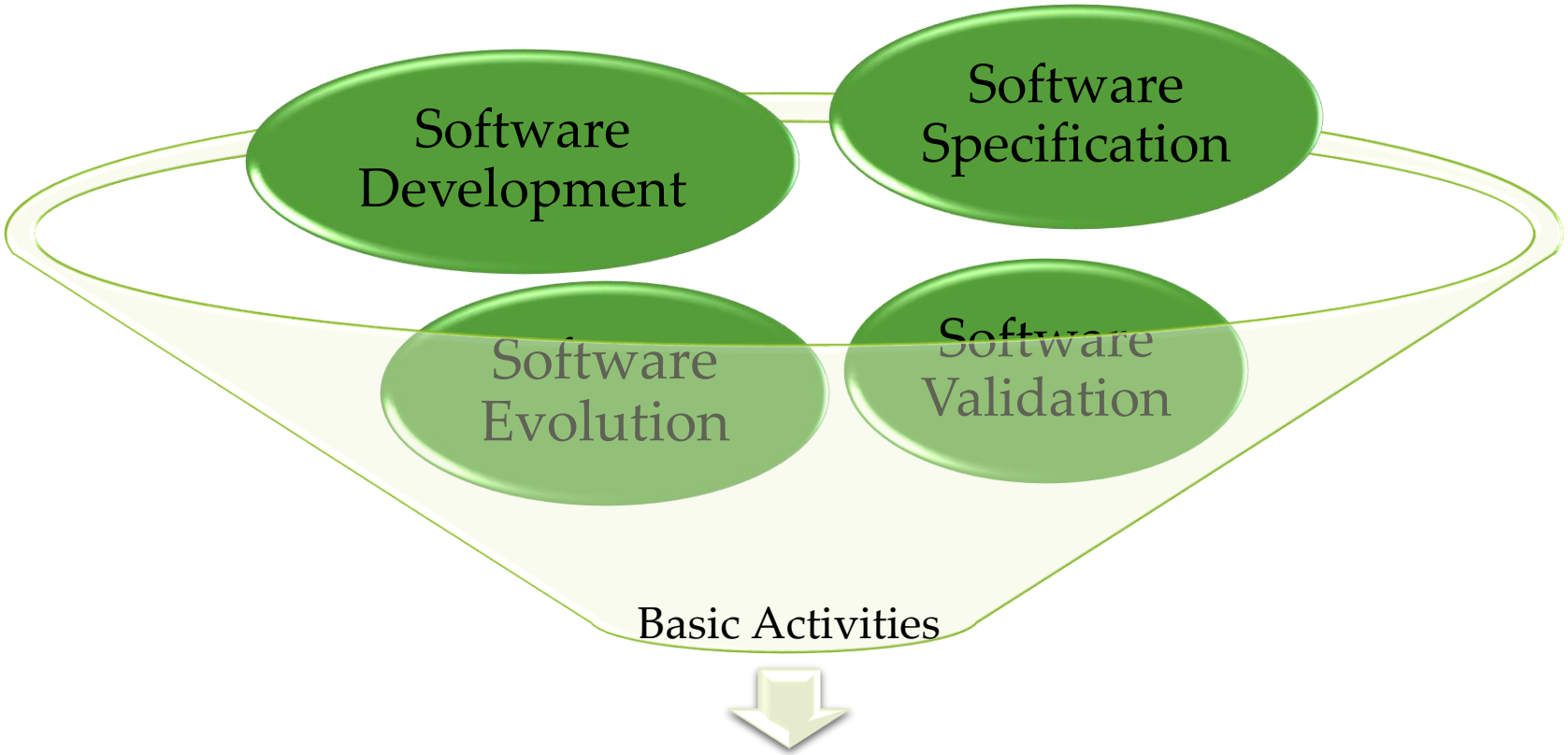
Chapter 2
Software Process Models

Lesson Objectives

- Describe and evaluate various software process models
- Suggest which process models are appropriate under a given situation of software development

Software Process

Activities to develop software



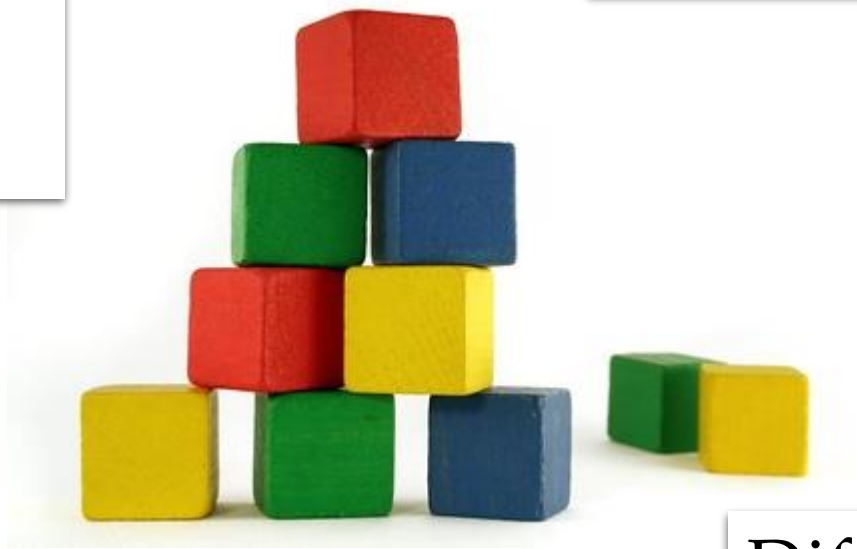
Software Process

- Software process consists of activities, which are involved in developing software products.
- Basic activities are:-
  - ***Software Specification***
  - ***Software Development***
  - ***Software Validation***
  - ***Software Evolution***

Software Process Model

Including:
- ✓SW processes
- ✓Roles
- ✓Products

Simplified description of software processes



Different perspectives

Software Process Model

- Software process model is a simplified description of a software process which is presented from a particular perspective.
- May include activities which are part of the software process, software products and the roles of people involved in software engineering

Software Process Model

Linear Sequential Model/Waterfall Model

Prototyping Model

Rapid Application Model

Evolutionary Model

- Incremental Model
- Spiral Model

Component-based Development Model

Agile Software Development

- Extreme Programming

1. Linear Sequential Model/Waterfall Model

1. Linear Sequential Model/Waterfall Model

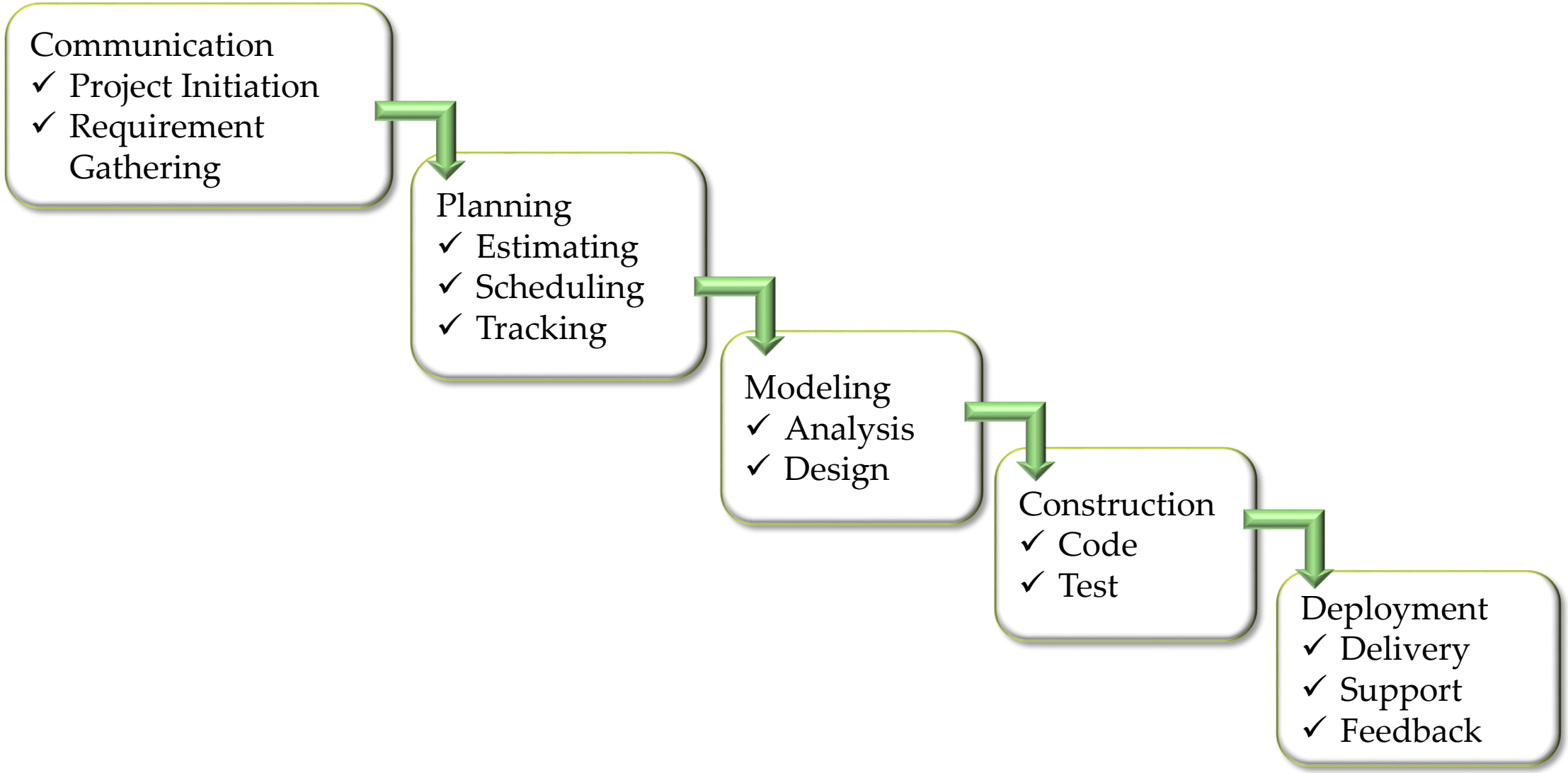
- Commonly known as classic life cycle or the waterfall model

Hardware, people, database

At strategic business level

- Activities include:-
  - System/information engineering and modeling
  - Software requirements analysis
  - Design
  - Code Generation
  - Testing
  - Support

1. Linear Sequential Model/Waterfall Model



1. Linear Sequential Model/Waterfall Model

1. Systematic

2. Easy to follow

Advantages

1. Linear Sequential Model/Waterfall Model

What are the drawbacks of this model?

1. "Blocking state" situation - Real projects rarely follow linear flow
2. Difficult for customers to state requirement explicitly
3. Working version of software unavailable until later stage

1. Linear Sequential Model/Waterfall Model

- When to use Waterfall approach?

2. Prototyping Model

2. Prototyping Model

- Software customers and end-users usually find it very difficult to express their real requirements
- A system prototype can be developed to give end-users a concrete impression of the system capabilities

2. Prototyping Model

- It may therefore help in establishing and validating system requirements and allows the users to experiment with requirements and to see how the system supports their work
- It's a technique of risk reduction (Boehm et al., 1984)

