MSc Information Systems

Systems Development Life Cycle

Module SITS code: COIY059H7

Dr Brian Gannon

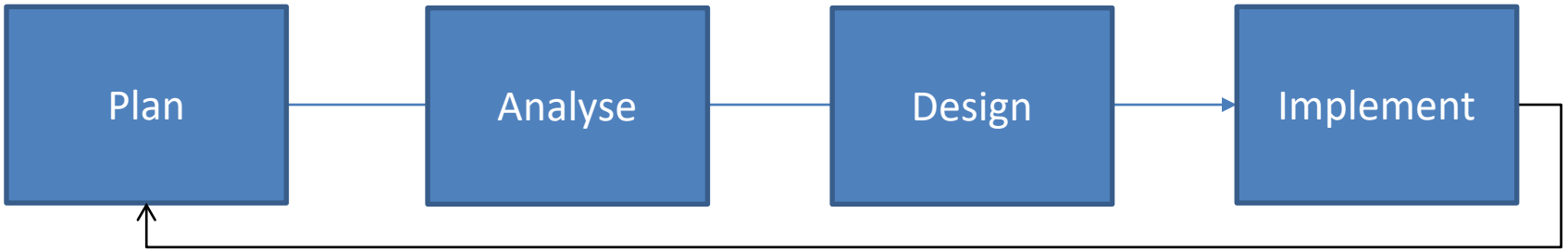
Systems Development Life Cycle

What is the SDLC?

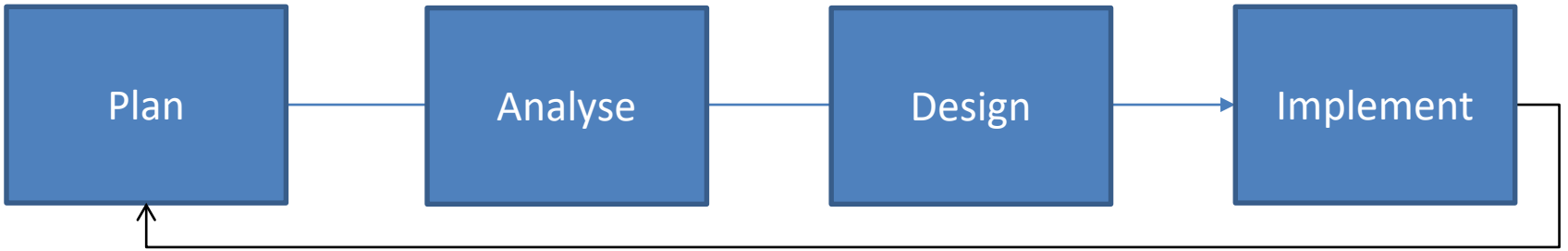
Why do we need a formal process?

- Failures occur (too) often
- Creating systems is not intuitive
- Projects are late, over budget or delivered with fewer features than planned
- Designed systems can be focused to deliver business value

