Information Systems Analysis & Modeling

Lecture 2: Analysis and modeling fundamentals-SDLC

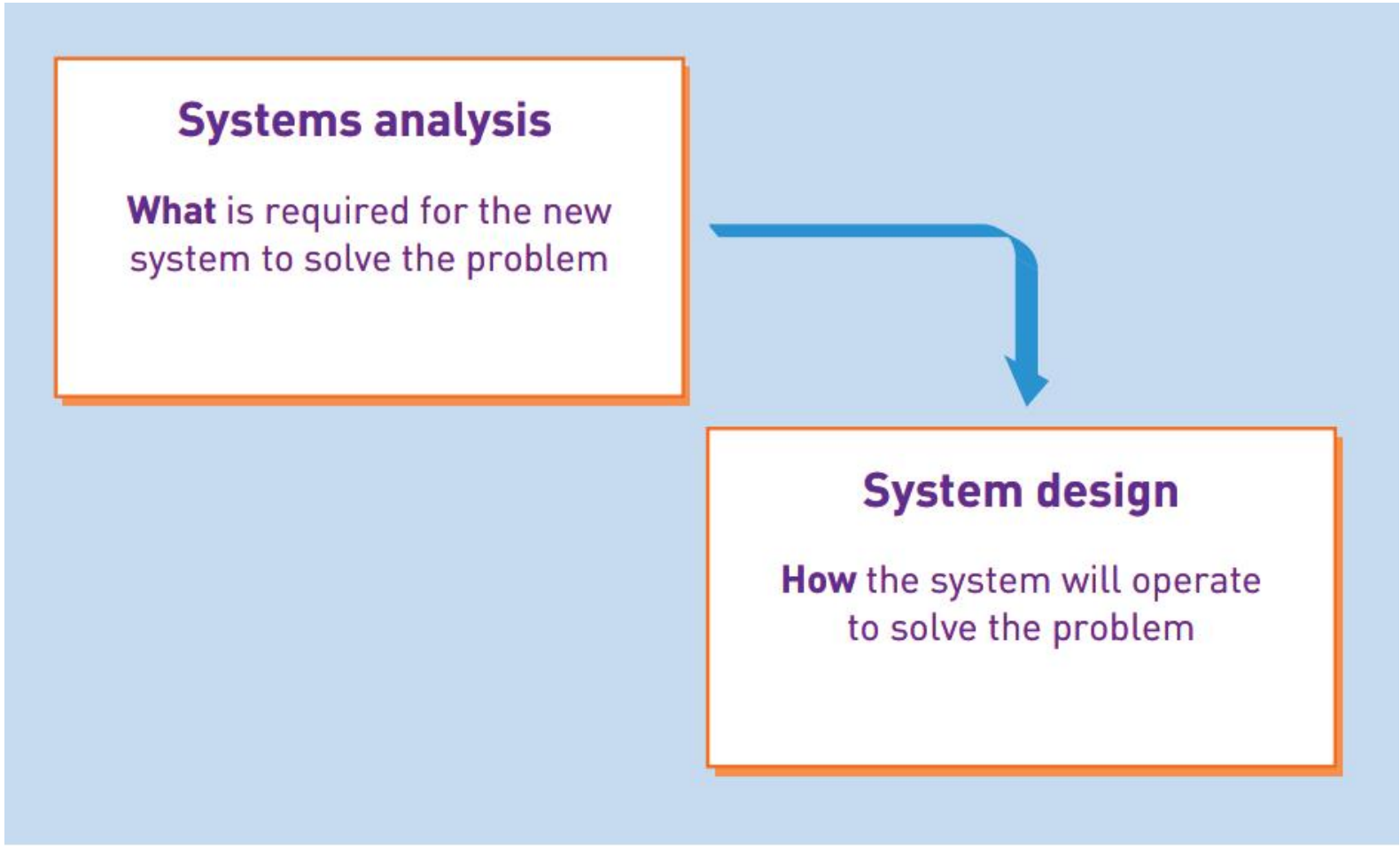
Mona Taghavi

LaSalle College
Montréal

System Analysis and Design

- Systems analysis consists of those activities that enable a person to ***understand*** and ***specify*** what the new system should accomplish. Systems analysis is more than a brief statement of the problem.
  - For example, a customer management system must track customers, register products, monitor warranties, and track service levels, among many other functions—all of which have many details. Systems analysis describes in detail what a system must do to satisfy the need or solve the problem.
- Systems design consists of those activities that enable a person to describe in detail how the information system will actually be implemented to provide the needed solution. In other words, systems design describes how the system will actually work.

System Analysis and Design (continue…)