2. Prototyping Model

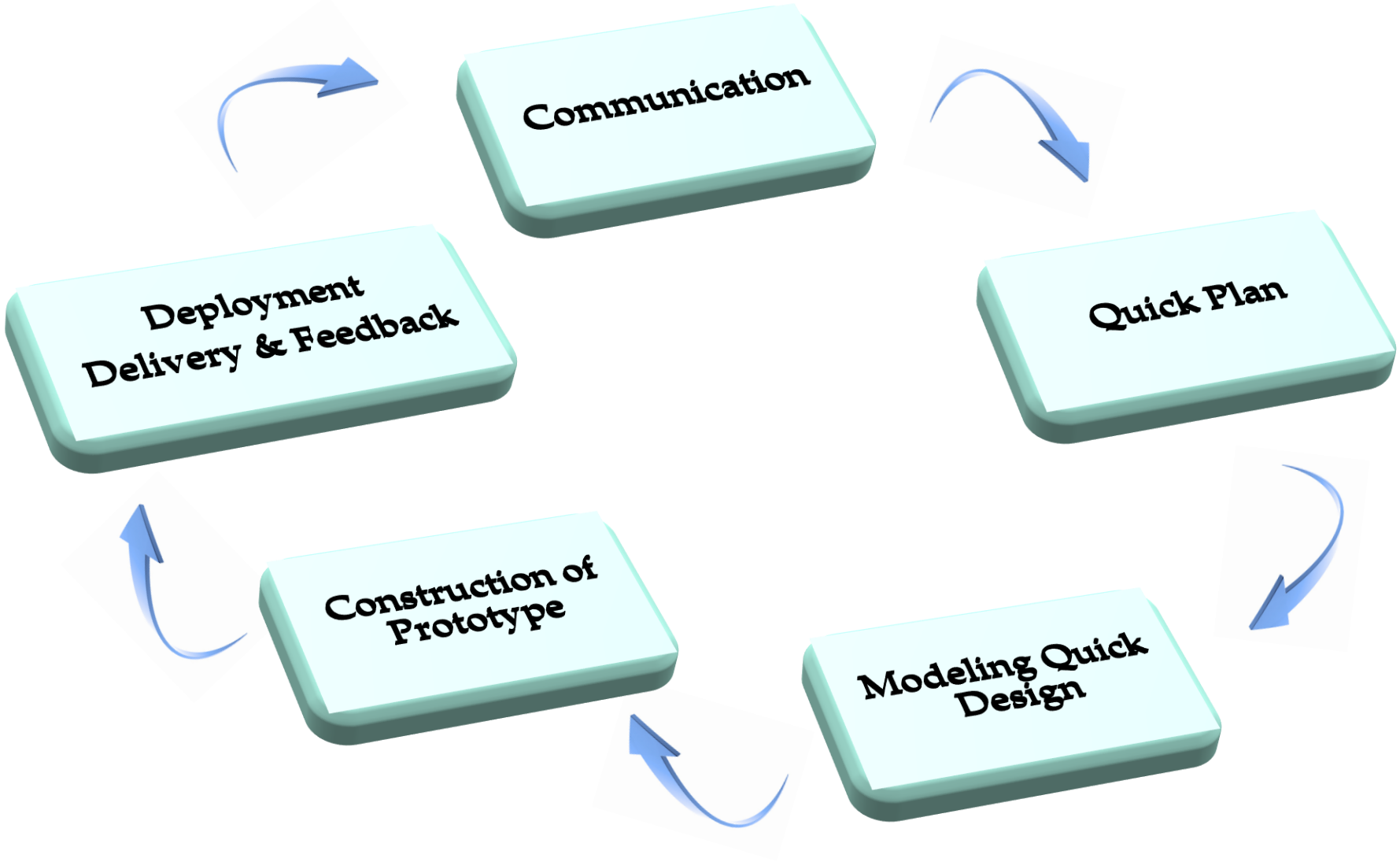
Customer test drives mock-up

Listen to customer

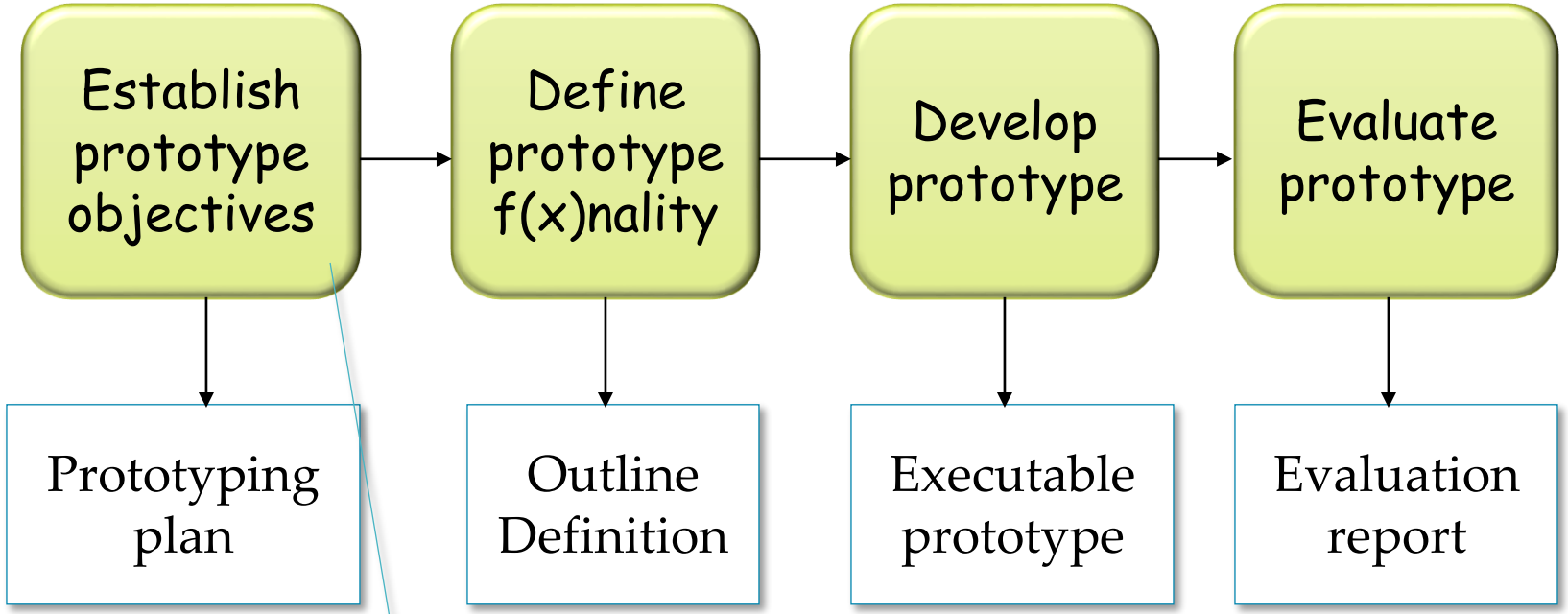
Activities

Build/revise mock-up

2. Prototyping Model



2. Prototyping Model



Is it to:

- Prototype UI
- validate functional requirement
- show feasibility of application

2. Prototyping Model – How to create?

- Prototyping techniques include the use of the following: -
  - Dynamic high-level language development
  - Fourth-generation languages – e.g. SQL
  - Prototype construction from reusable components
  - Visual programming language
  - Internet-based prototyping

Throwaway vs Evolutionary Prototyping

- Prototyping in the software process may involve :
  - ***'Throw-away' prototyping*** in which a prototype is developed to understand the system requirements, or
  - ***Evolutionary prototyping*** in which a prototype evolves through a number of versions to the final system

Throwaway vs Evolutionary Prototyping

THROWAWAY

validate or derive the system requirements discarded

EVOLUTIONARY

Deliver a working system

Problems:

- Structure is corrupted

Throwaway Prototyping

- Prototyping in the software process may involve :
- ***'Throw-away'*** The objective of throw-away prototyping is to *validate* or *derive the system requirements*

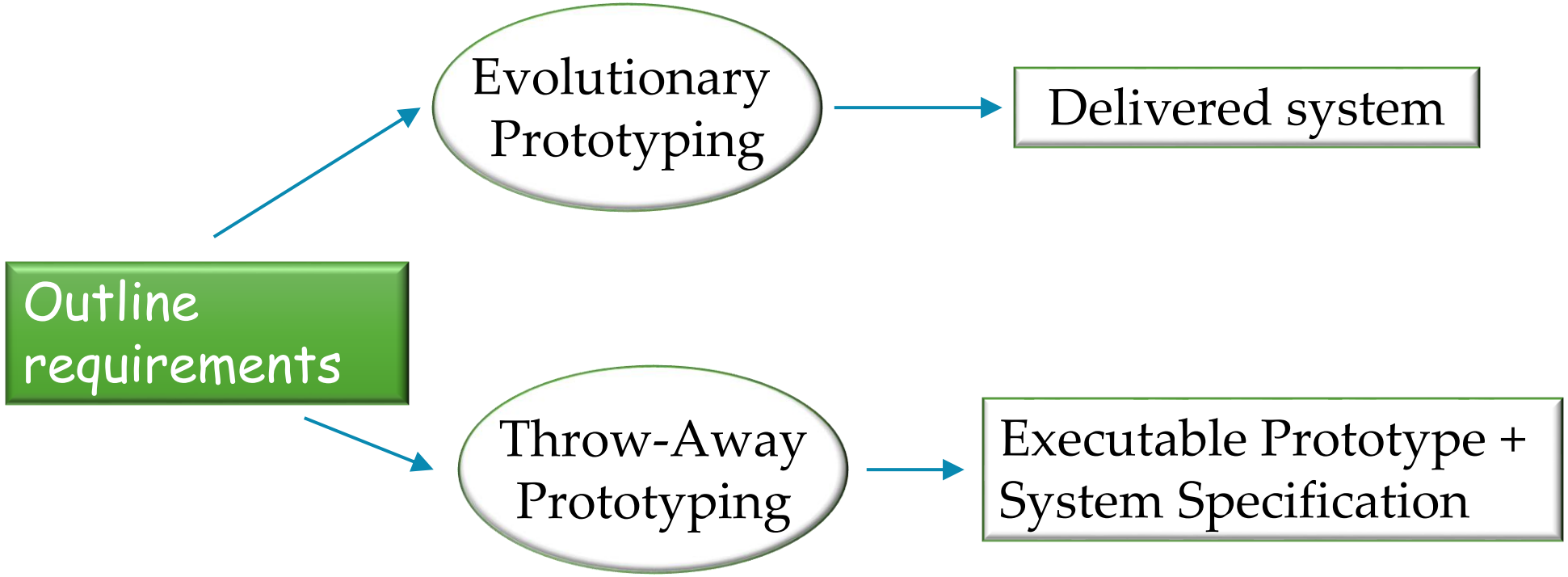
- It will be discarded once the requirement specs are specified
  - ***prototyping*** in which a prototype is developed to understand the system requirements, or
  - ***Evolutionary prototyping*** in which a prototype evolves through a number of versions to the final system

Evolutionary Prototyping

- The objective of evolutionary prototyping is to **deliver a working system** to end-users

- The principal problem with evolutionary prototyping is that the system **structure becomes corrupted** by constant change. Further changes to the system become increasingly difficult to make

Throwaway vs Evolutionary Prototyping



Throwaway vs Evolutionary Prototyping

Throw-Away Prototyping

Develop the parts you understand LEAST

Throwaway vs Evolutionary Prototyping

Evolutionary Prototyping

Develop the parts you understand BEST

WHY?

- ✓ Build very robust prototype in a structured manner & constantly refine it.
- ✓ Minimize risk

Benefits of Prototyping

- ***Misunderstandings*** between software developers and users may be identified as the system functions are demonstrated
- ***Missing*** user services may be detected
- Software development staff may find ***incomplete*** and/or ***inconsistent requirements*** as the prototype is developed

Benefits of Prototyping

- ***Difficult-to-use or confusing*** user services may be identified and refined
- A working system, although incomplete is available ***quickly to demonstrate*** the feasibility and usefulness of the application to management
- The prototype serves as a ***basis*** for ***writing the specification*** for a production quality system

Ince and Hekmatpour (1987) claim that prototype helps in:

1. User training
2. System testing – 'back-to-back' tests

Drawbacks of Prototyping

- Planning, costing and estimating a prototyping project is ***outside the experience*** of many software project managers
- Procedures for change and configuration management may be ***unsuitable for controlling the rapid change*** inherent in prototyping. However, if there are no change management procedures, evaluation is difficult.
- Managers may exert pressure on prototype evaluators to reach swift conclusions about the prototype. These may result in ***inappropriate requirements.***

Drawbacks of Prototyping

- Customers demand "few fixes" to it to become the working version → low quality

- Developers initially used familiar OS, algorithm, etc. which are not appropriate but to demonstrate system capability only. But in the end, forgot the reasons why they were inappropriate: hence did not use the better OS, algorithm, etc to develop end product.

Discussion

- Common argument against prototyping is the cost may be too high. Why not '***build it cheap and fix it later' i.e. modify a finished system to meet unperceived needs***?
- Do you agree with such notion?

2. Prototyping Model

When to use prototyping approach?

1. Developers unsure of the efficiency of an algorithm
2. Developers unsure of the human/machine interaction that should take place

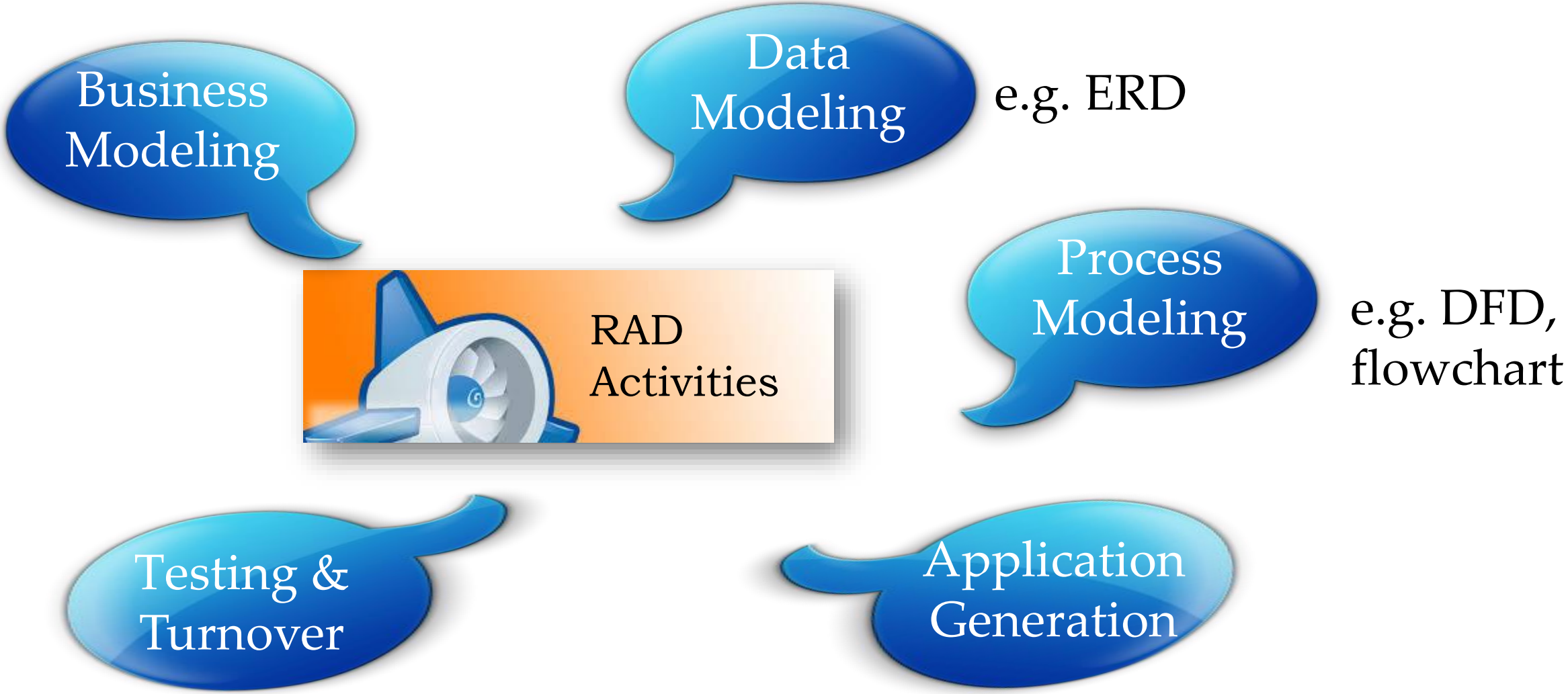
3. Rapid Application Development (RAD) Model

3. The RAD Model

- Rapid Application Development

- Which is an incremental software development process model that emphasizes an extremely short development cycle.