SDLC Activities

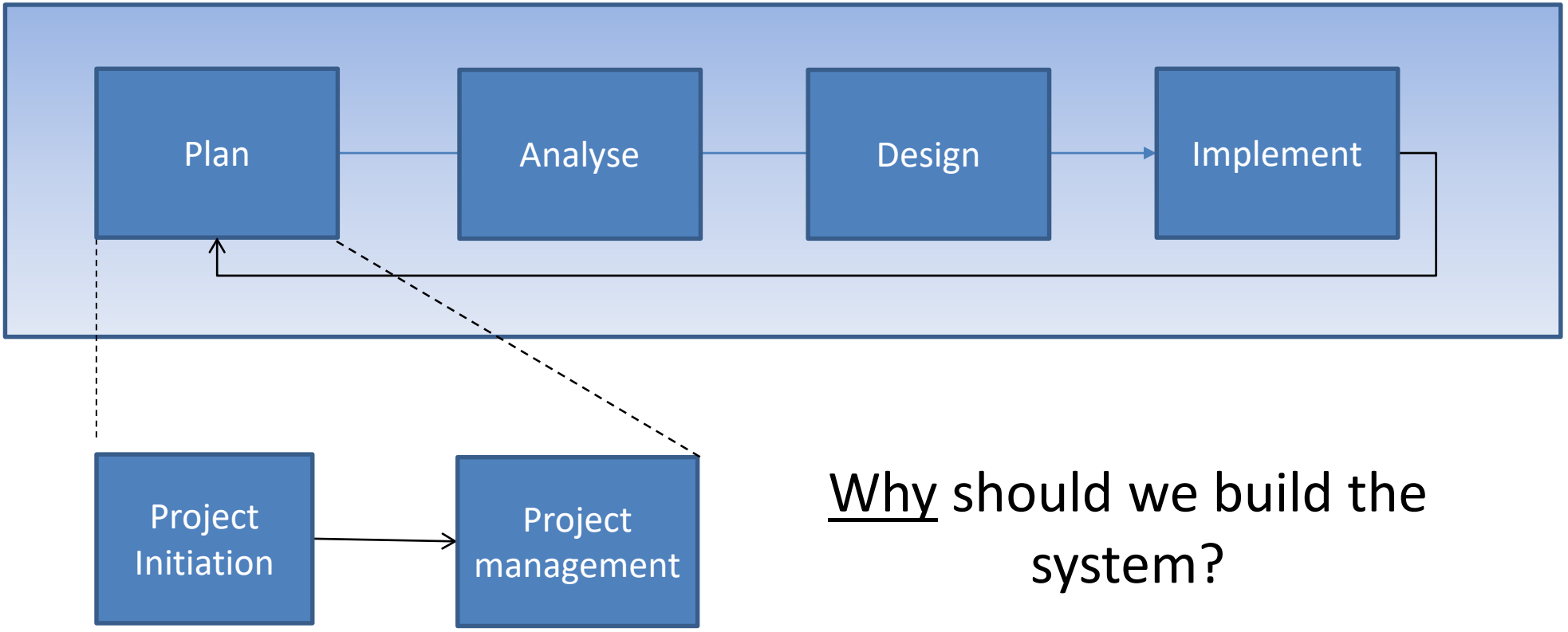


SDLC Activities



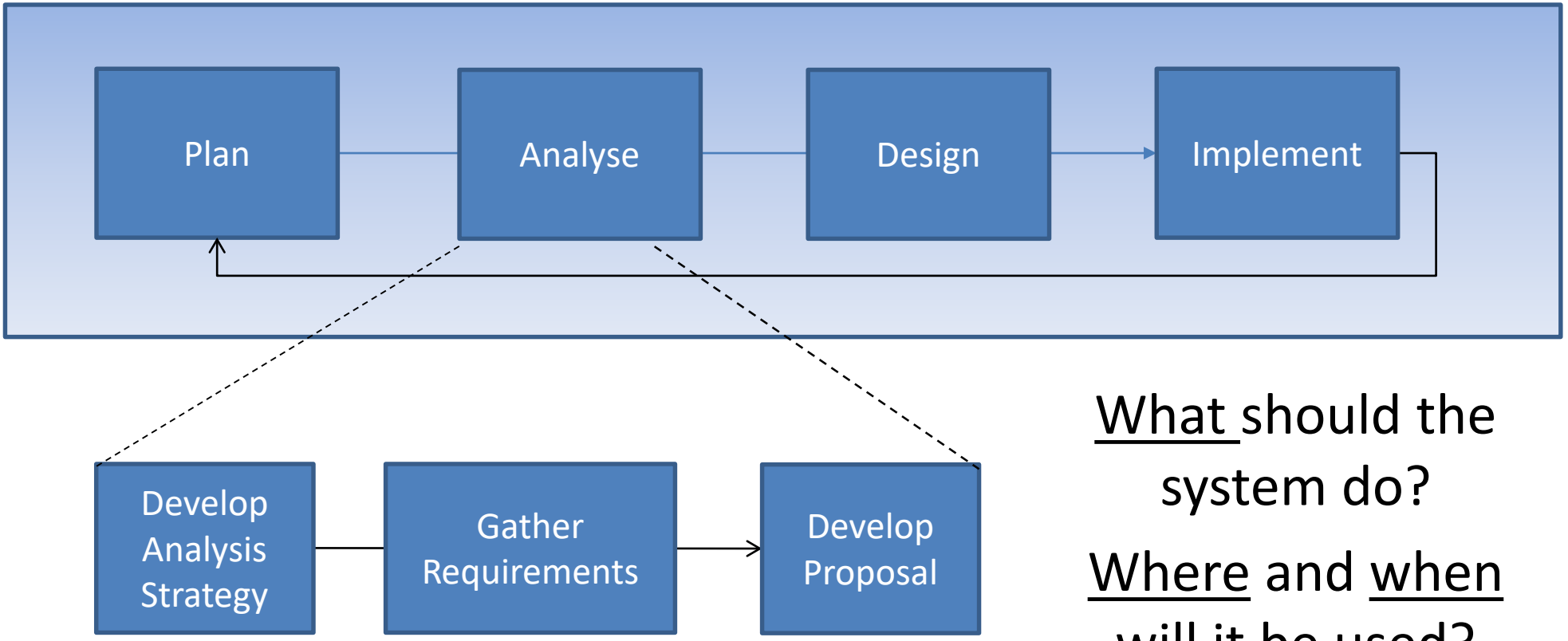
- Each phase consists of steps that lead to specific outputs (‘deliverables’)
- The system evolves through gradual refinement
- Once the system is implemented, it may go back into a planning phase for its next revision, a follow-on system, or maintenance releases

Planning



Why should we build the system?

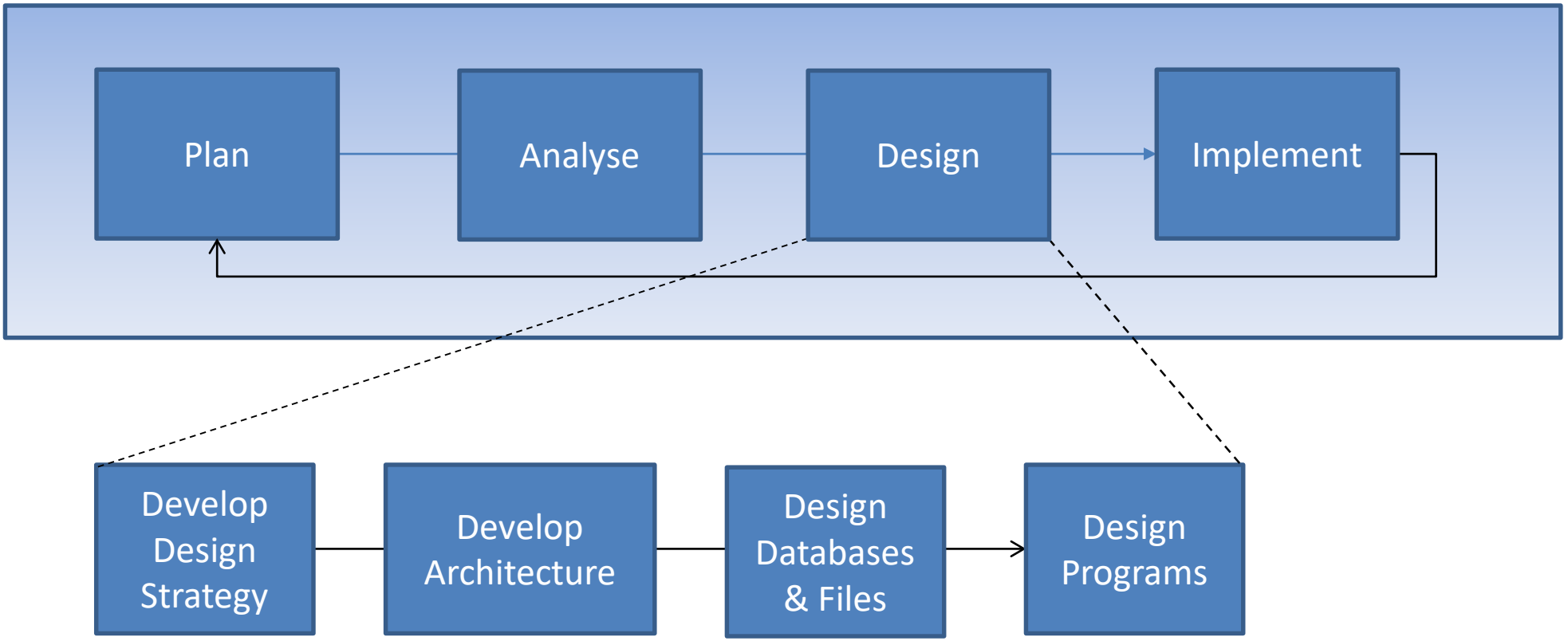
Analysis



What should the system do?