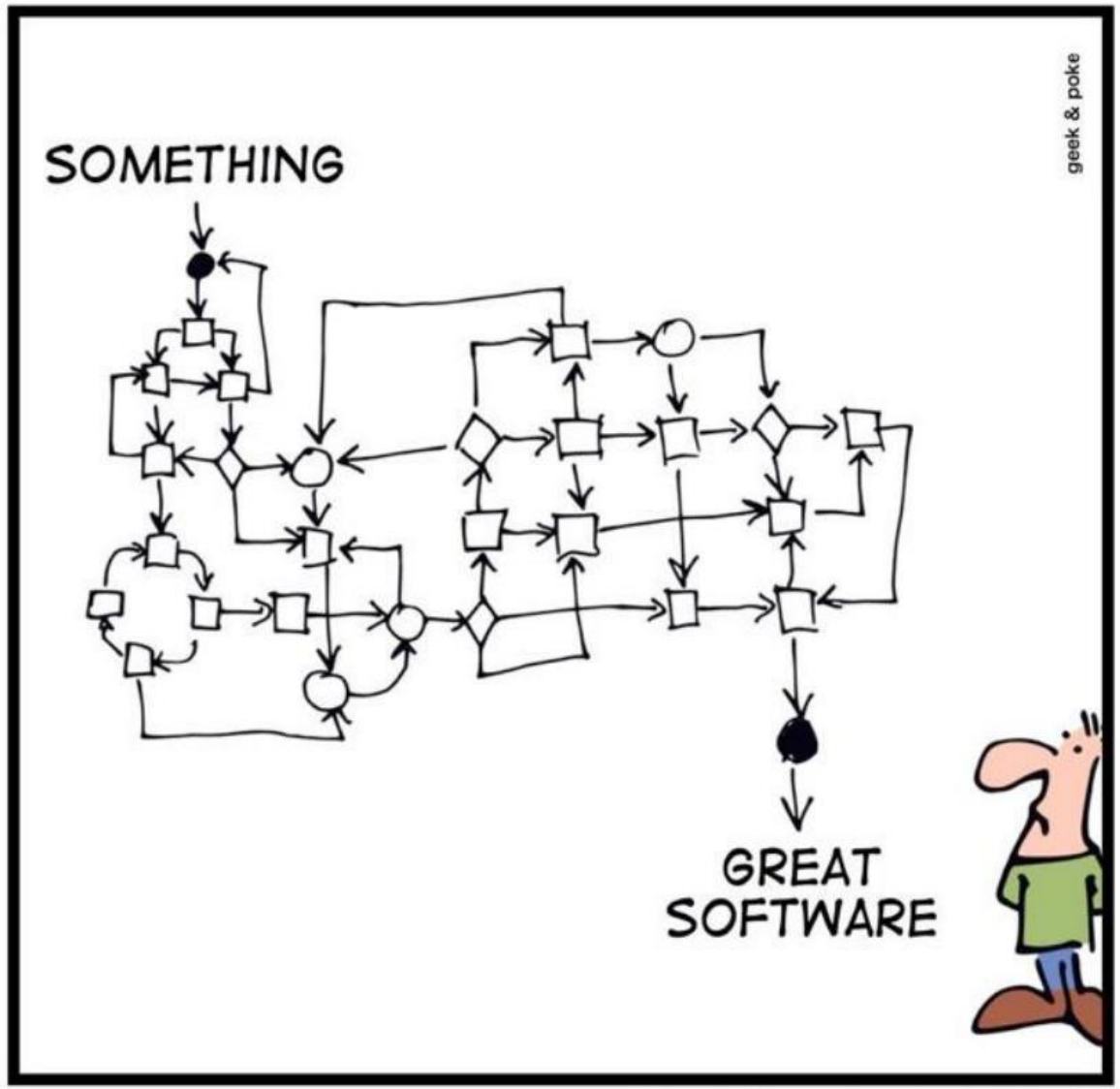
Example: Creating a new building

- The owner of the land: has the vision
- The builder: construct the building
- The architect: serves as the bridge between the owner and the builder
- The architect helps the owner develop the vision, but must also communicate the building's specifications to the builder. In doing so, the architect uses various tools to first capture the vision from the owner and to then provide the builder with instructions—including line drawings, blueprints, to-scale models, detail specifications, and even on-site inspection reports.
- Just as a builder doesn't start construction without plans, programmers don't just sit down and start writing program code.

Role of a system analyst

- Systems analyst: functions like an architect—planning, capturing the vision, understanding details, specifying needs—before designing and writing the code that satisfies the vision.
- What does a system analyst do?
  - Plan, develop and maintain IS
  - Manages IT projects including tasks, resources, schedules and costs
  - Meeting, presentation, report and documentation

SIMPLY EXPLAINED



DEVELOPMENT PROCESS

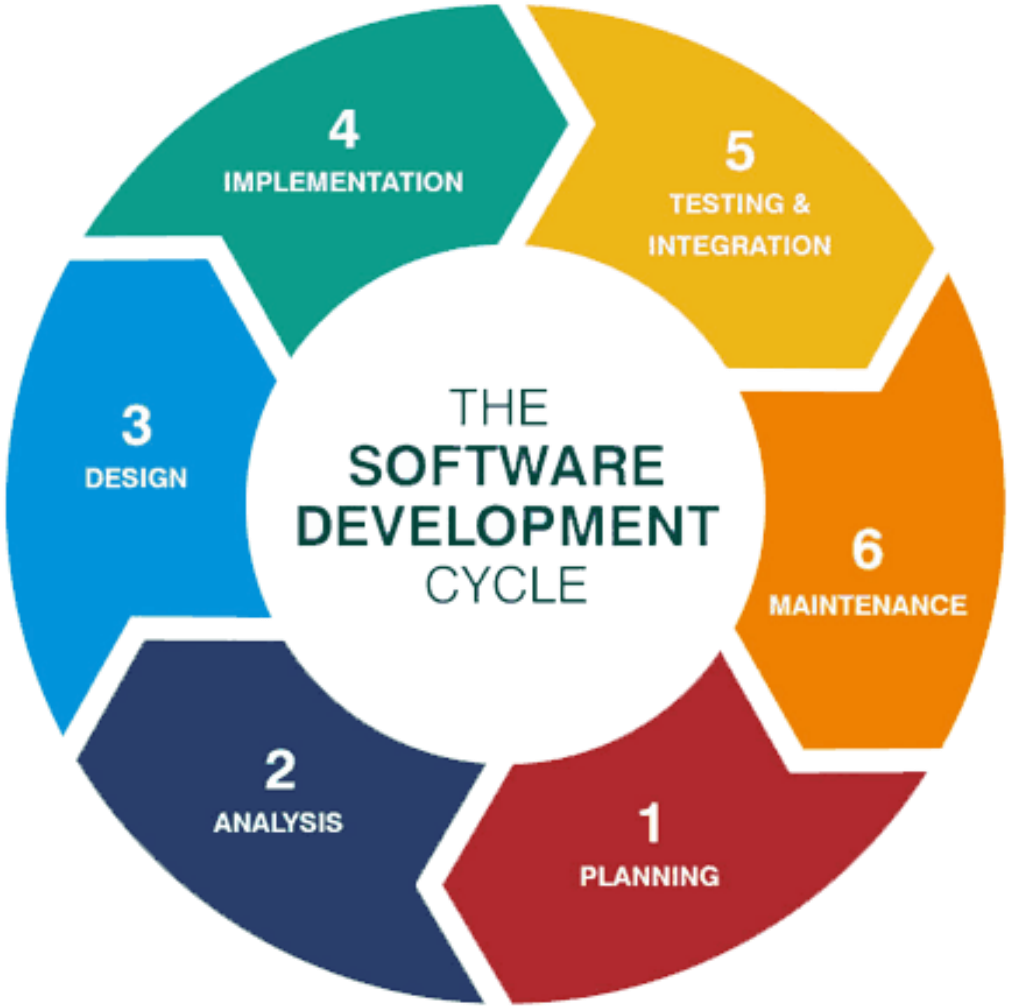
System development is a project!