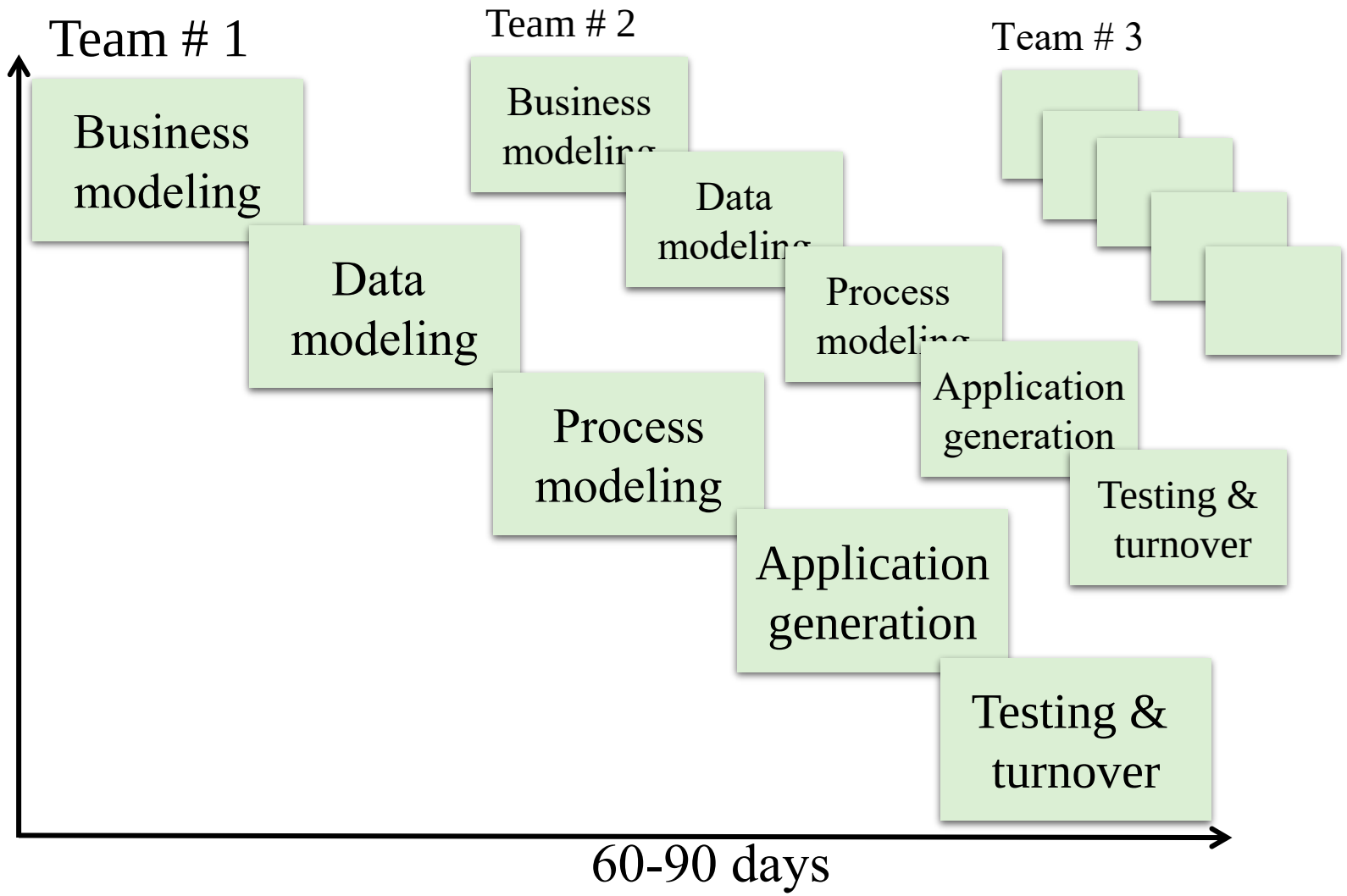
3. The RAD Model



3. The RAD Model

- Activities include:
  - Business modeling : get to know the business procedures
  - Data modeling
  - Process modeling
  - Application generation: use 4th GL & reusable components
  - Testing and turnover

3. The RAD Model



3. The RAD Model

User requirements are well-known

Project scope is well defined

High user involvement

When to use?

Exercise

	Waterfall	Prototyping	RAD
Staff			
Development time			
User involvement			
User Requirements			
Release			

Evolutionary Software Process Model

Evolutionary Software Process Model

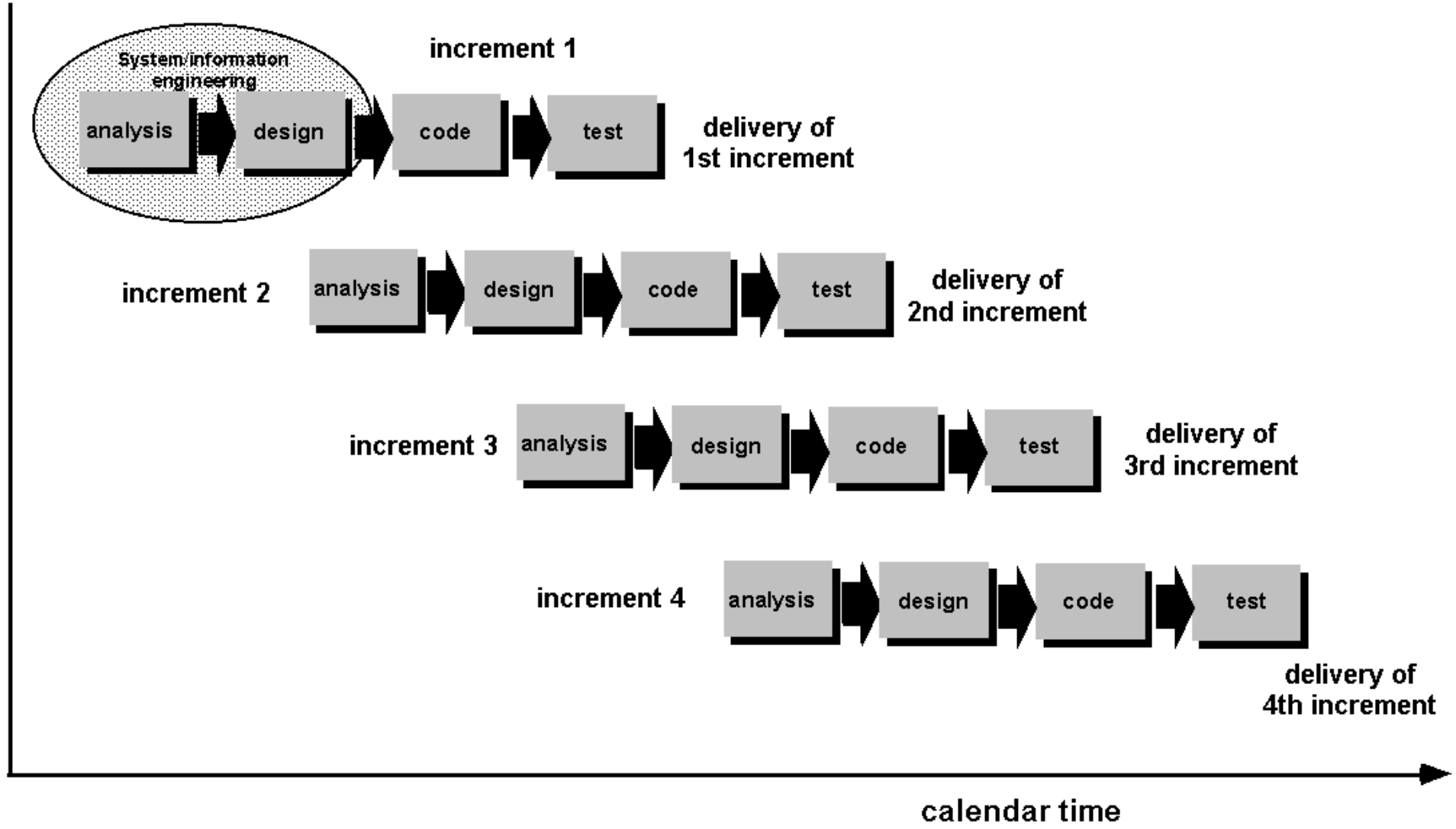
- It is based on the idea of developing an initial implementation, exposing this to user comment and refining this through many versions until an adequate system has been developed.

- These models are iterative. They are characterized in a manner that enables software engineers to develop increasingly more complete versions of the software.

- Two common examples:
  - Incremental Model
  - Spiral Model

4. Evolutionary Software Process Model: Incremental Model

4. Evolutionary Software Process Model: Incremental Model



4. Evolutionary Software Process Model: Incremental Model

- How it works?
- Deliver the core product function(s)/feature(s) at the first increment/version and deliver the supplementary product function(s)/feature(s) in next increment(s)/version(s).

4. Evolutionary Software Process Model: Incremental Model

What are the advantages of this model?

1. Particularly useful if staff unavailable.

2. Can be planned to manage technical risks.
   - E.g. If the required h/w is not available at the early stage, plan not to use it in the early increment. So, able to deliver partial functionality to customer without delay!

4. Evolutionary Software Process Model: Incremental Model

What is the main difference between incremental model and RAD model?

1. RAD needs more resources to create several teams.

2. Incremental model delivers software product version by version

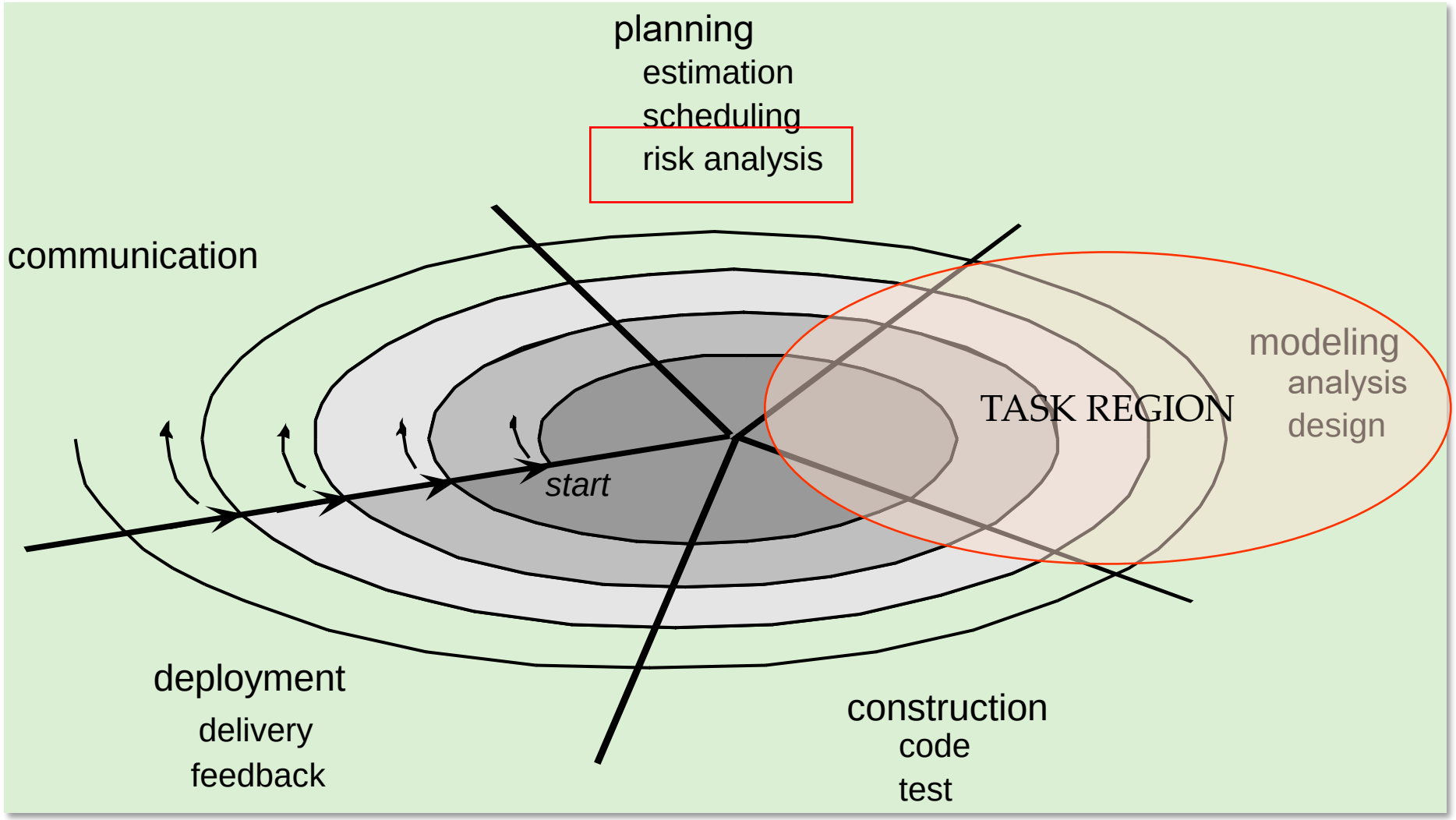
5. Evolutionary Software Process Model: Spiral Model