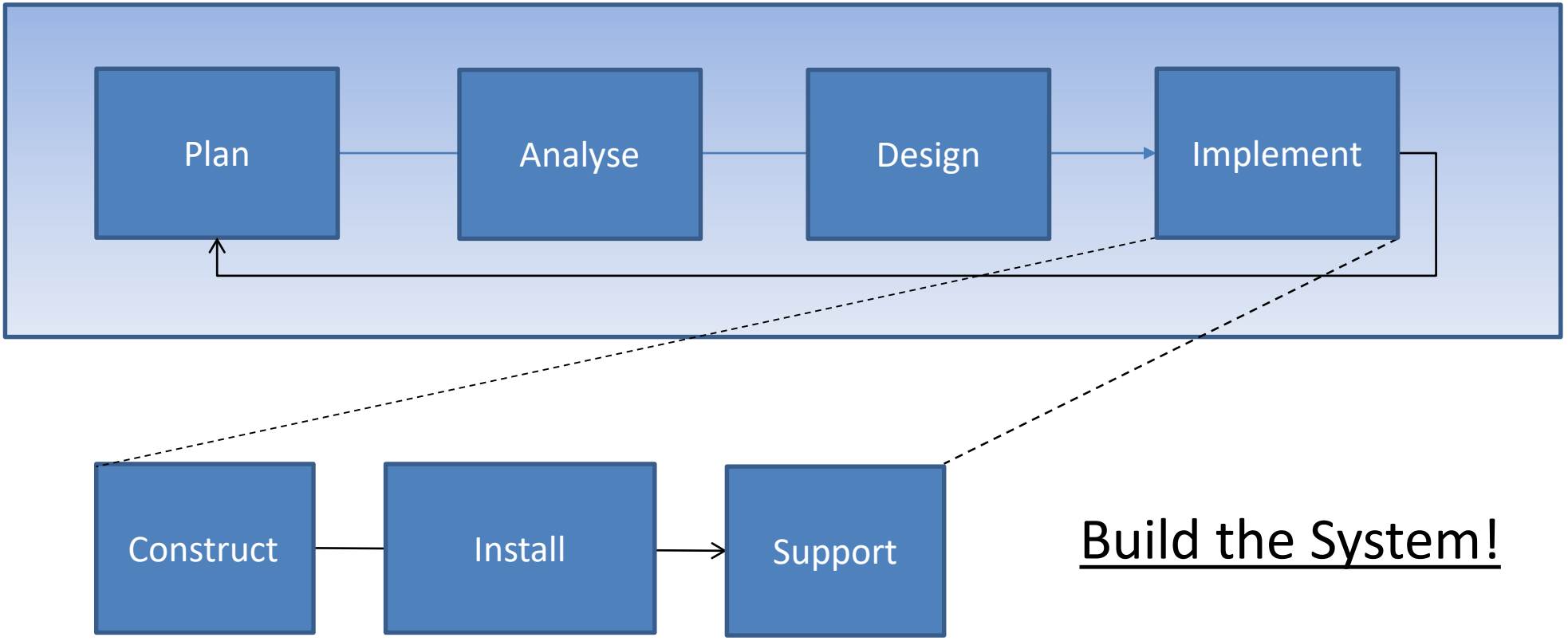
Where and when will it be used?

Design

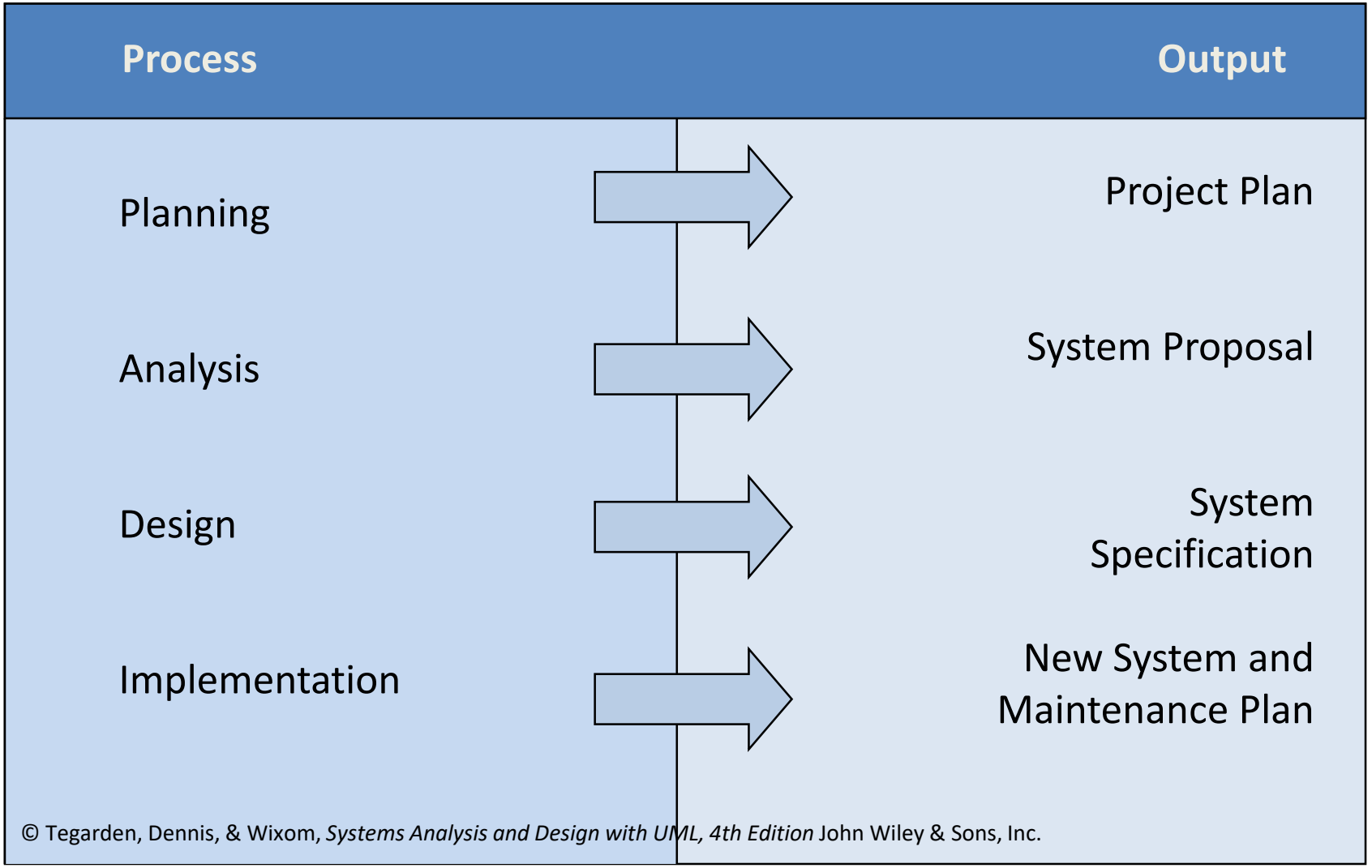


How will the system be built?