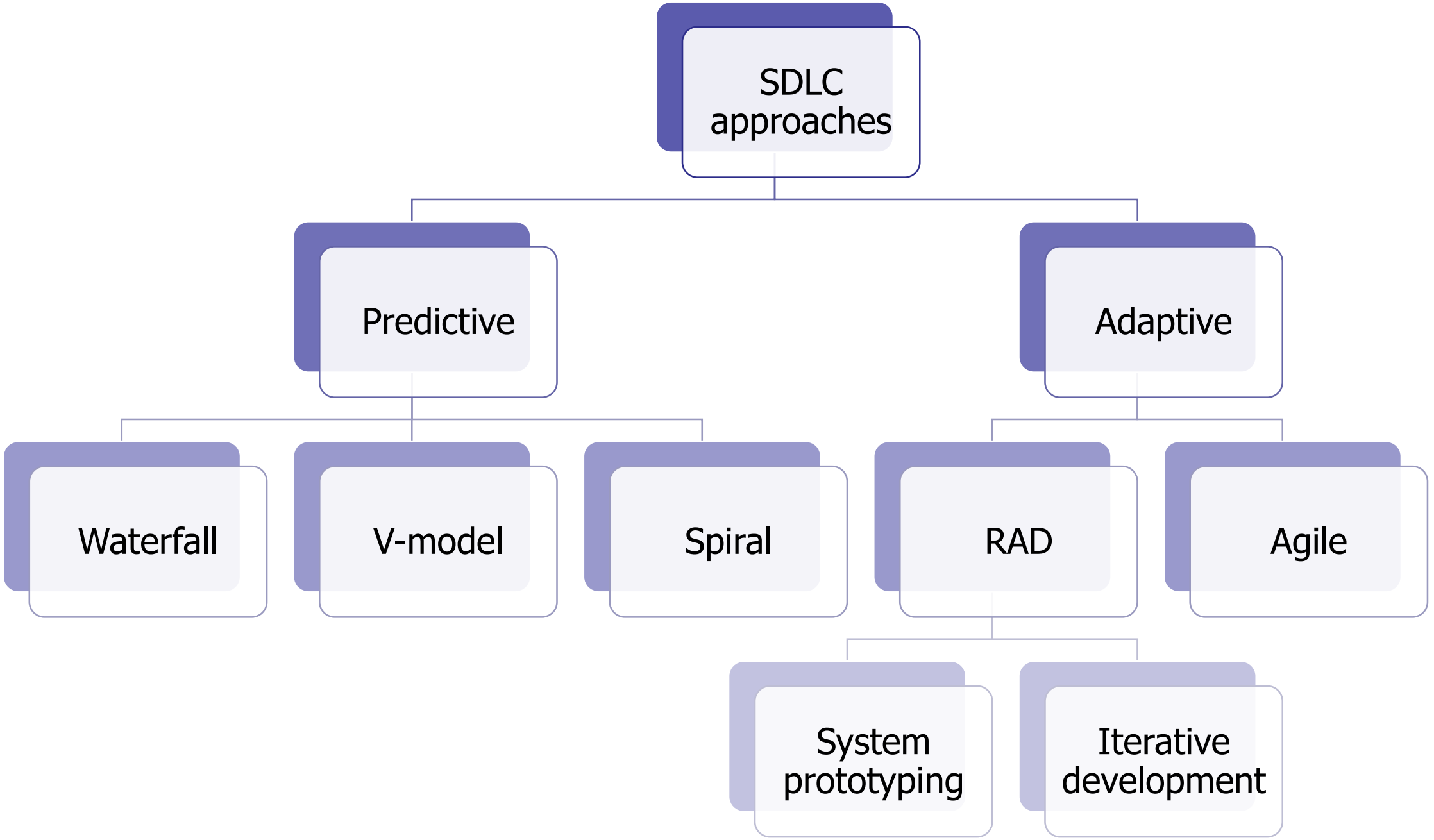
- Initial development of a new information system is usually done as a project.
- A project is a sequence of tasks that has a beginning and an end and produces some end result. This means that the activities required to develop a new system are identified, planned, organized, and monitored.

System Development Life Cycle (SDLC)

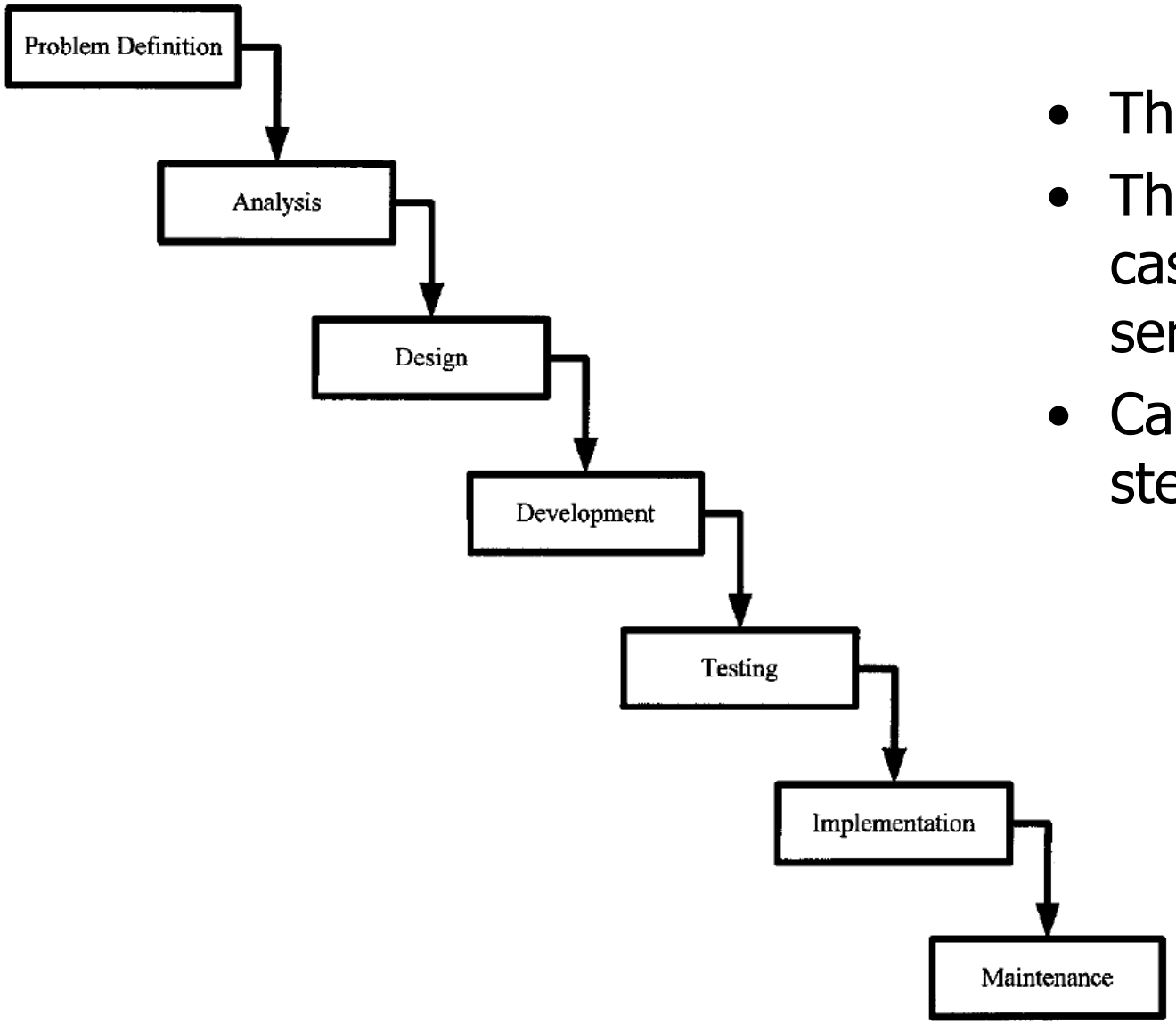
- To manage a project with analysis, design, and other development activities, you need a project management framework to guide and coordinate the work of the project team.
- The system development life cycle (SDLC) is a framework that identifies all the activities required to research, build, deploy, and often maintain an information system.