5. Evolutionary Software Process Model: Spiral Model

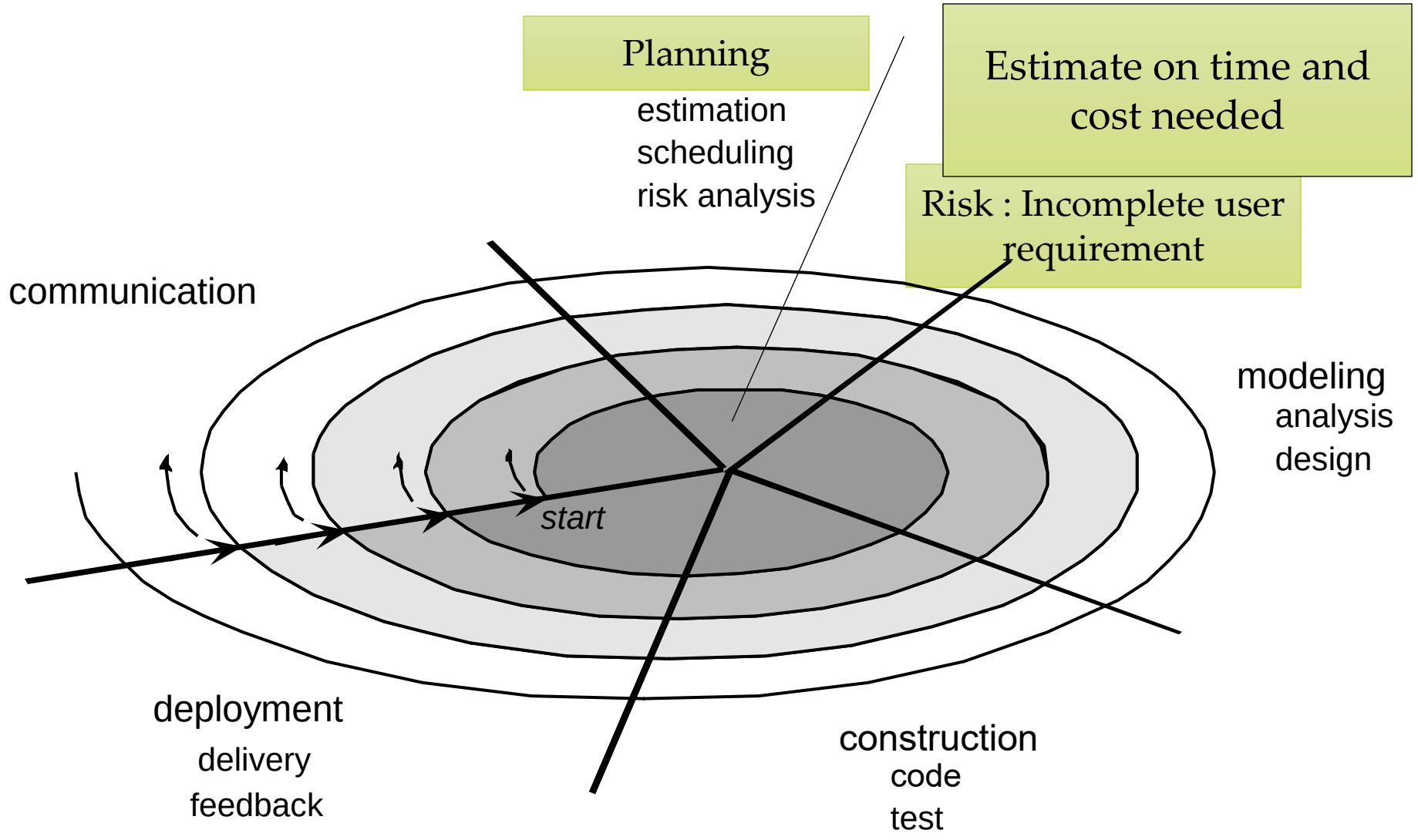
- Divided into a number of framework activities, called task regions.

- Software engineering team moves around the spiral, beginning at the center.

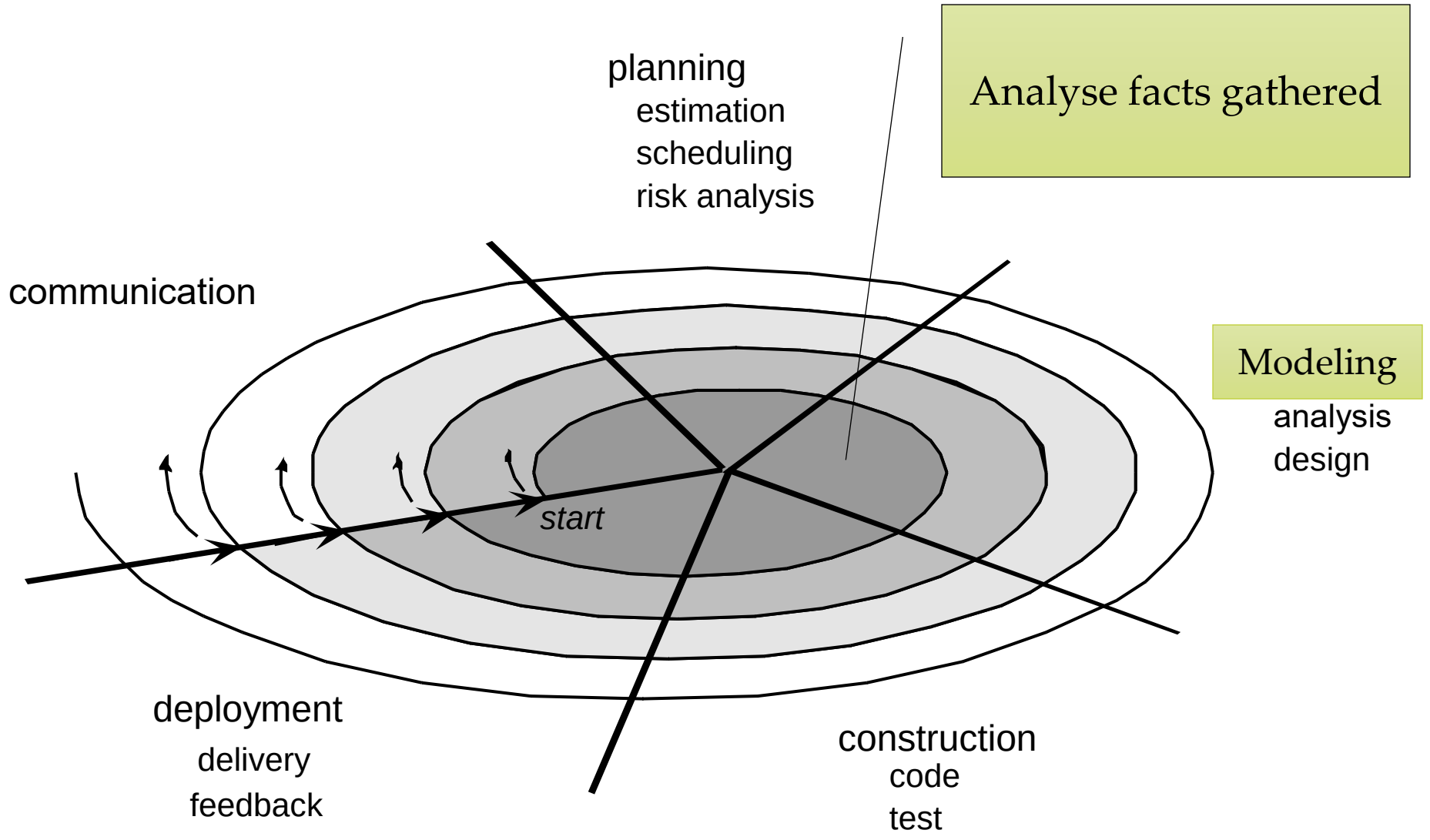
5. Evolutionary Software Process Model: Spiral Model



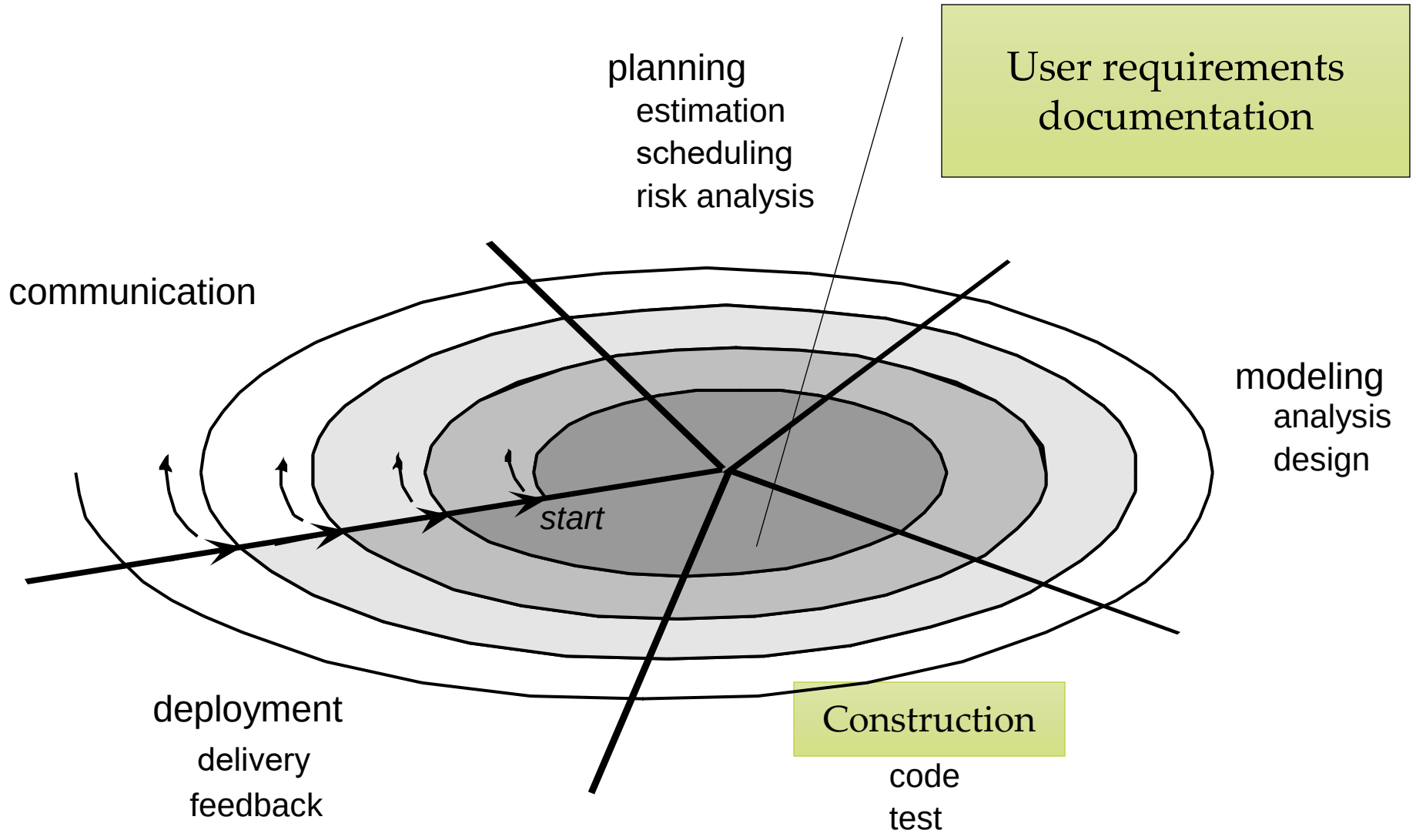
5. Evolutionary Software Process Model: Spiral Model Example



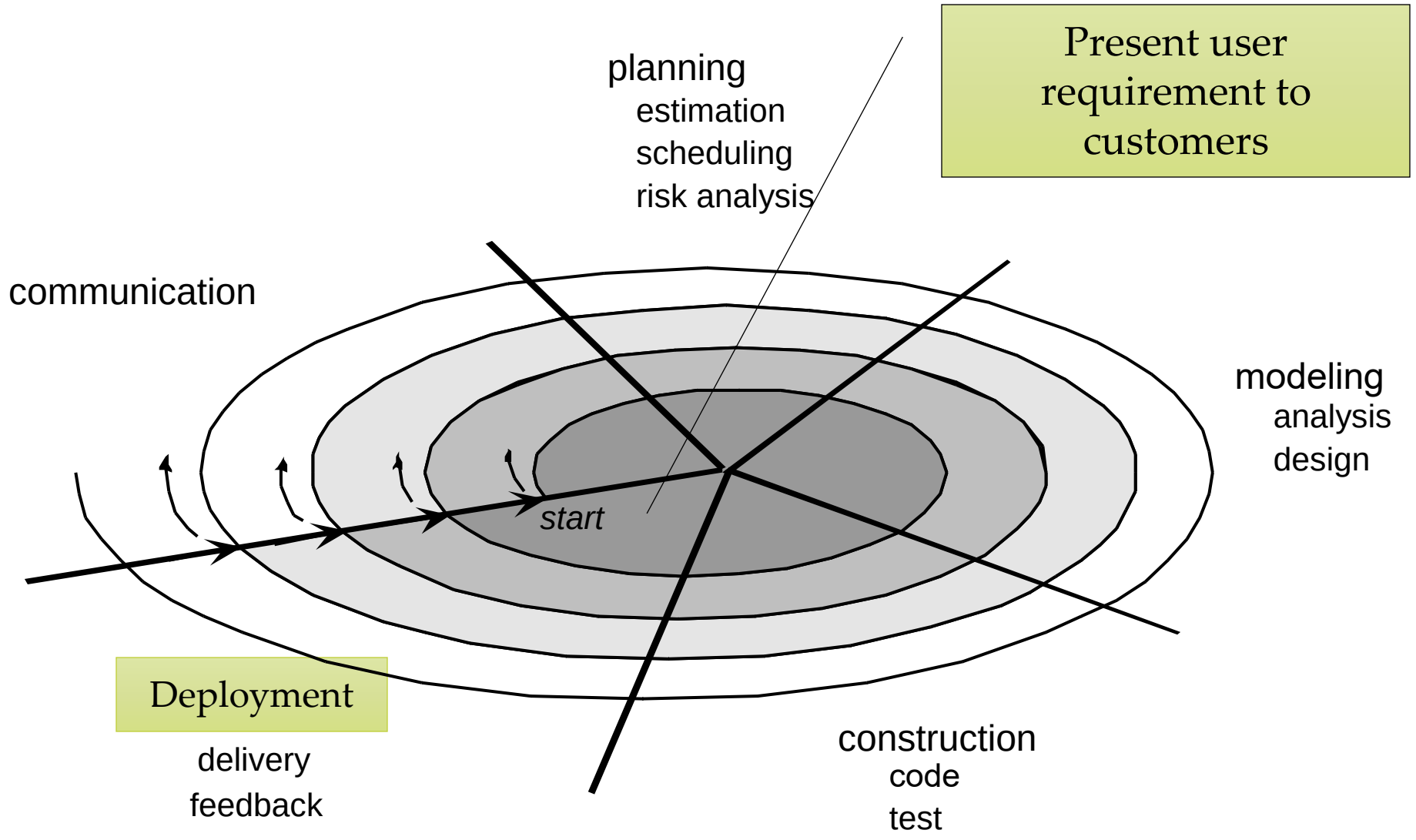
5. Evolutionary Software Process Model: Spiral Model Example