Implementation



SDLC Process And Outputs



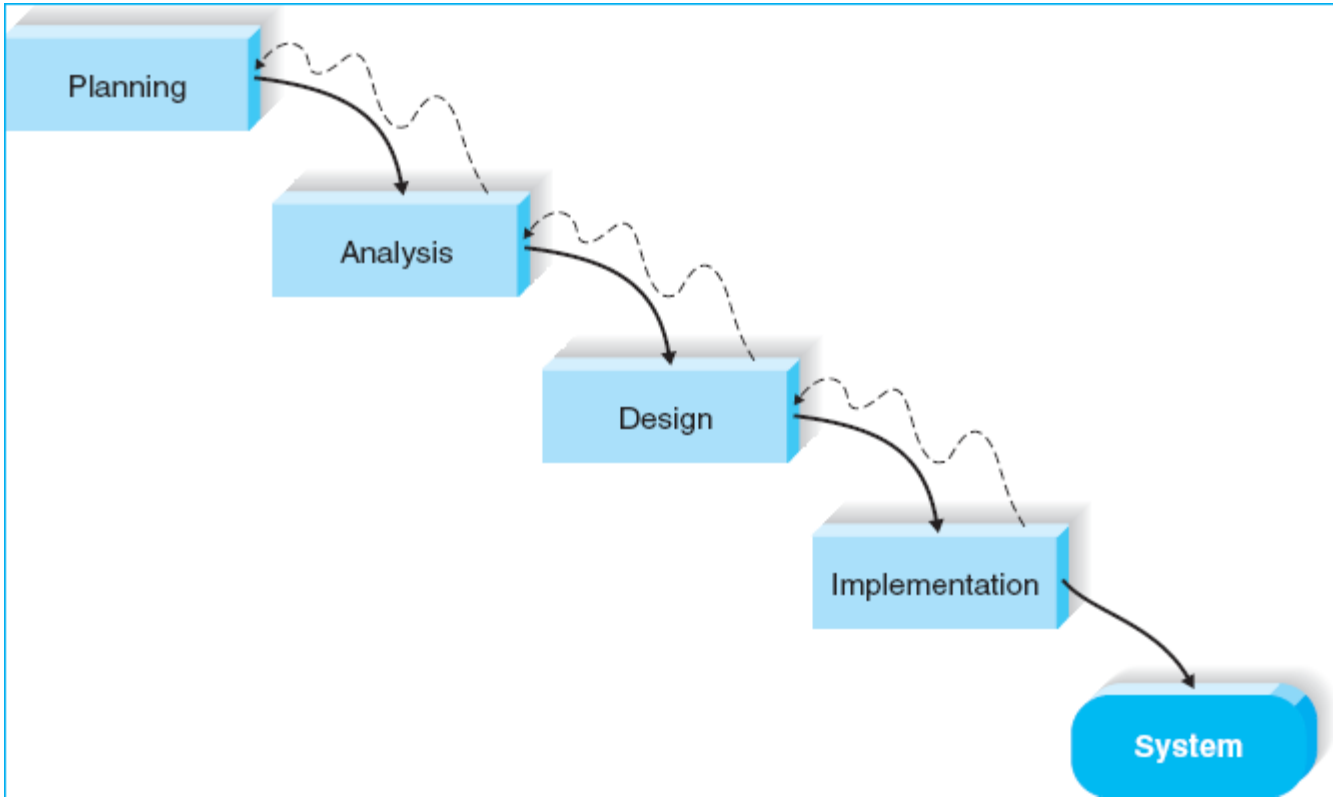
Systems Development Methodology

- A methodology is a formalised approach to implementing the SDLC
- A methodology can be:
    - Process-oriented
    - Data centered
    - Object-oriented
    - Structured
    - Agile
    - Other….

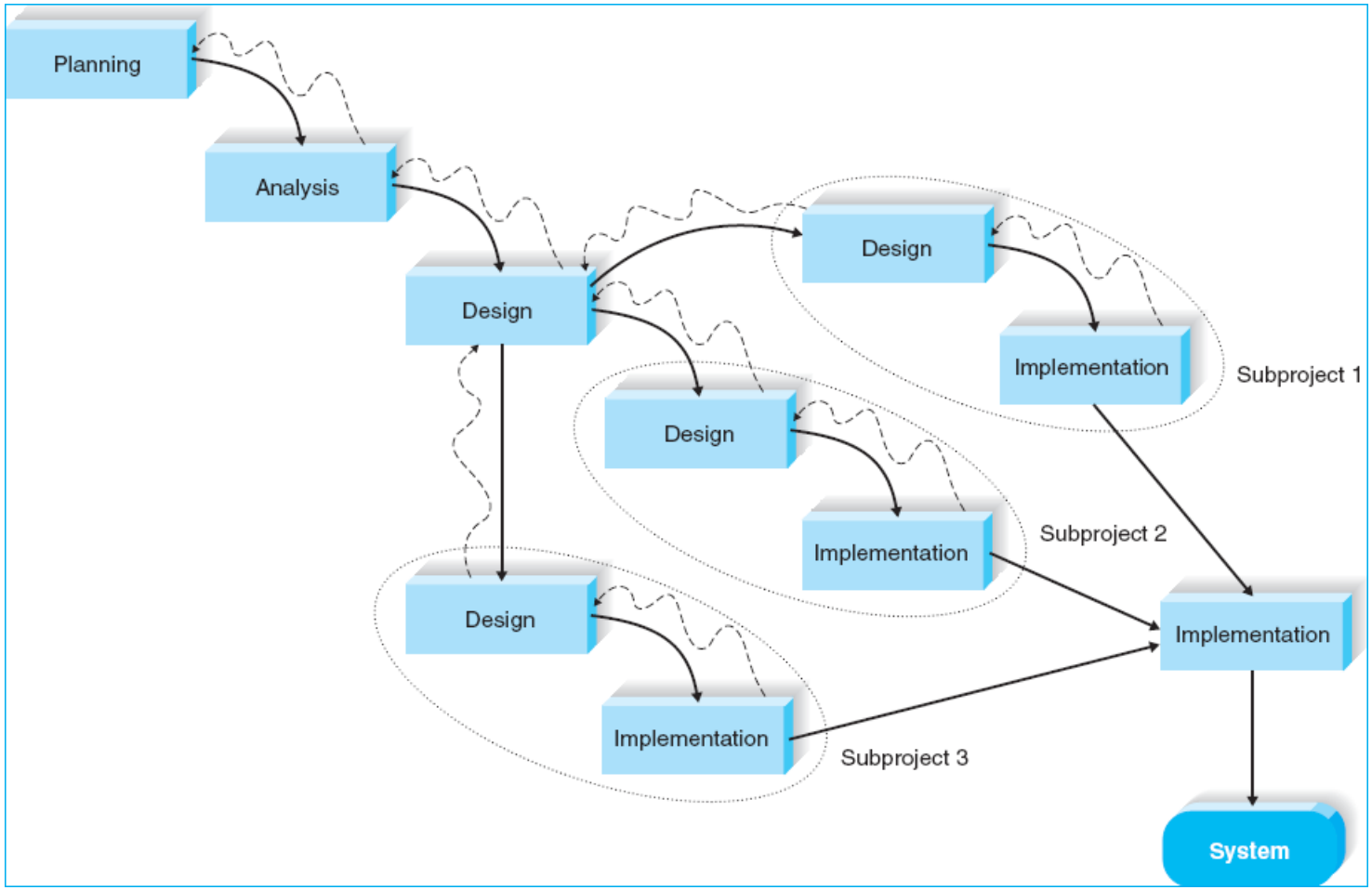
Methodology Types

- Structured Design
  - Waterfall Development
  - Parallel Development
- Rapid Application Development
  - Phased
  - Prototyping
  - Throwaway Prototyping
- Agile Development
  - eXtreme Programming
  - SCRUM