SDLC core processes




SDLC approaches
Predictive
Adaptive
Waterfall
V-model
Spiral
RAD
Agile
System prototyping
Iterative development

Waterfall model



- This model may be called SDLC.
- The model visually suggests work cascading from step to step like a series of waterfalls.
- Cannot go back to the previous step.

Waterfall Model - Application

- Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors
- Some situations where the use of the Waterfall model is most appropriate are:
  - Requirements are very well documented, clear and fixed.
  - Product definition is stable.
  - Technology is understood and is not dynamic.
  - There are no ambiguous requirements.
  - Crucial systems

Waterfall Model - Advantages

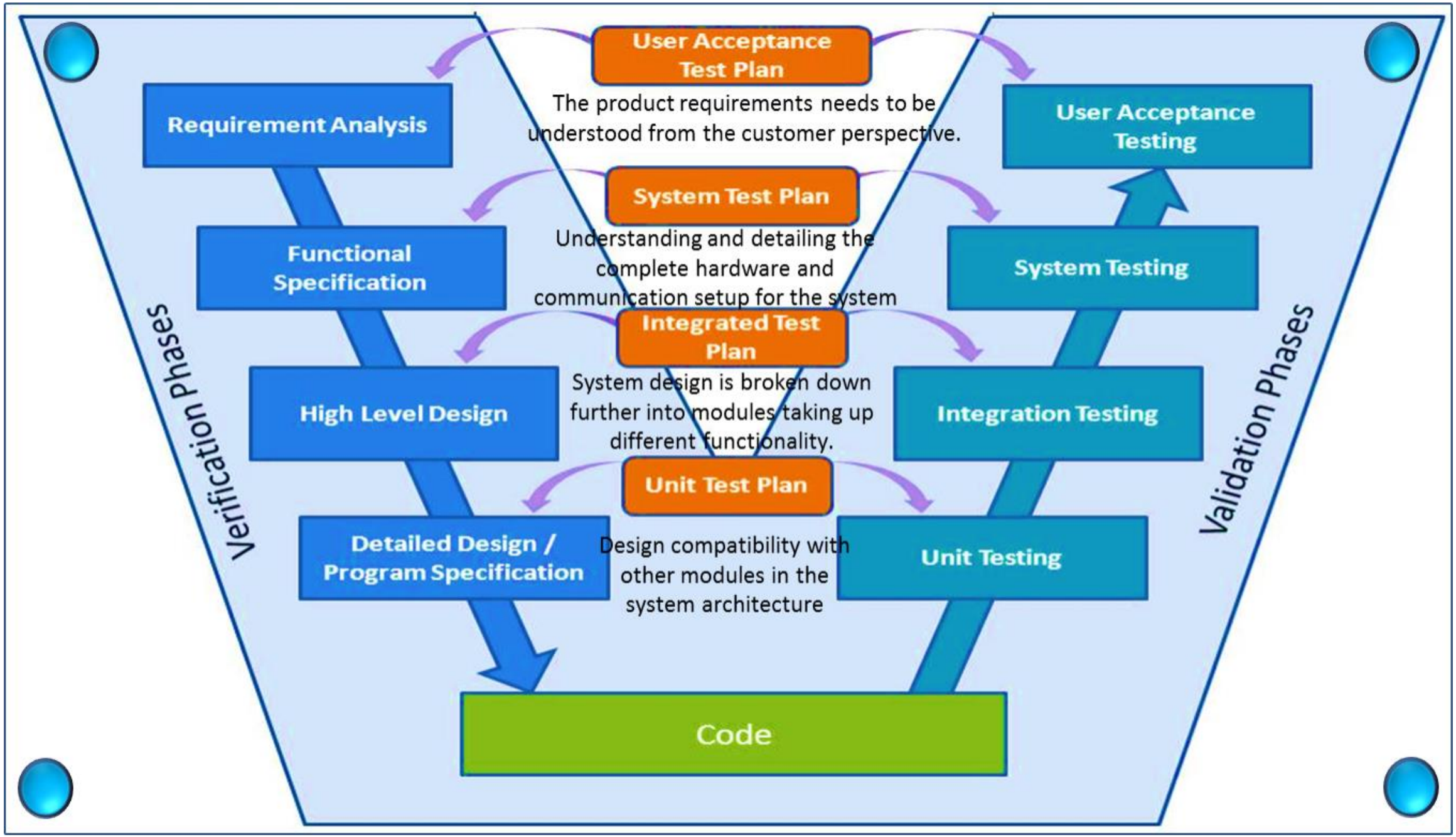
- It allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one. Each phase of development proceeds in strict order.
- Simple and easy to understand and manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Clearly defined stages and well-understood milestones.
- Process and results are well documented.

Waterfall Model - Disadvantages

- It does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex projects where requirements are at a moderate to high risk of changing.
- Adjusting scope during the life cycle can end a project.

V-model

- Execution of processes happens in a sequential manner in a V-shape.
- It is also known as Verification and Validation model.
- The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase.
- This is a highly-disciplined model and the next phase starts only after the completion of the previous phase.


