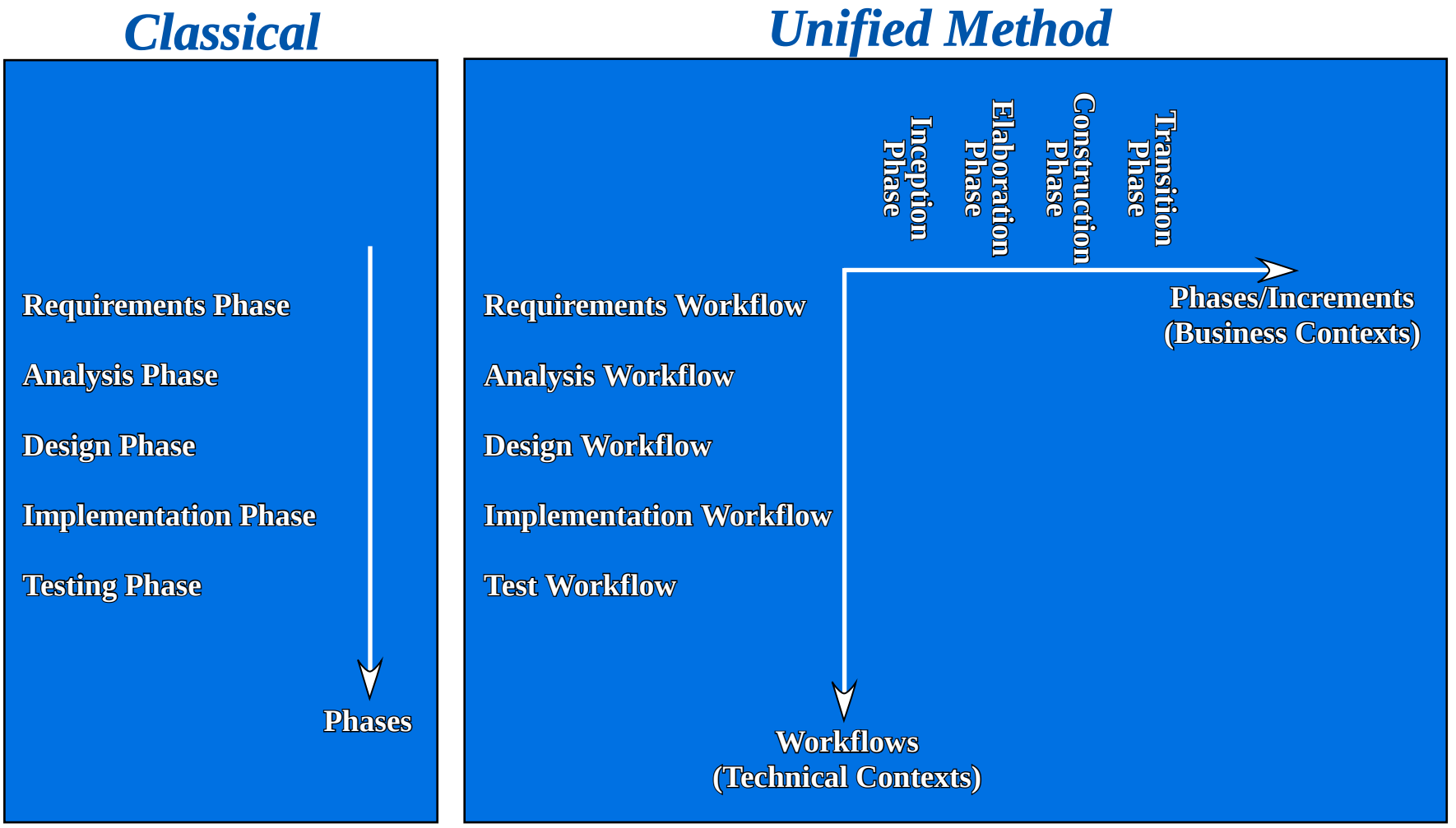
Primary aim is to produce the first operational version of the software (the beta release).

Transition Phase

Primary aim is to ensure the client‘s requirements have been met.

The Unified Process

Comparison of Classical and Unified Methods



One-dimensional Life Cycle Model

Two-dimensional Life Cycle Model

The Unified Process

Warnings

The OO approach can also be misused.

- **There is a learning curve.**
- **Fragile base class problem.**
- **Overuse of inheritance.**
- **Problems with polymorphism and dynamic binding.**
- **It is possible to code badly in any language.**

The Unified Process
Inception Phase

Primary aim is to is to determine if it is worthwhile to develop the target software product.

Deliverables

1. *Initial version of the domain model.*
2. *Initial version of the business model.*
3. *Initial version of the requirements artifacts (The Requirements Description Document).*
4. *Preliminary version of analysis artifacts (notes on what was decided and why).*
5. *Initial list of risks*
6. *Initial Use Cases.*
7. *Plan for Elaboration Phase*
8. *Initial version of the Business Case.*

The Unified Process
Elaboration Phase

Primary aim is to refine the initial requirements, refine the architecture, monitor the risks and refine their priorities, redefine the business case, produce the software management plan.

Deliverables

1. *Completed domain model.*
2. *Completed business model.*
3. *Completed requirements artifacts.*
4. *Completed analysis artifacts.*
5. *Updated version of the architecture.*
6. *Updated list of risks.*
7. *Software Management Plan (for the rest of the project)*
8. *Completed business plan.*

The Unified Process
Construction Phase

Primary aim is to produce the first operational version of the software (the beta release).

Deliverables

1. *Initial user manual and other manuals as appropriate.*
2. *The beta release version of the software.*
3. *The completed architecture.*
4. *The updated risk list.*
5. *Software Management Plan*
   (updated for the rest of the project)
6. *If required, the updated business case.*

The Unified Process

Transition Phase

Primary aim is to ensure the client's requirements have been met.

Deliverables

1. All the artifacts
(final versions of documentation and application).
2. The completed manuals.