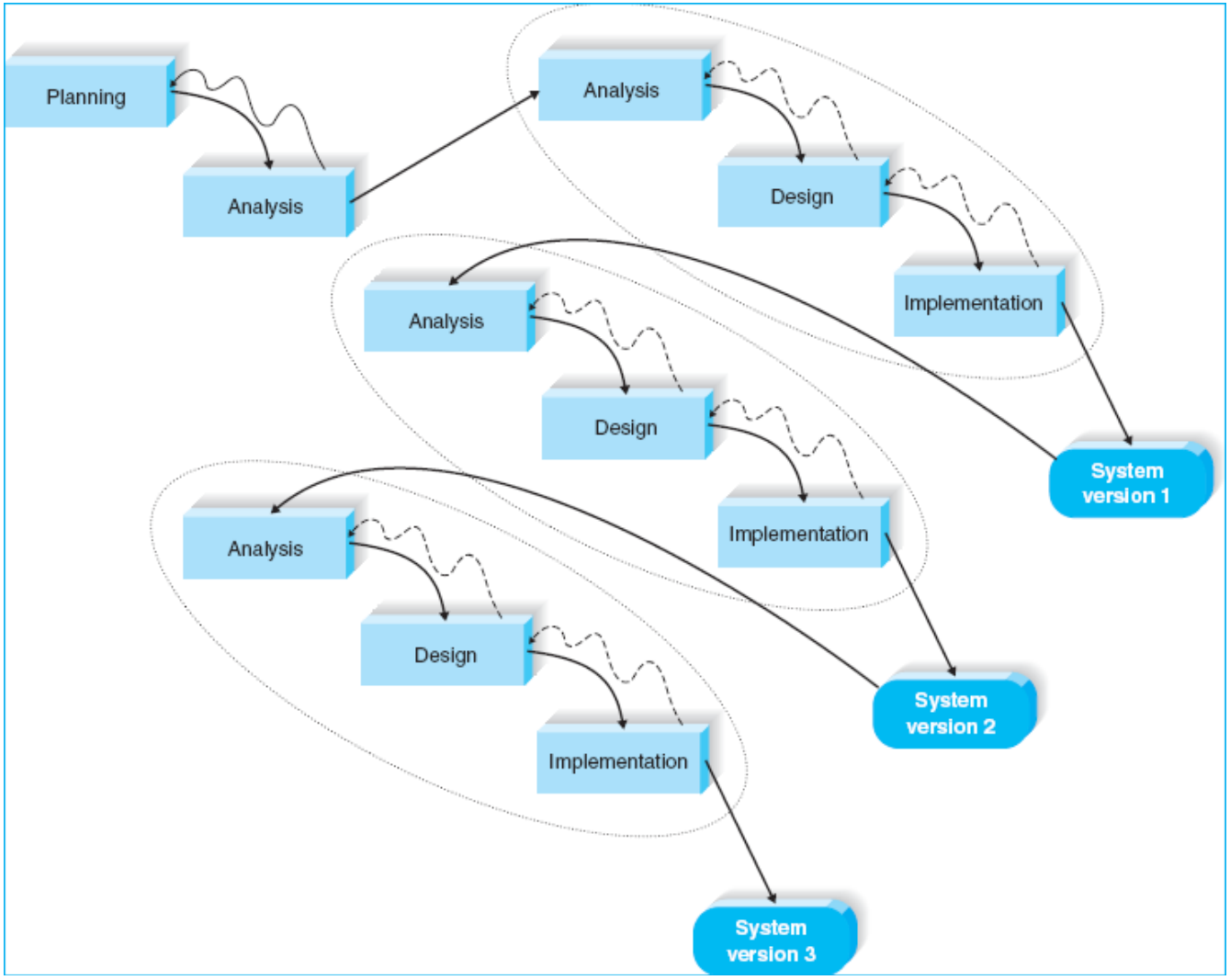
Structured Design - Waterfall



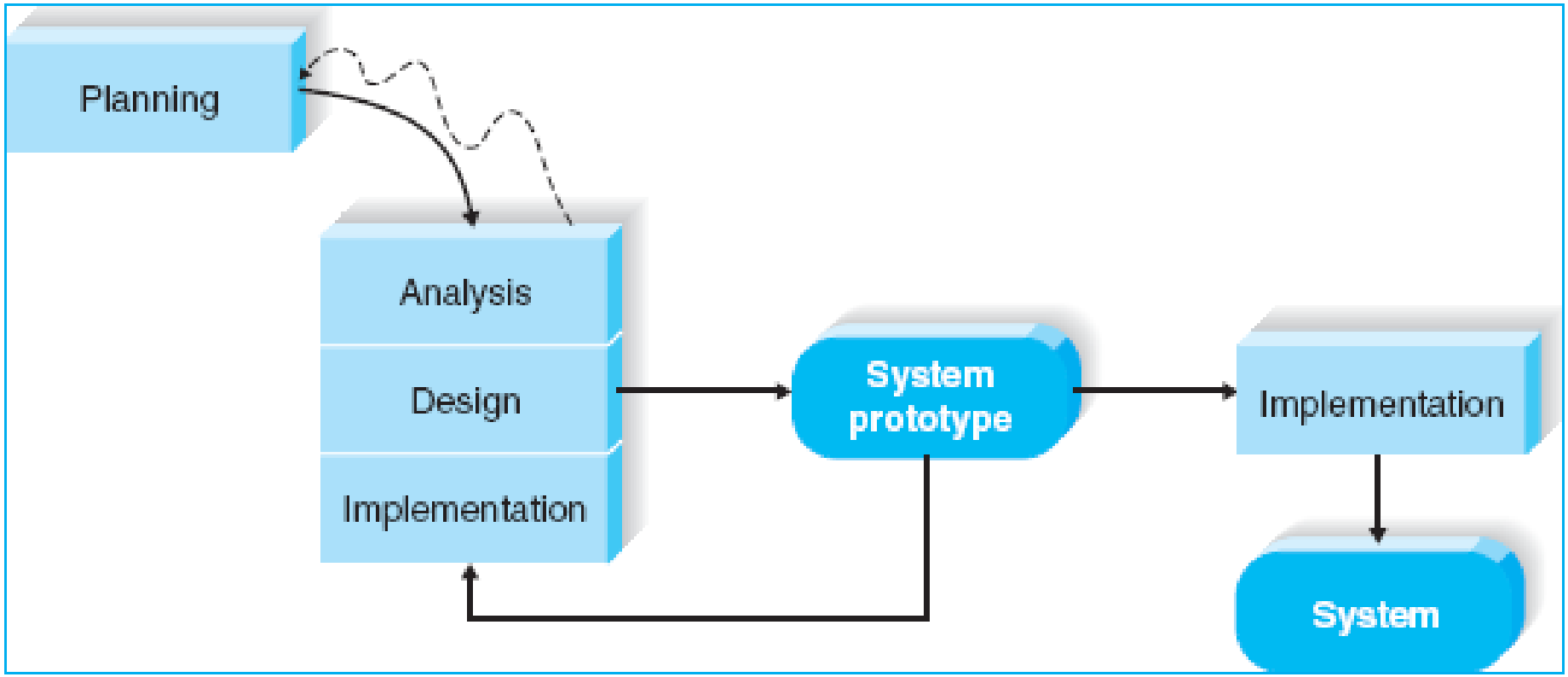
Parallel Development



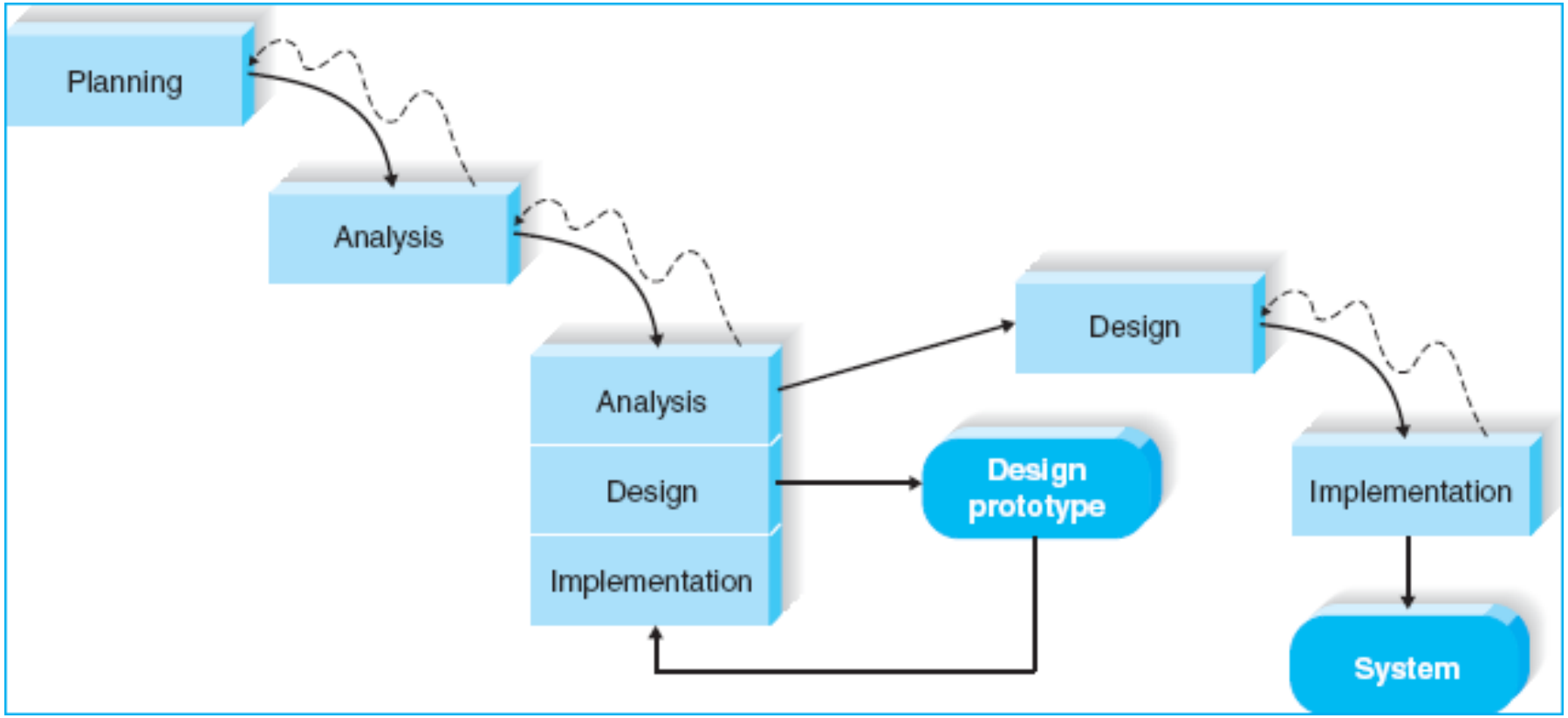
Rapid Application Development - Phased



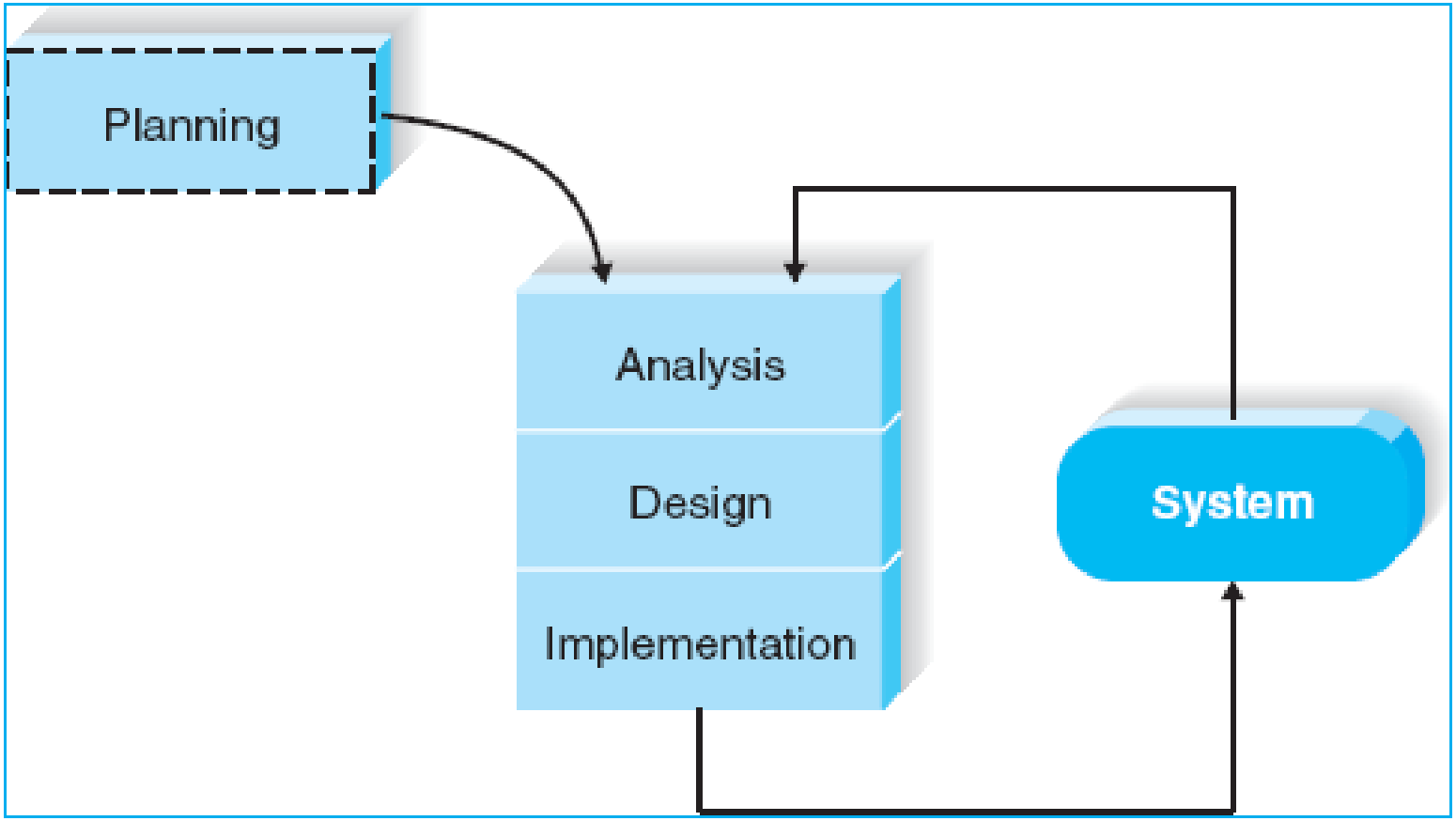
Rapid Application Development - Prototyping



Rapid Application Development – Throwaway Prototyping



Agile Methodologies



Object-Oriented Analysis and Design (OOAD)

- In the past, a program was viewed as a logical procedure that takes input data, processes it, and produces output data
- Early programming paradigms were based on **structured programming** techniques: (subroutines, IF-THEN loops / simple series of computational steps)
- Early design/analysis was based on structured flowcharts
- Early structured languages: C, Fortran, Pascal, BASIC, COBOL
- Typical applications included computationally-intensive processes (scientific number crunching, General Ledger, Payroll, etc.)