User Acceptance Test Plan
The product requirements needs to be understood from the customer perspective.
Requirement Analysis
User Acceptance Testing
System Test Plan
Understanding and detailing the complete hardware and communication setup for the system
Functional Specification
System Testing
Integrated Test Plan
System design is broken down further into modules taking up different functionality.
High Level Design
Integration Testing
Unit Test Plan
Design compatibility with other modules in the system architecture
Detailed Design / Program Specification
Unit Testing
Code
Verification Phases
Validation Phases

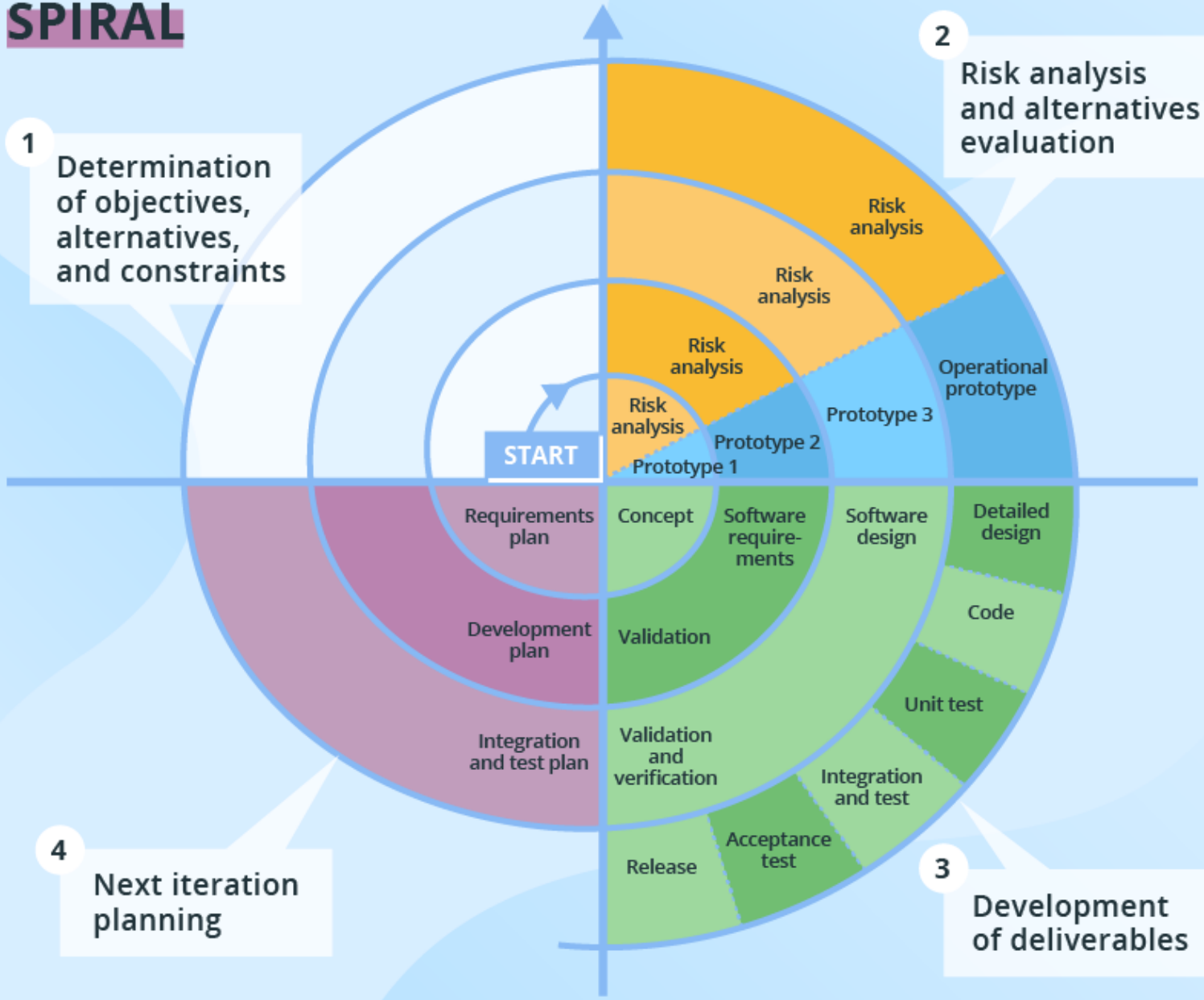
Spiral Model

- The spiral model combines the idea of incremental development with the systematic, controlled aspects of the waterfall model. Core components are developed first and then additional components are added. It is also called iterative because the six core development processes are repeated for each component.
- Very high emphasis on risk analysis.
- It allows incremental releases of the product or incremental refinement through each iteration around the spiral.
- The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

SPIRAL

1 Determination of objectives, alternatives, and constraints

2 Risk analysis and alternatives evaluation



3 Development of deliverables

4 Next iteration planning

Spiral Model – Pros and cons

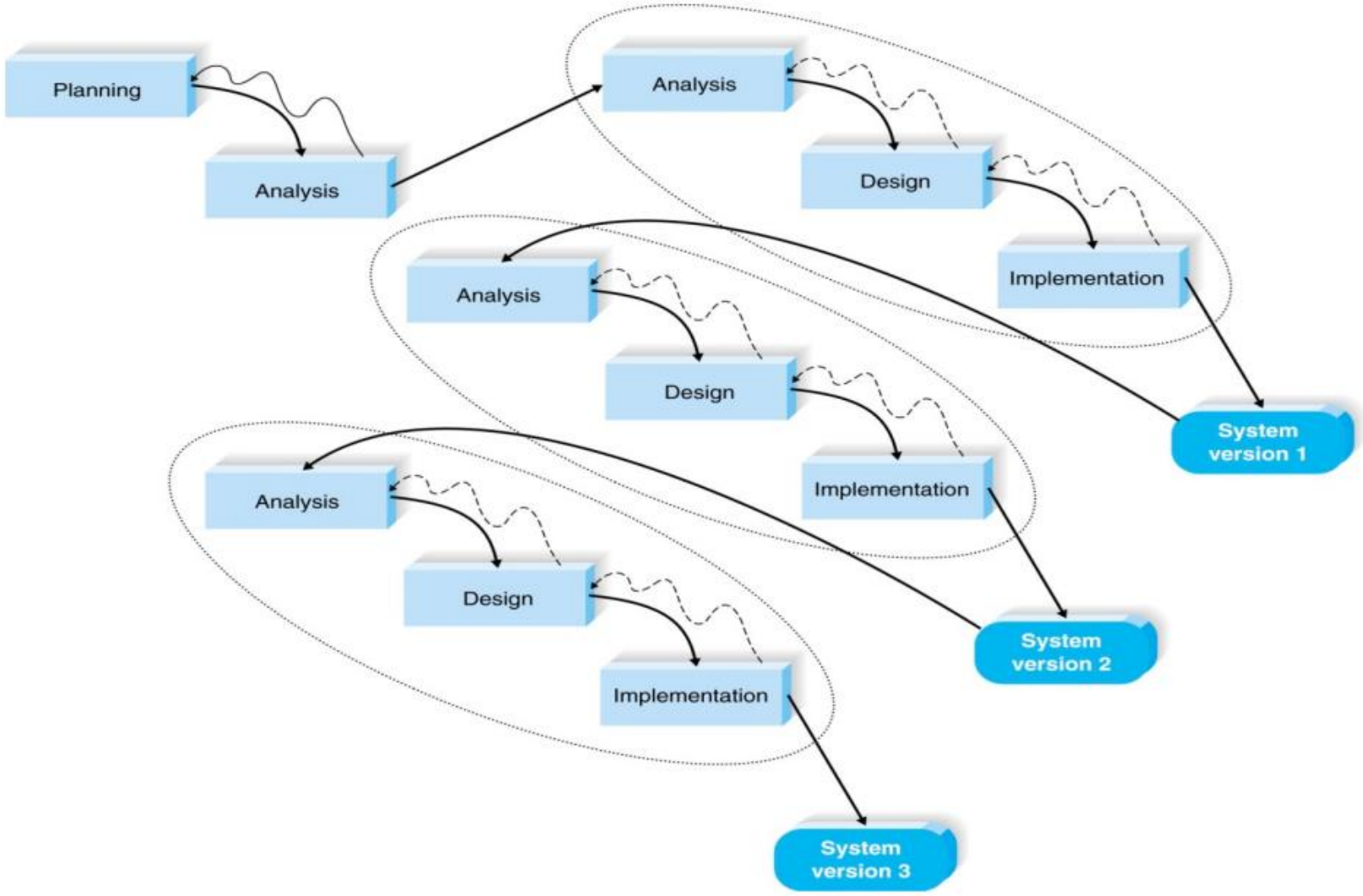
+ Changing requirements can be accommodated.
+ Allows extensive use of prototypes.
+ Requirements can be captured more accurately.
+ Users see the system early.
+ Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