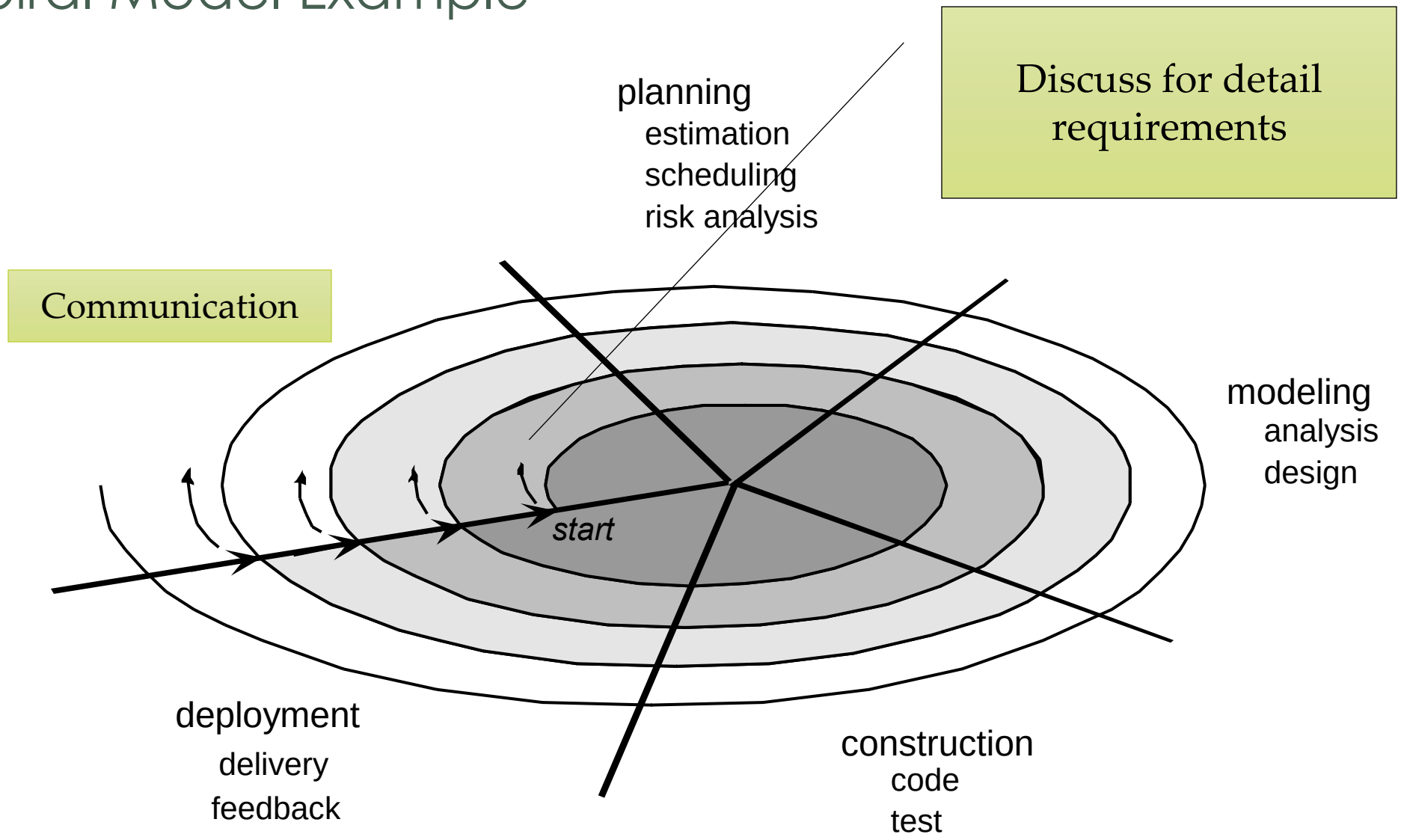
5. Evolutionary Software Process Model: Spiral Model Example



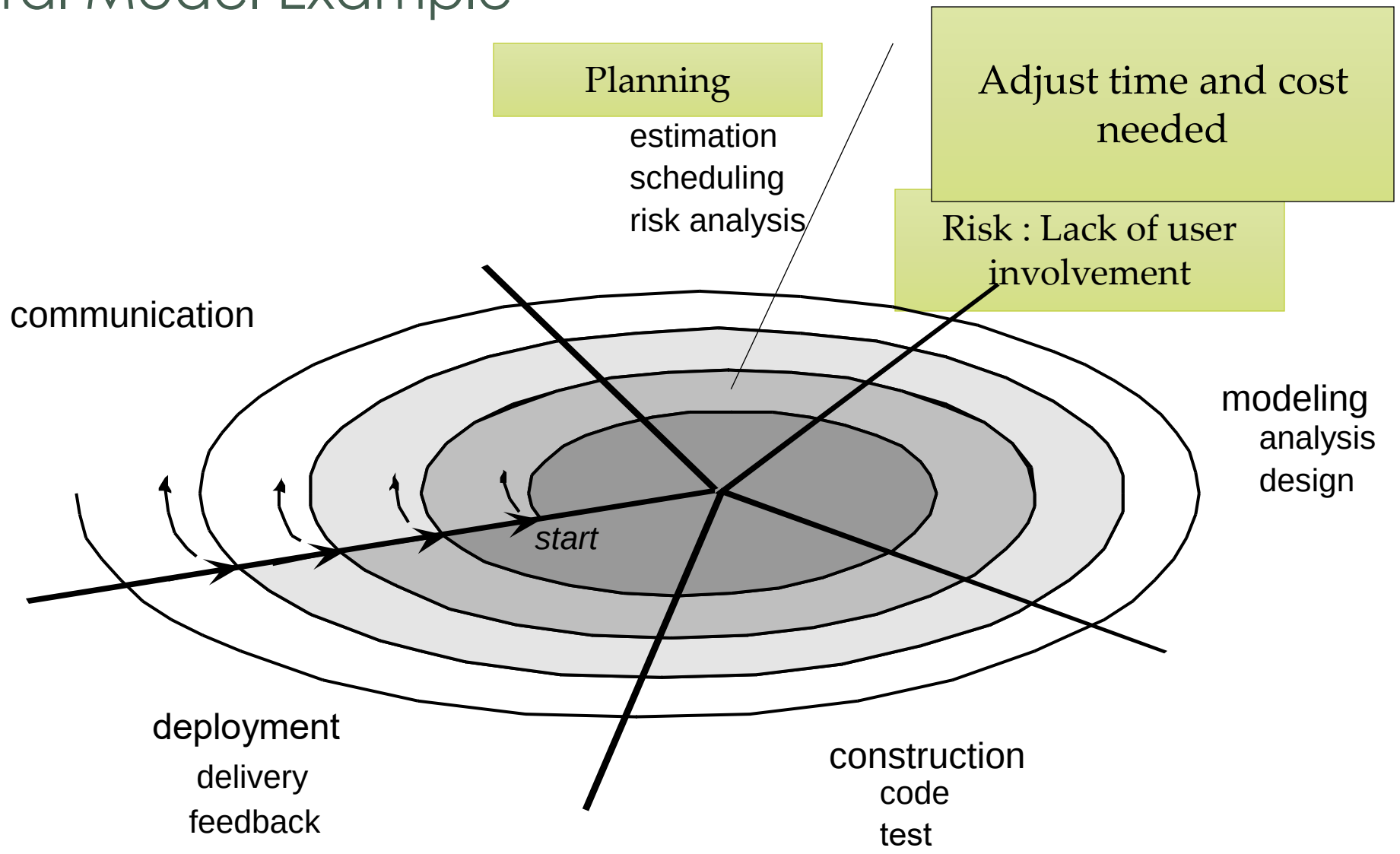
5. Evolutionary Software Process Model: Spiral Model Example



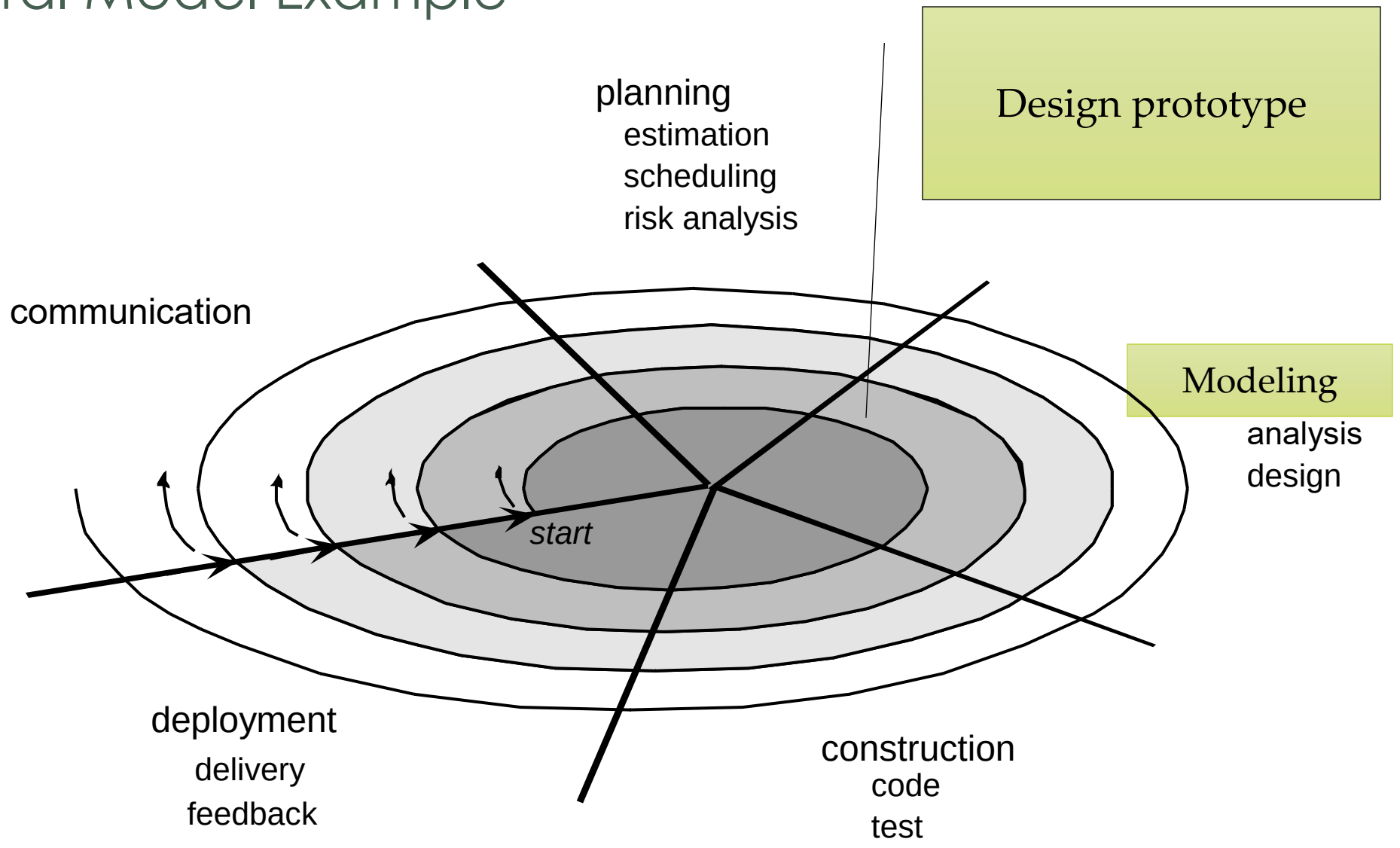
5. Evolutionary Software Process Model: Spiral Model Example