Object-Oriented Analysis and Design (OOAD)

- Introduction of event-driven programs
- Introduction of GUI (which allowed lots of different paths through workflow)
- Increasing complexity, longevity, cost of programming, with need for efficiency and reuse
- Drive for better linkage between business process and computer program
- …all led to the OO paradigm – and attempt to balance data and process

Object-Oriented Analysis and Design (OOAD)

- OOAD is **Use-Case driven**, **Architecture-centric**, **Iterative and Incremental**

- Use-case driven
  - Use-cases define the behavior of a system
  - Each use-case focuses on one business process

- Architecture centric
  - Functional (external) view: focuses on the user's perspective
  - Static (structural) view: focuses on attributes, methods, classes & relationships
  - Dynamic (behavioral) view: focuses on messages between classes and resulting behaviors

OOAD Benefits

- **Code Reuse and Recycling**: objects created for OO programs can be reused in other programs

- **Encapsulation**: once an Object is created, knowledge of its implementation is not necessary for its use.

- **Encapsulation**: objects can hide certain parts of themselves from programmers, preventing them from tampering with values they shouldn't.

- **Design Benefits**: OO programs force designers to go through an extensive planning phase, which makes for better designs with less flaws.

- **Software Maintenance:** OO programs are easier to modify and maintain than non-OO programs. (More time spent on analysis, but less effort to maintain).

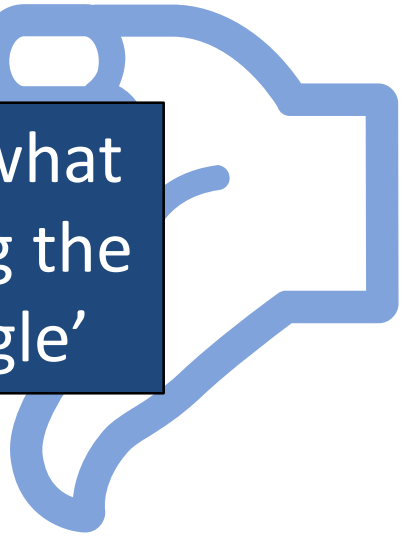
OOAD Criticisms

- OOP not really reusable and modular
- Emphasises design and modeling (data/objects) at the expense of computation/ algorithms
- OOP code is "intrinsically less efficient" than procedural code (can take longer to compile, and extremely complex)
- No significant difference in productivity between OOP and procedural approaches

OOAD Criticisms

- OOP not really reusable and modular
- Emphasises design and modeling (data/objects) at the expense of computation/
- OOP code is "
than procedu
compile, and
- No significant difference in productivity between OOP and procedural approaches

‘You wanted a banana but what you got was a gorilla holding the banana and the entire jungle’



The Unified Process

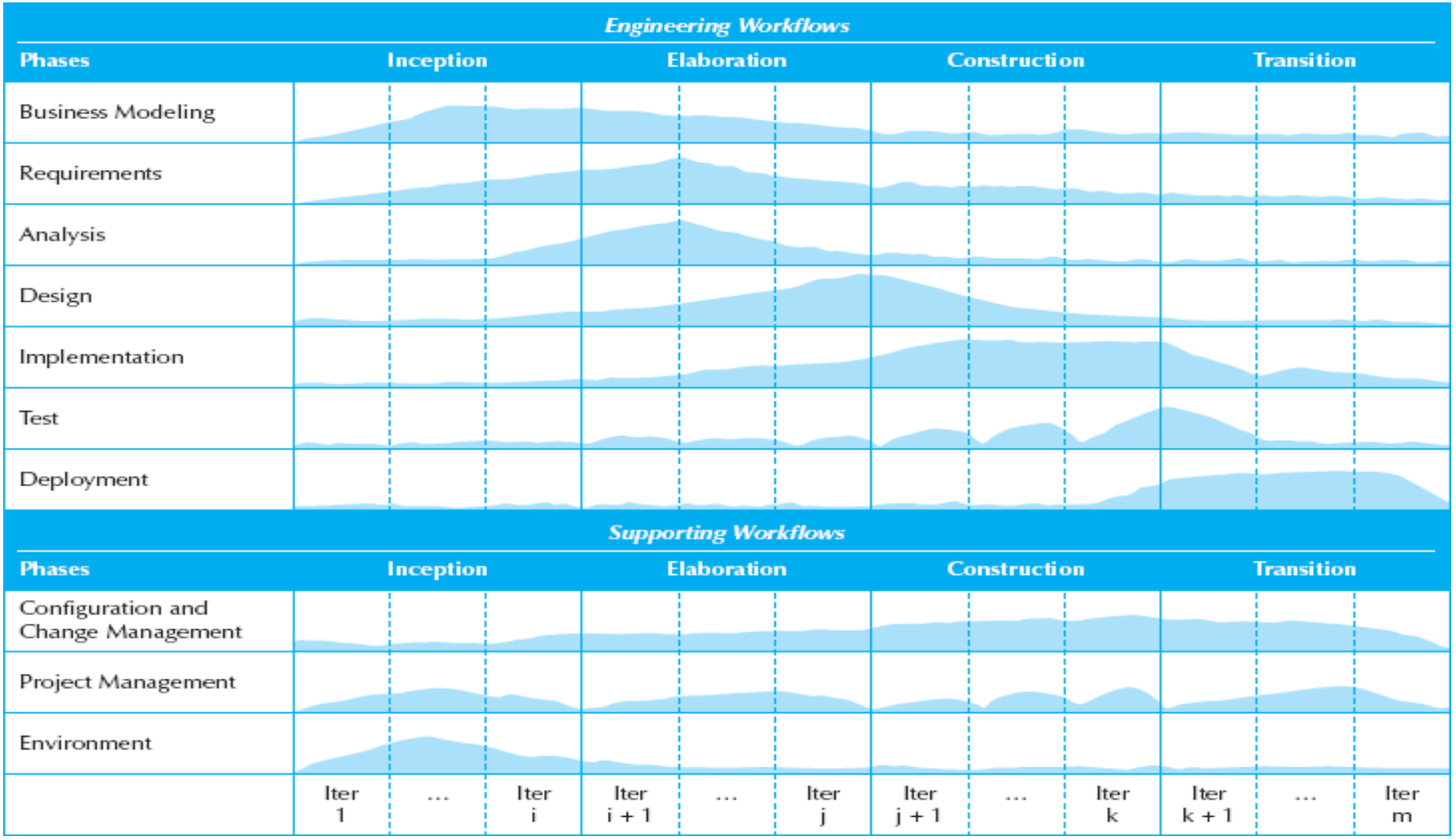
- A specific methodology for object-oriented analysis and design