- Management and process are more complex.
- End of the project may not be known early.
- Not suitable for small or low-risk projects and could be expensive for small projects.
- Spiral may go on indefinitely.

Rapid Application Development (RAD) Model

- The RAD model is based on prototyping and iterative development with no specific planning involved. It uses minimal planning in favor of rapid prototyping. The process of writing the software itself involves the planning required for developing the product.
- A prototype is a model that is functionally equivalent to a component of the product.
- Rapid Application Development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using the iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.
- RAD model should be chosen only if domain experts are available with relevant business knowledge.
- Should be used where the requirements change during the project and working prototypes are to be presented to the customer in small iterations of 2-3 months.

RAD- Iterative development



RAD- Iterative development (continue…)

- Iterative means that the same functionality is refined incrementally
  - the same feature or group of features are further refined based on feedback and inputs from users.
- It starts with some of the software specifications and develops the first version of the software. After the first version if there is a need to change the software, then the new version of the software is created with a new iteration. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through a repeated cycle are is smaller portions at a time.

RAD – Pros and cons

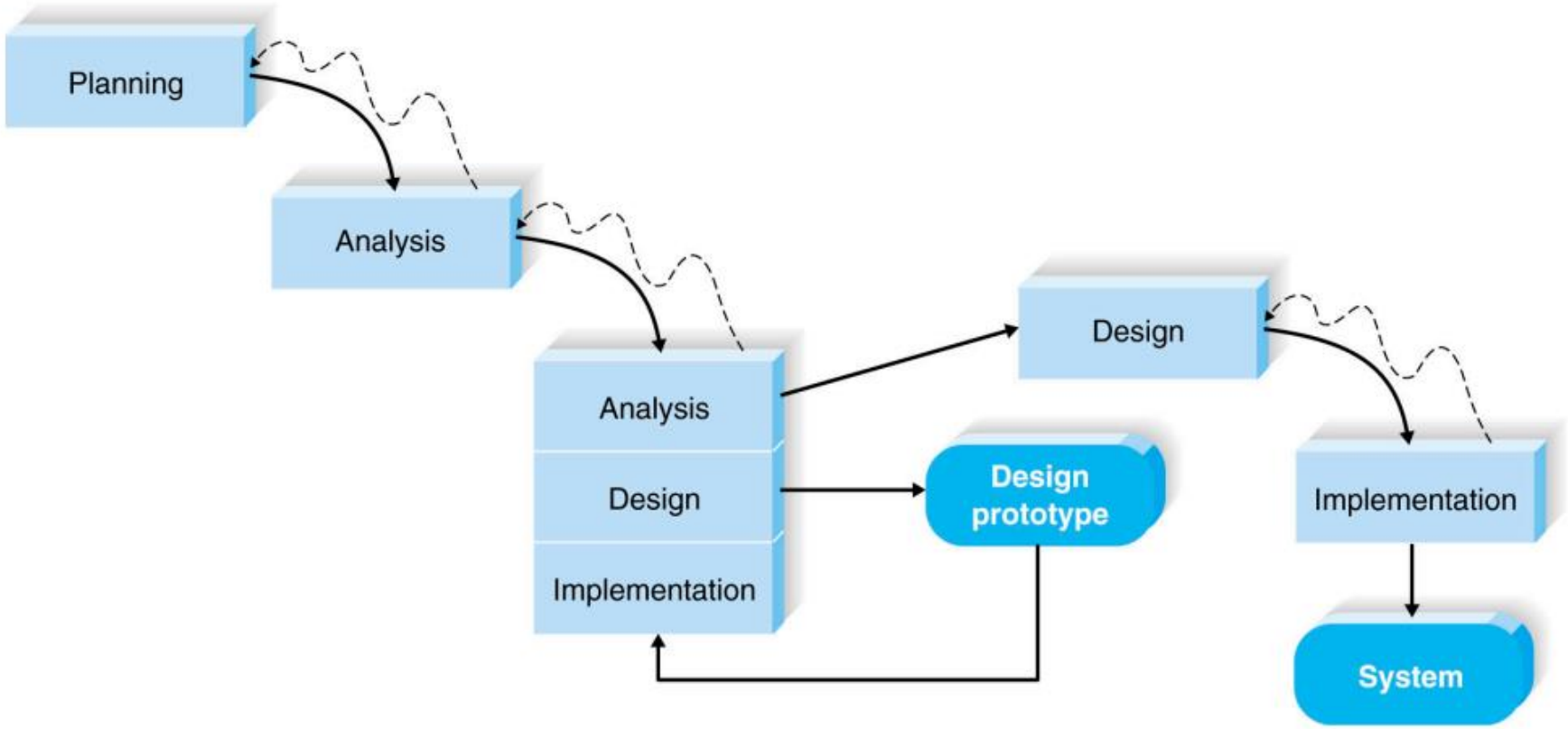
+ Changing requirements can be accommodated.
+ Progress can be measured.
+ Reduced development time.
+ Increases reusability of components.
+ Quick initial reviews occur through customer feedback.
+ Due to code generators and code reuse, there is a reduction in manual coding.

- Dependency on technically strong team members for identifying business requirements.
- Only a system that can be modularized can be built using RAD.
- Requires highly skilled designers for modeling.
- Inapplicable to cheaper projects as the cost of modeling and automated code generation is very high.
- Requires user involvement throughout the life cycle.