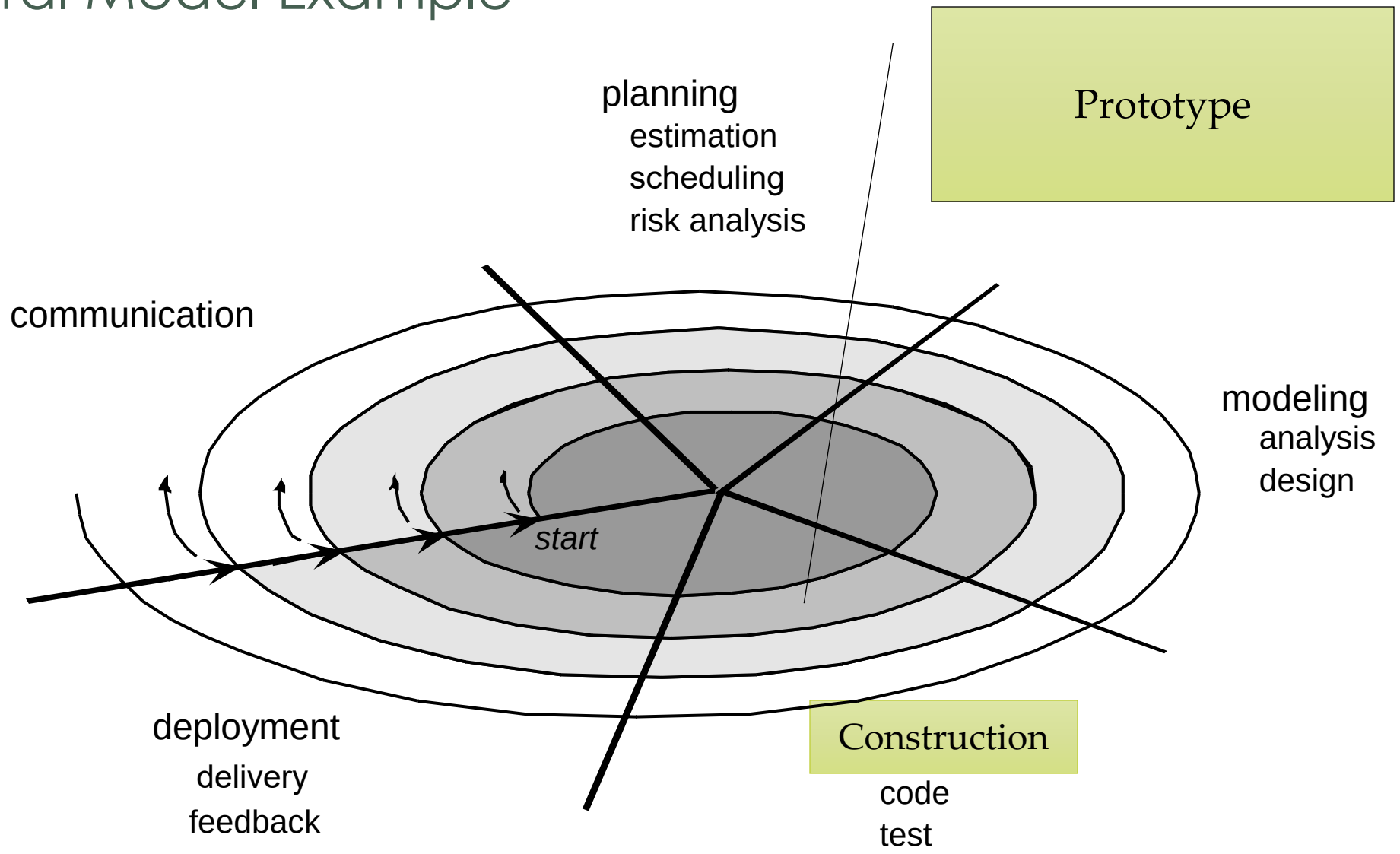
5. Evolutionary Software Process Model: Spiral Model Example



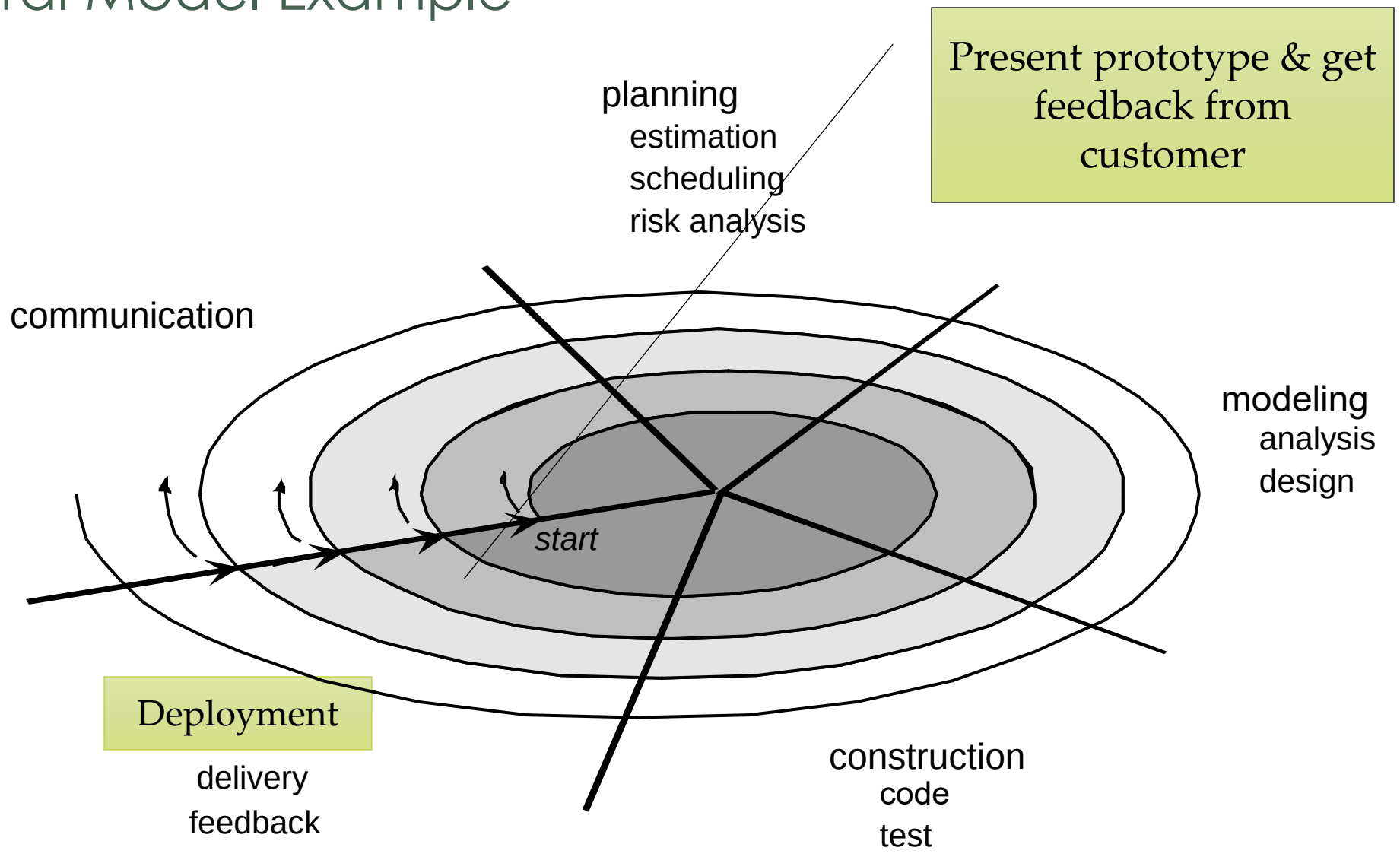
5. Evolutionary Software Process Model: Spiral Model Example



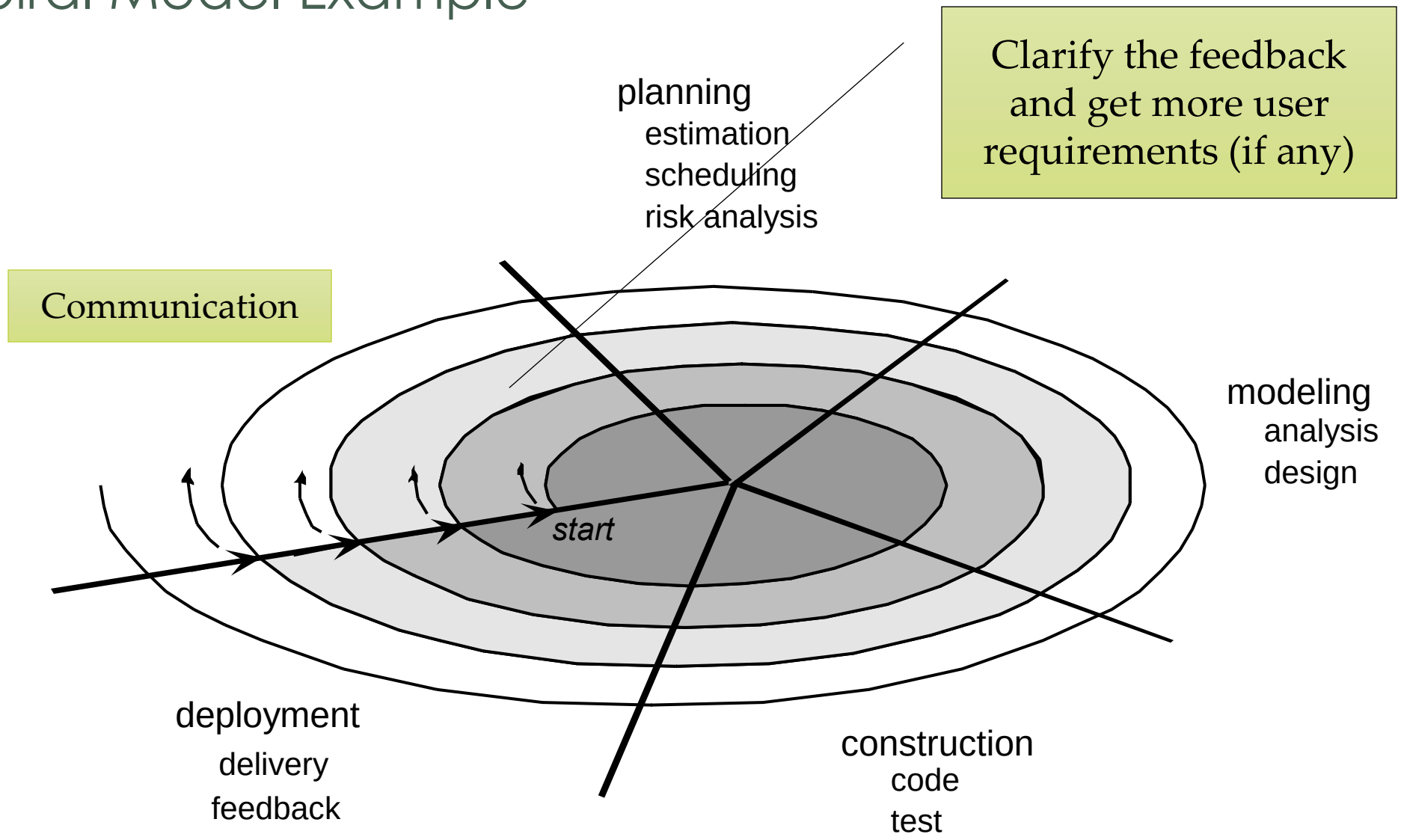
5. Evolutionary Software Process Model: Spiral Model Example



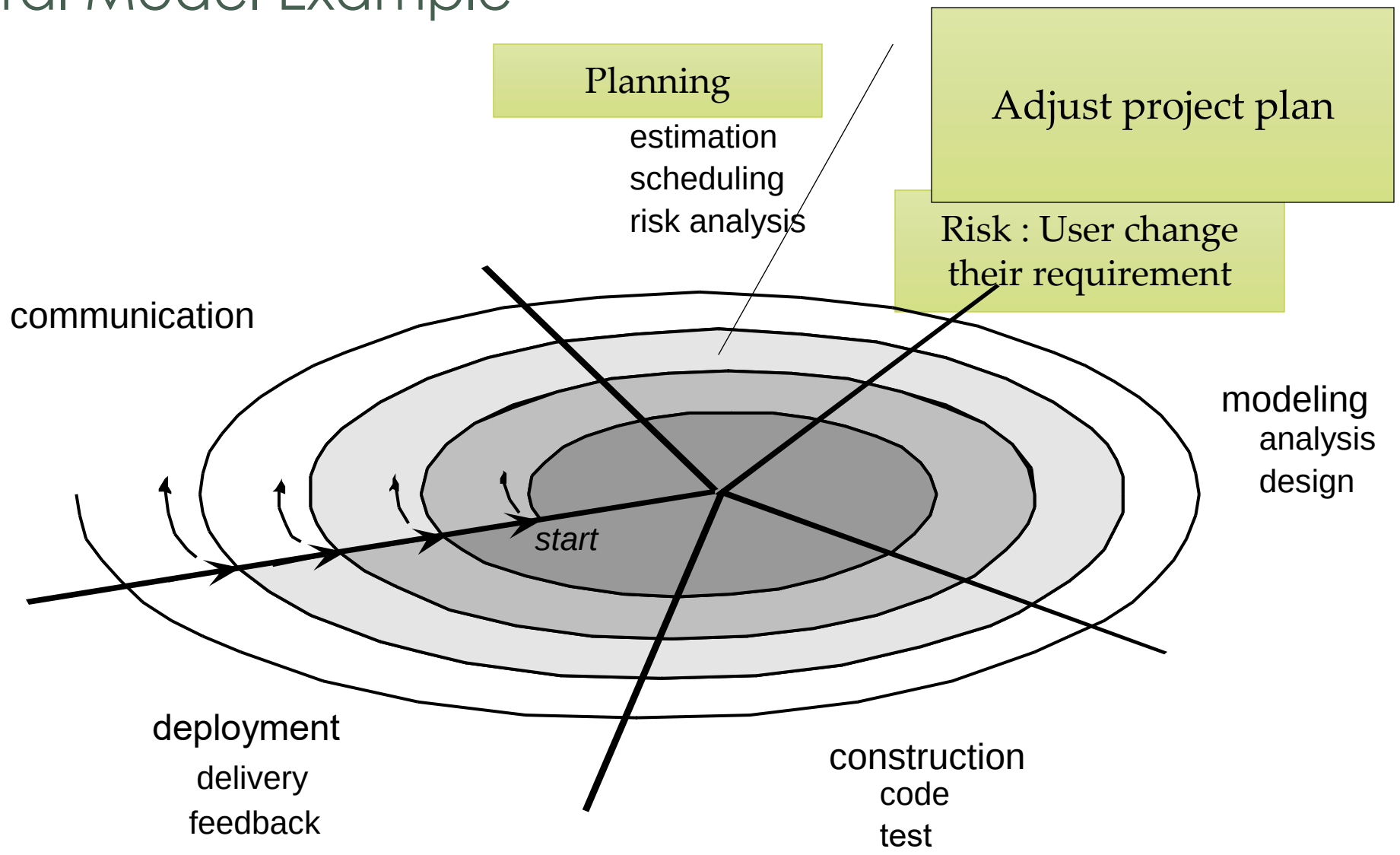
5. Evolutionary Software Process Model: Spiral Model Example