- Comprises:
  - Four **Phases** that describe how the system evolves over time (Inception/Elaboration/Construction/Transition)

  - Seven **Engineering Workflows** that describe collections of tasks that occur throughout the lifecycle

  - Five **Support Workflows** that describe disciplines needed to ensure consistency of approach and method

The Unified Process



Unified Modelling Language (UML)

- The Unified Process is described by the **Unified Modelling Language (UML)**
- UML provides a common vocabulary of object-oriented terms and diagramming techniques to model any systems development project from analysis through implementation
- UML uses **Structure** diagrams and **Behavior** diagrams
- UML does not include Staffing, Budgeting, Contract Management, Maintenance, Operations, Support, Cross-project issues

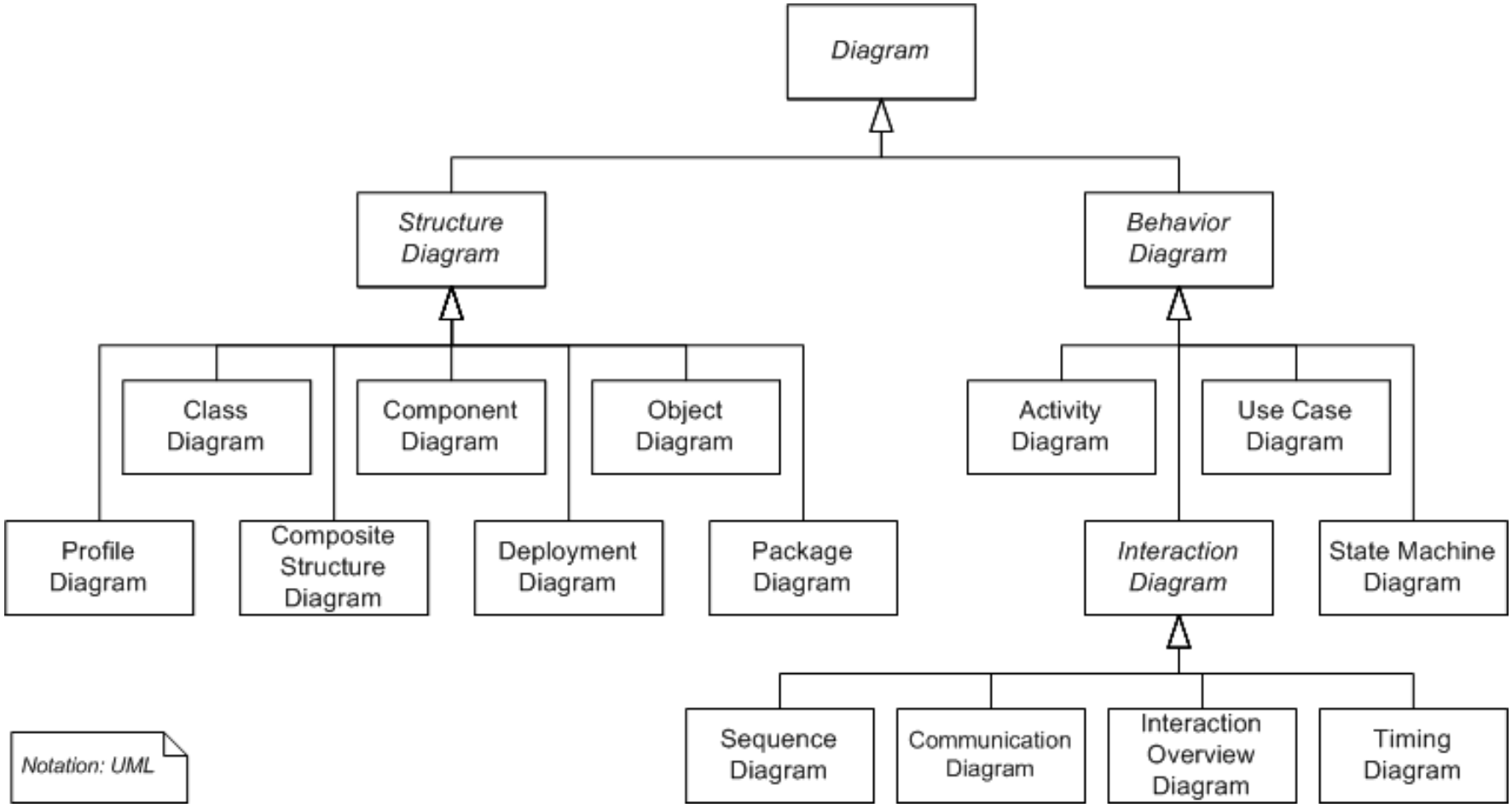
UML Diagrams

- UML diagrams represent two different views of a system model

    1. **Static (or Structural) view**: emphasises the static structure of the system using objects, attributes, operations and relationships. It includes class diagrams and composite structure diagrams.

    2. **Dynamic (or Behavioral) view**: emphasises the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams and state machine diagrams

UML Diagram Hierarchy



Paulo Merson: https://commons.wikimedia.org/wiki/File%3AUml_diagram2.png

OOP Languages

- Most widely used programming languages are multi-paradigm programming languages that support object-oriented programming to a greater or lesser degree, typically in combination with imperative, procedural programming.

- Significant object-oriented languages include Java, C++, C#, Python, PHP, Ruby, Perl.

OOP Terminology

- **Classes & Objects**
  - Object (instance): instantiation of a class
  - Attributes: information that describes the class
  - State: describes its values and relationships at a point in time
- **Methods & Messages**
  - Methods: the behavior of a class
  - Messages: information sent to an object to trigger a method (procedure call)

OOP Terminology

- **Encapsulation & information hiding**
  - Encapsulation: combination of process & data
  - Information hiding: functionality is hidden
- **Inheritance**
  - General classes (superclasses)
  - Subclasses can inherit data and methods from a superclass
- **Polymorphism & dynamic binding**
  - Polymorphism: the same message can have different meanings
  - Dynamic binding: type of object is not determined until run-time
  - Contrast with static binding

Open Source (free)

Violet UML Editor: UML drawing tool.
UMLet: UML drawing tool.
yEd : free general-purpose diagramming program.
StarUML: UML modelling tool. Open-Source.
BOUML: UML modelling tool. Can specify and generate code in C++, Java, Python, etc.

Commercial

Microsoft Visio: general purpose drawing tool with UML3 stencil.
Poseidon for UML: UML modelling tool. Available in school labs.
Rational Rose: UML modelling tool. Available in school labs.
Wikipedia: List of UML tools

References

Wikipedia: Object-Oriented Analysis and Design
Wikipedia: Unified Modeling Language (UML)
OMG: Unified Modeling Language (UML)

Dr. BRIAN GANNON

E: b.gannon@bbk.ac.uk
E: brian.gannon@vesime.com
M: 07870 153855
LI: http://lnkd.in/QDXaH6
Tw: @bjgann