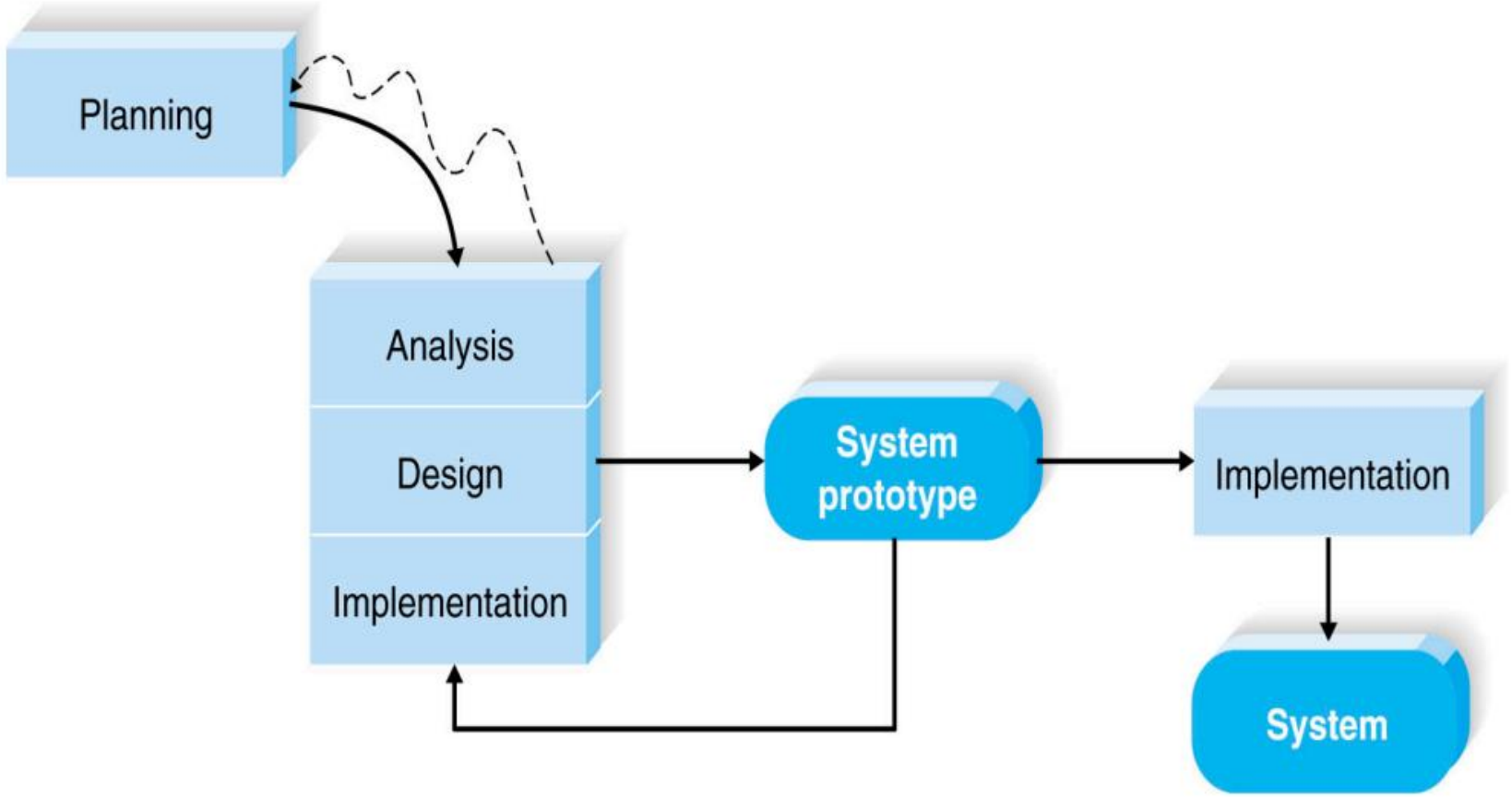
RAD- System prototyping

- System Prototyping refers to building software application prototypes which displays the functionality of the product under development, but may not actually hold the exact logic of the original software.
- Prototyping is used to allow the users evaluate developer proposals and try them out before implementation.
- There are different types of software prototypes used in the industry. Following are the two major software prototyping types used widely:
  - Throwaway/Rapid Prototyping
  - Evolutionary Prototyping

System prototyping – Throwaway prototyping



System prototyping – Evolutionary prototyping



Software prototyping – Pros and cons

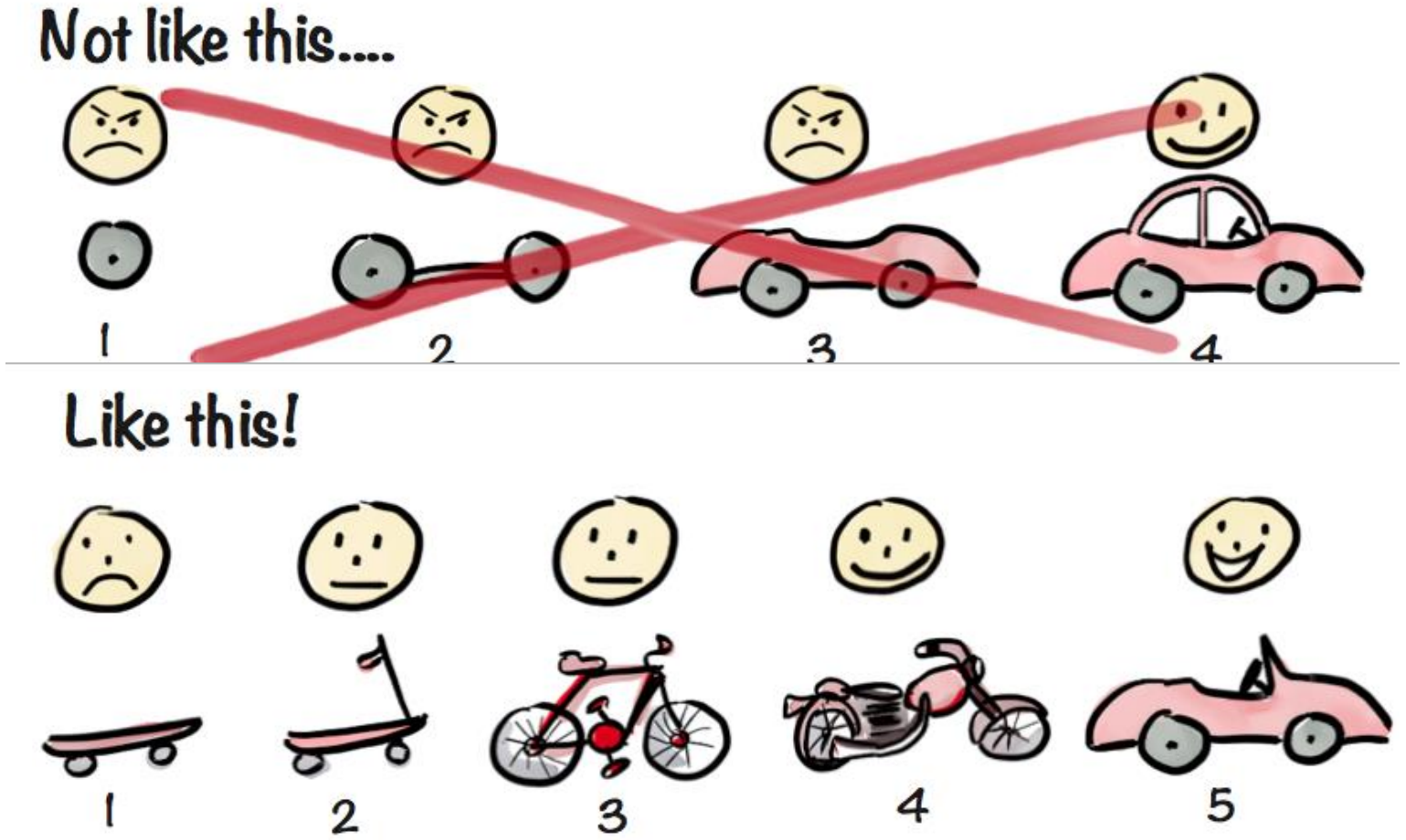
+ Increased user involvement in the product even before its implementation.
+ Reduced time and cost as the defects can be detected much earlier.
+ Missing functionality can be identified easily.