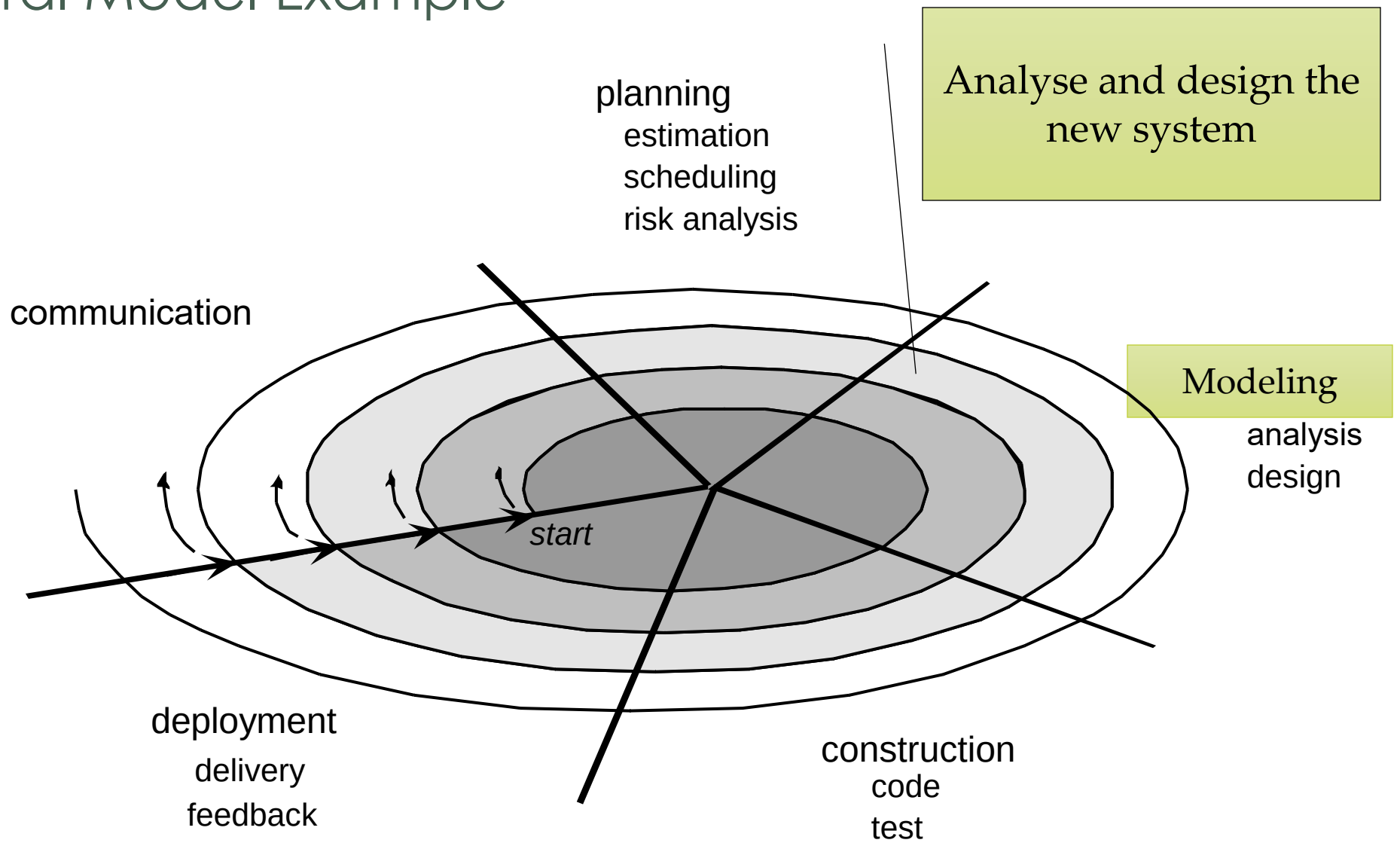
5. Evolutionary Software Process Model: Spiral Model Example



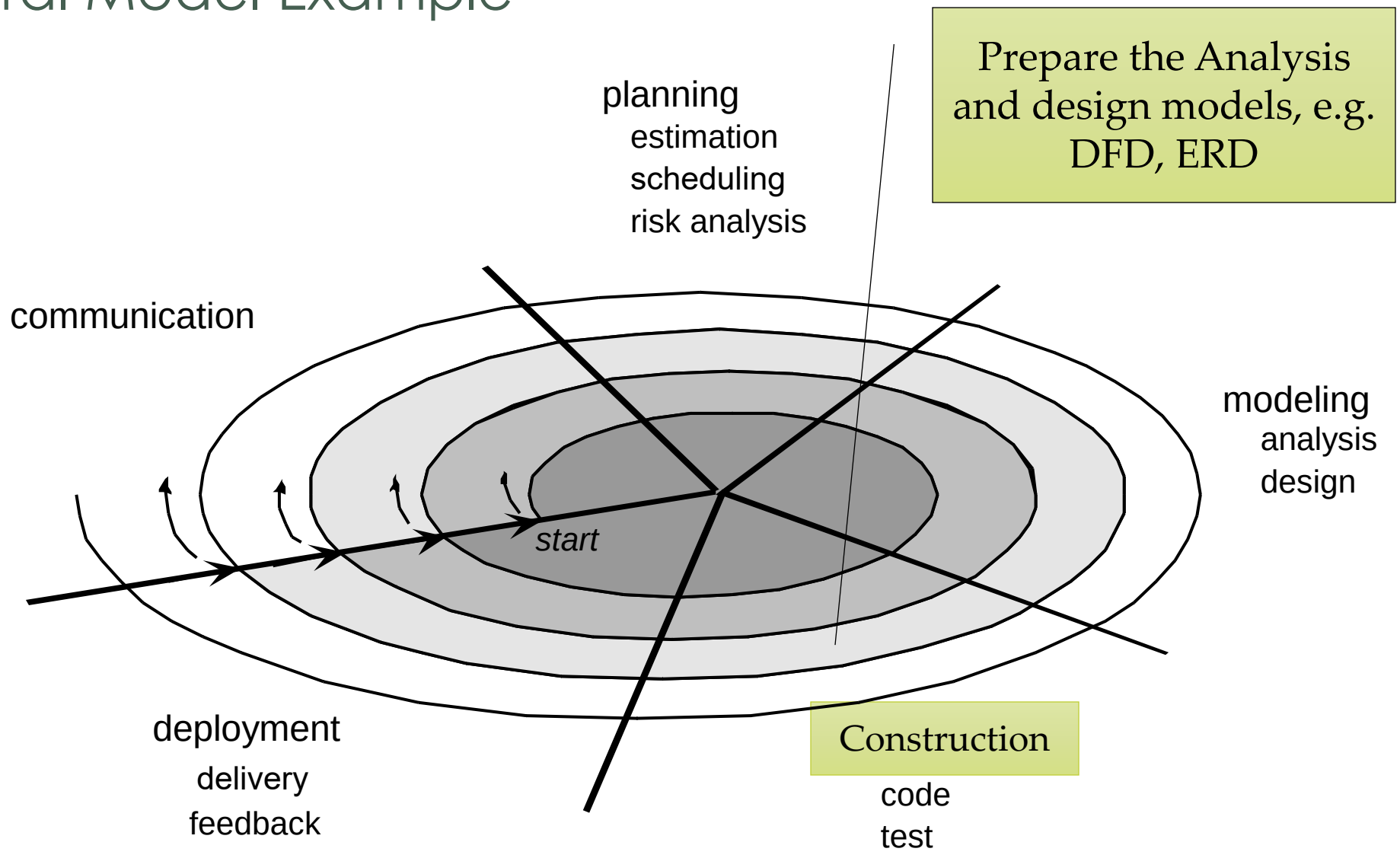
5. Evolutionary Software Process Model: Spiral Model Example



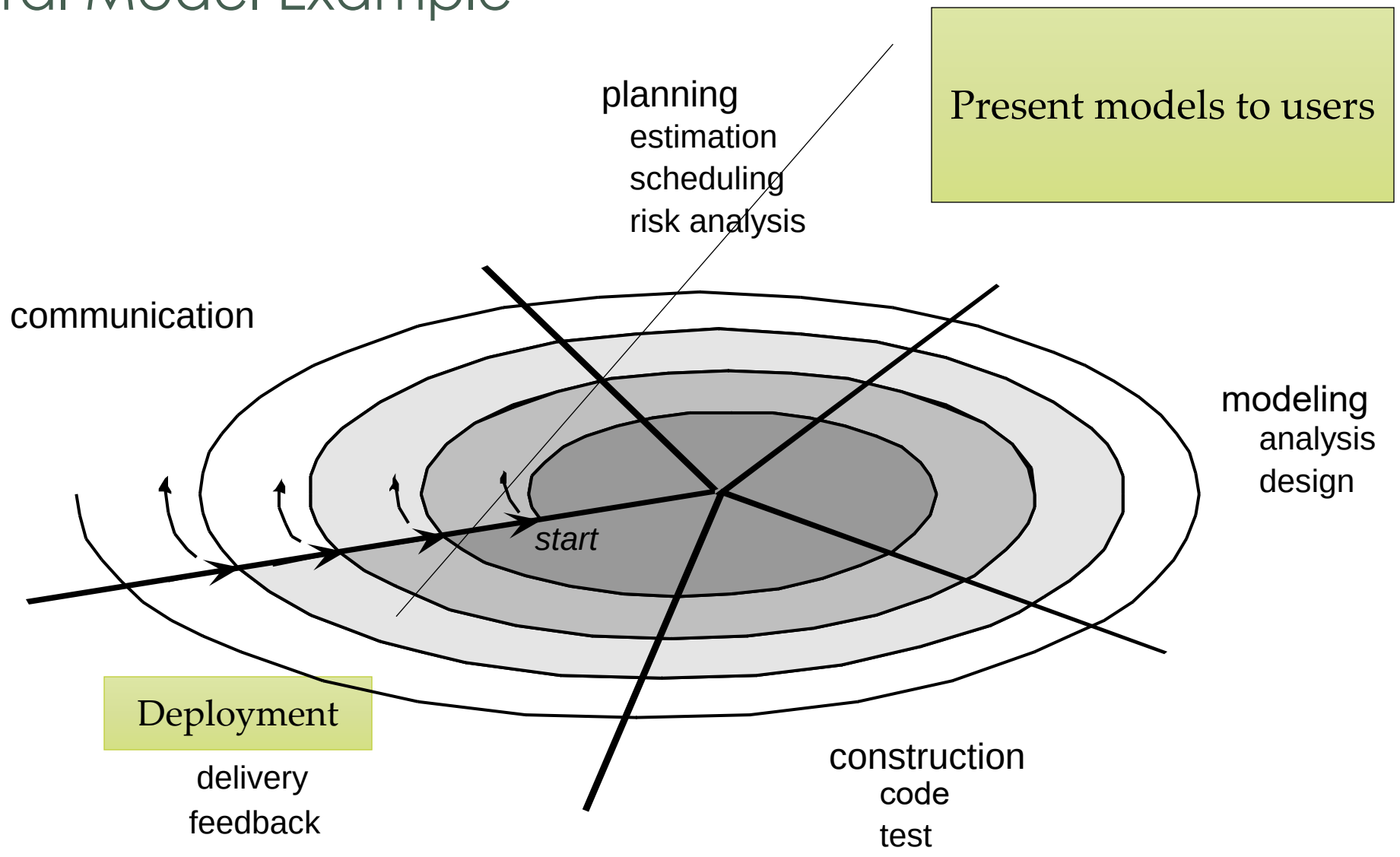
5. Evolutionary Software Process Model: Spiral Model Example