- Risk of insufficient requirement analysis owing to too much dependency on the prototype.
- Users may get confused in the prototypes and actual systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- The effort invested in building prototypes may be too much if it is not monitored properly.

SDLC - Agile Model

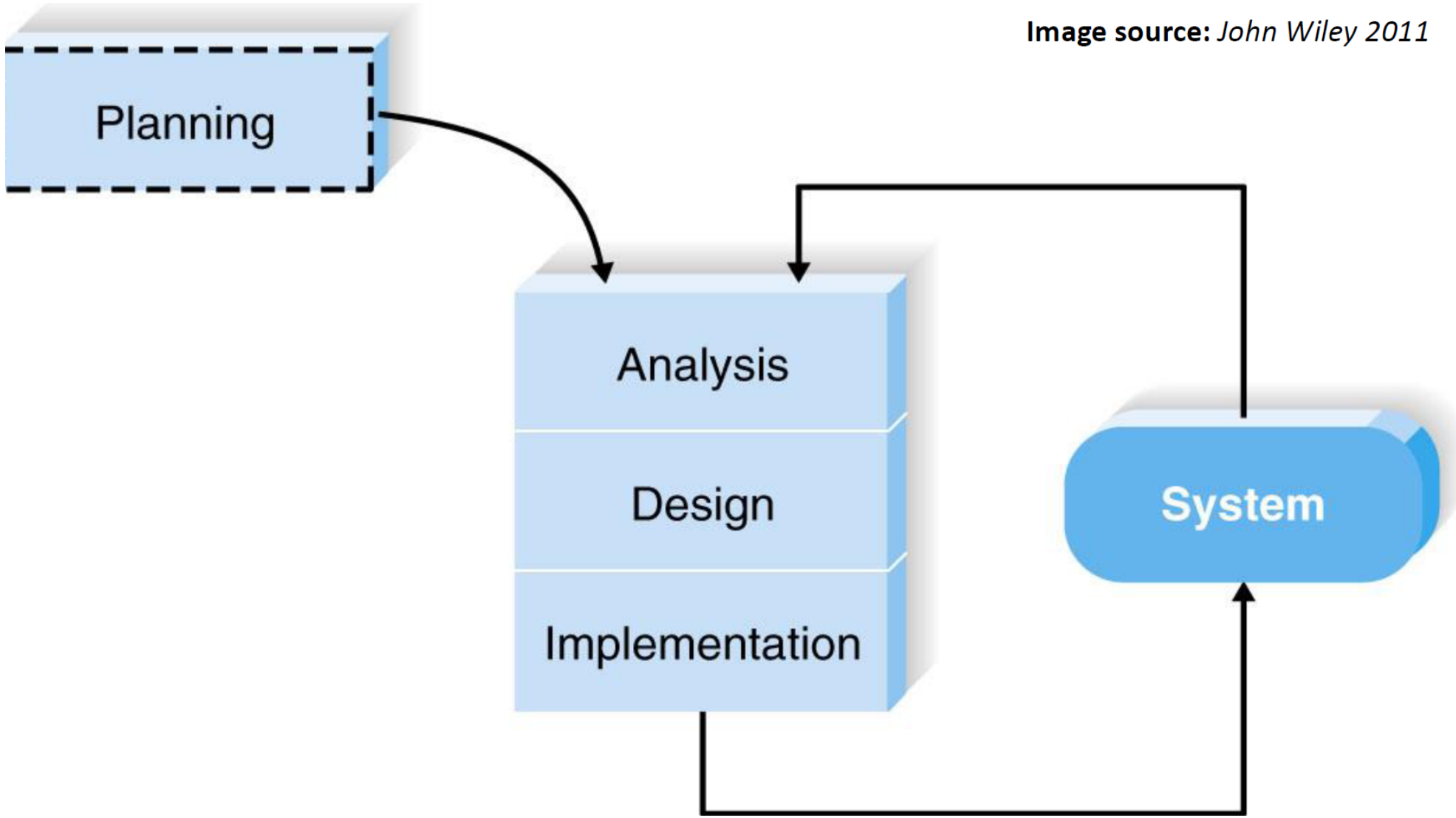
- A combination of iterative and incremental process models with a focus on process adaptability and customer satisfaction by rapid delivery of working software products. Agile Methods break the product into small incremental builds. These builds are provided in iterations.
- Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements.
- In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release. Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.
- Includes face-to-face communication.

The Agile bicycle



Henrik Kniberg

Image source: *John Wiley 2011*



Agile – Advantages

- Is a very realistic approach to software development.
- Promotes teamwork and cross-training.
- Functionality can be developed rapidly and demonstrated.
- Suitable for fixed or changing requirements.
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Little or no planning required.

Agile - Disadvantages

- Not suitable for handling complex dependencies.
- An overall plan, an agile leader, and agile PM practice are a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if the customer is not clear, the team can be driven in the wrong direction
- There is a very high individual dependency since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

Discussion

- Please read Question 1 and discuss in groups of 2 and 3.