5. Evolutionary Software Process Model: Spiral Model Example



5. Evolutionary Software Process Model: Spiral Model Example



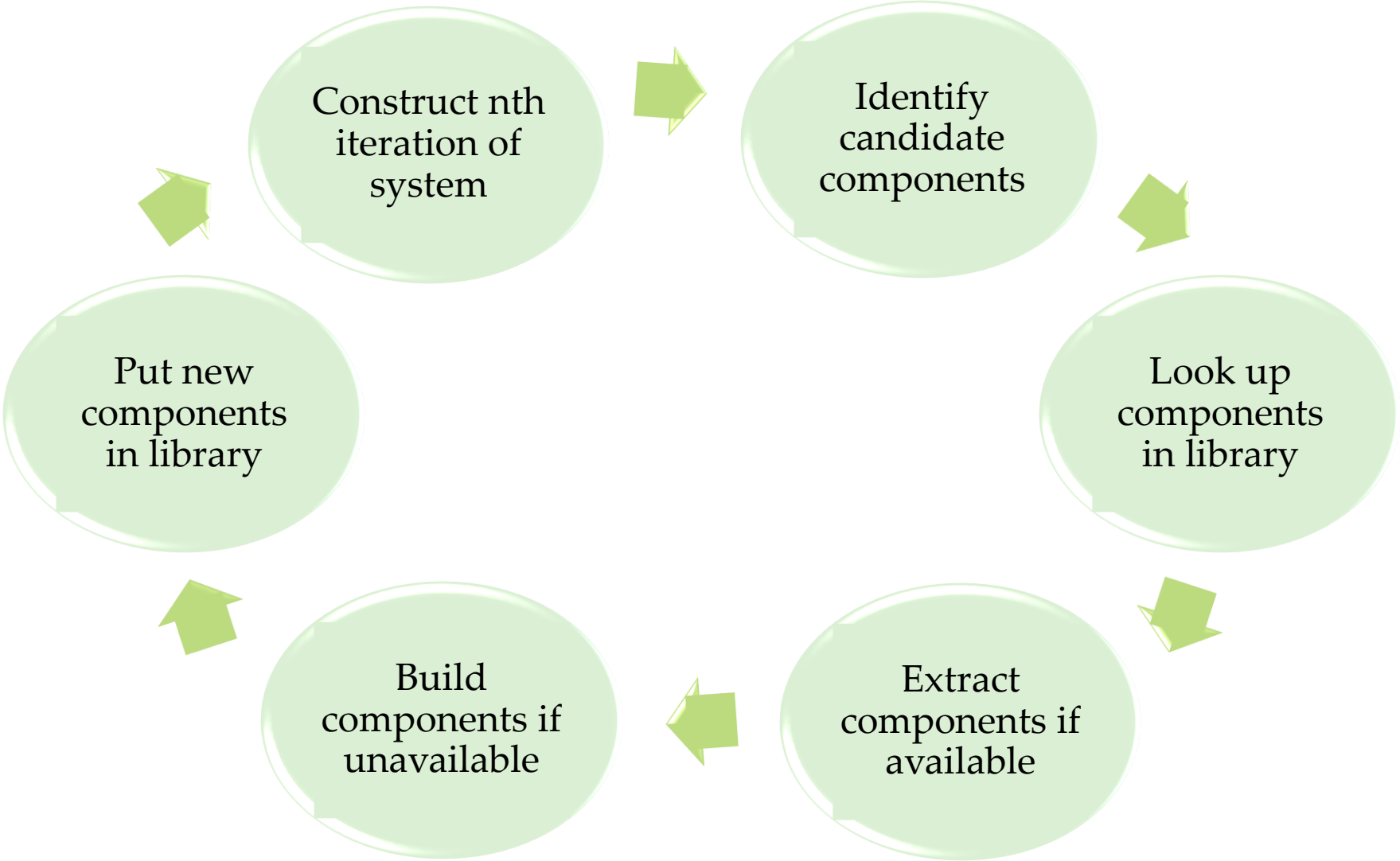
6. Component-based Development

6. Component-based Development

- OO Technologies provide the technical framework for a component-based process model for software engineering.

- This model assumes that parts of the system already exist.

6. Component-based Development



6. Component-based Development

1. It focuses on integrating these parts rather than developing them from scratch

2. Hence, this model leads to software reuse

Advantages

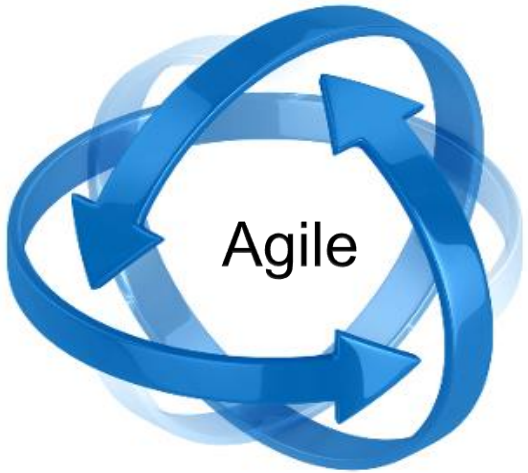
Note : Discuss more detail in later chapter - CBSE.

7. Agile Software Development

7. Agile Software Development

Increment available in 2-3 weeks

Incremental & Iterative



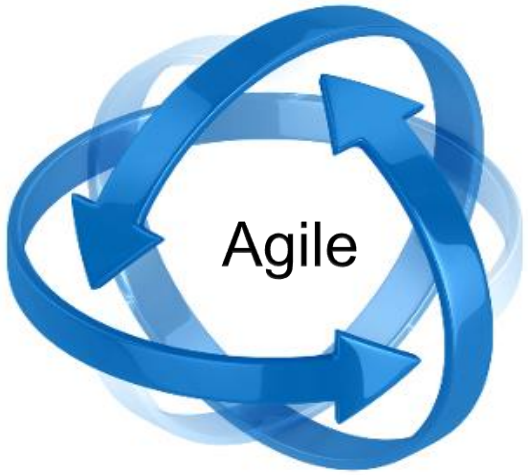
Requirement changing

Minimal documentation

Cross functional team

Iterations with short timeframe

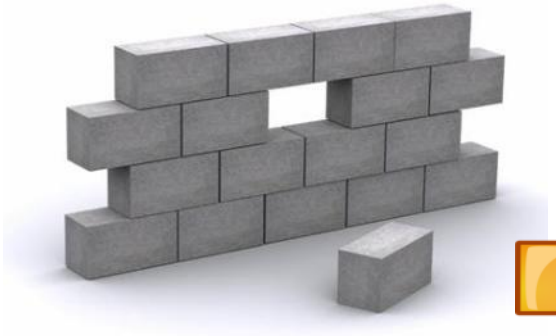
7. Agile Software Development



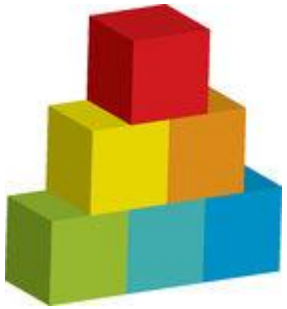
Each iteration:

- Working product
- Review

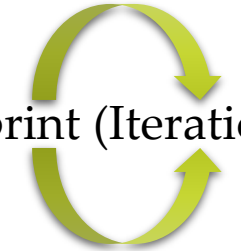
7. Agile Software Development



"To-do" list or
Product Backlog



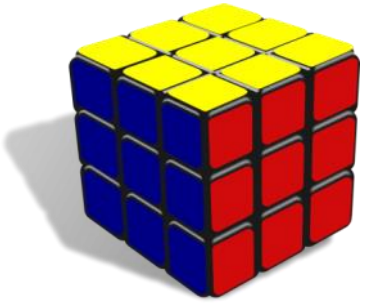
Sprint Backlog



Sprint (Iteration)

- Analysis
- Develop
- Test
- Iteration

Client and stakeholders:
✓Review increment
✓Plan for next iteration

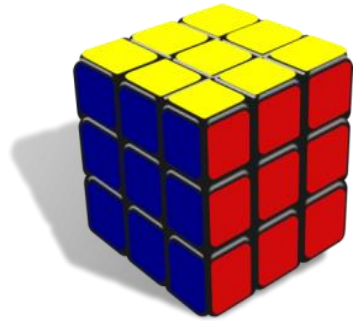


Working software

7. Agile Software Development

Twelve Agility Principles

1. Customer satisfaction
2. Welcome changing requirements
3. Working software is delivered frequently
4. Working software is the principal measure of progress
5. Sustainable development
6. Close, daily co-operation between business people and developers



Working software

7. Agile Software Development

Twelve Agility Principles

7. Face-to-face conversation
8. Motivated individuals
9. Attention to technical excellence and good design
10. Simplicity
11. Self-organizing teams
12. Regular adaptation to changing circumstances

7. Agile Software Development

Plan-Driven vs. Agile Development

Agile Approach	Plan-Driven Approach
Consider design and implementation to be the central activities and thus incorporate requirements elicitation and testing into design and implementation	Identifies separate stages in the software process with outputs associated with each stage.

7. Agile Software Development

Plan-Driven vs. Agile Development

Agile Approach	Plan-Driven Approach
Iteration occurs across activities. Therefore, the requirements and the design are developed together, rather than separately	Iteration occurs within activities with formal documents used to communicate between stages of the process. E.g. requirements will evolve then a requirements specification will be produced as input to the design and implementation process.

7. Agile Software Development

Plan-Driven vs. Agile Development

Agile Approach	**Plan-Driven Approach**
Might produce some design documentation.	Support incremental development and delivery. It is feasible to allocate requirements and plan the design and development phase as a series of increments. E.g. incremental model, spiral model

7. Agile Software Development

Plan-Driven vs. Agile Development

Agile Development	**Plan-driven Development**
Low criticality	High criticality
Senior developers	Junior developers
Requirements change often	Requirements do not change often
Small number of developers	Large number of developers
Culture that responds to change	Culture that demands order

7. Agile Software Development

- Most software projects include practices from both plan-driven and agile approaches.

Exercise

In each of the following consideration, decide which approach (Agile Software Development or Plan-driven Development) is most appropriate.

1. Is it important to have a very detailed specification and design before moving to implementation?

2. Is an incremental delivery strategy, where you deliver the software to customers and get rapid feedback from them realistic?

Exercise

In each of the following consideration, decide which approach (Agile Software Development or Plan-driven Development) is most appropriate.

3. How large is the system?

4. What type of system?

5. What is the expected system lifetime?

6. What technologies are available?

Exercise

In each of the following consideration, decide which approach (Agile Software Development or Plan-driven Development) is most appropriate.

7. How is the development team organized?

8. Are there cultural issues?
9. How good are the designers/programmers?
10. Is the system subject to external regulation?

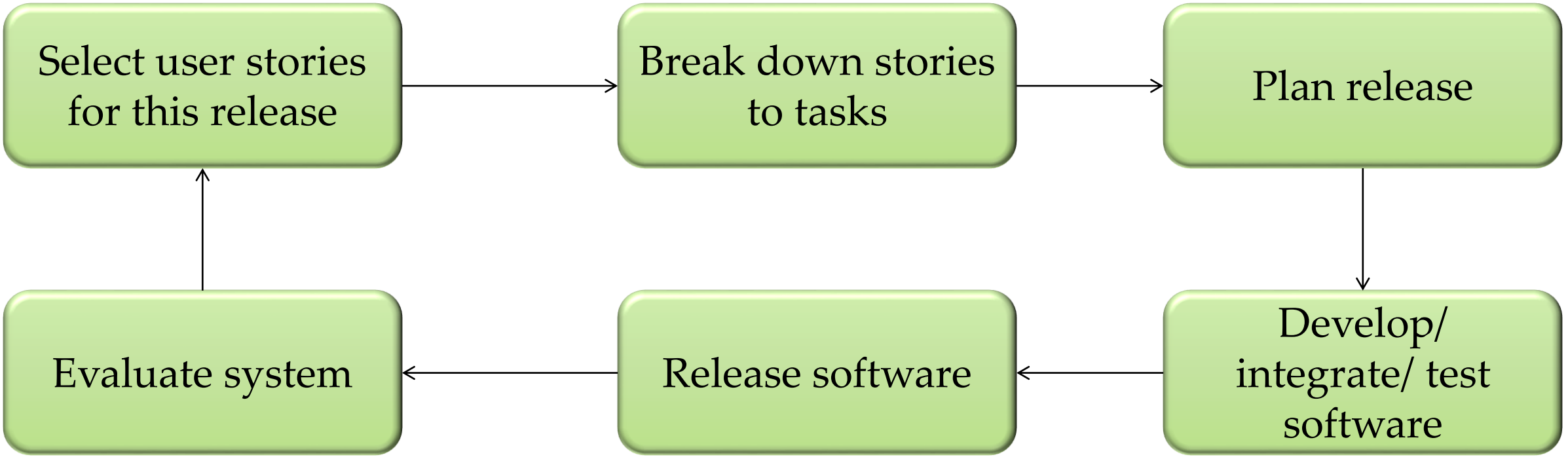
8. Extreme Programming

8. Extreme Programming

- Extreme Programming (XP) is one of the well-known agile software development methods.

- The name was coined by Beck (2000)

8. Extreme Programming



The XP release cycle

8. Extreme Programming

- Incremental planning
- Small releases
- Simple design
- Test-first development
- Refactoring

Refer Sommerville, figure 3.4, pg 66 for explanation

8. Extreme Programming

- Pair programming
- Collective ownership
- Continuous integration
- Sustainable pace
- On-site customer

Refer Sommerville, figure 3.4, pg 66 for explanation

Software Process Model

- Linear Sequential Model/Waterfall Model
- Prototyping Model
- Rapid Application Model
- Evolutionary Model
  - Incremental Model
  - Spiral Model
- Component-based Development Model
- Agile Software Development
  - Extreme Programming

Exercise

	Waterfall	Prototyping	RAD	Incremental Model	Spiral Model	Component-based	Agile	Extreme Programming
User Requirements								
User Involvement								
Time								
Cost								
HR								
HR Expertise								
Changes allowed?								
Iterative?								

Exercise

The software organization you are working with has recently successfully bid for a project to develop an e-learning system for a local established college. You are being assigned as a project manager to lead a project team of 12 members to handle this project.

During the requirement analysis stage, you found out that some of the requirements have not confirmed and finalized by your client. Furthermore, some of the hardware needed to develop certain parts of the system have to order from oversea vendors and they cannot promise the availability of these hardware in the early stage of development. However, your client wants the confirmed areas of the system to be delivered within two months time so that they can use this new system as part of the marketing strategies for the coming new academic intake.

Recommend an appropriate process model you would use to manage the above project. Justify your selection. You may state any assumptions to support your answer.