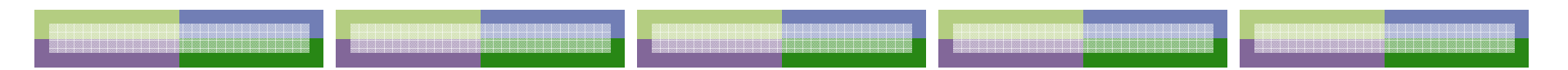


Chapter 3

Requirements and the Software Lifecycle

- The traditional software process models
 - Waterfall model
 - Spiral model
- The iterative approach



Any Good Software Development Process should have

- **Collect + Analysis:**

 - Identify what the system should do

- **Design:**

 - Determine what is the best way to do it

- **Implementation + Test:**

 - Build the system correctly



Traditional Software Process Models

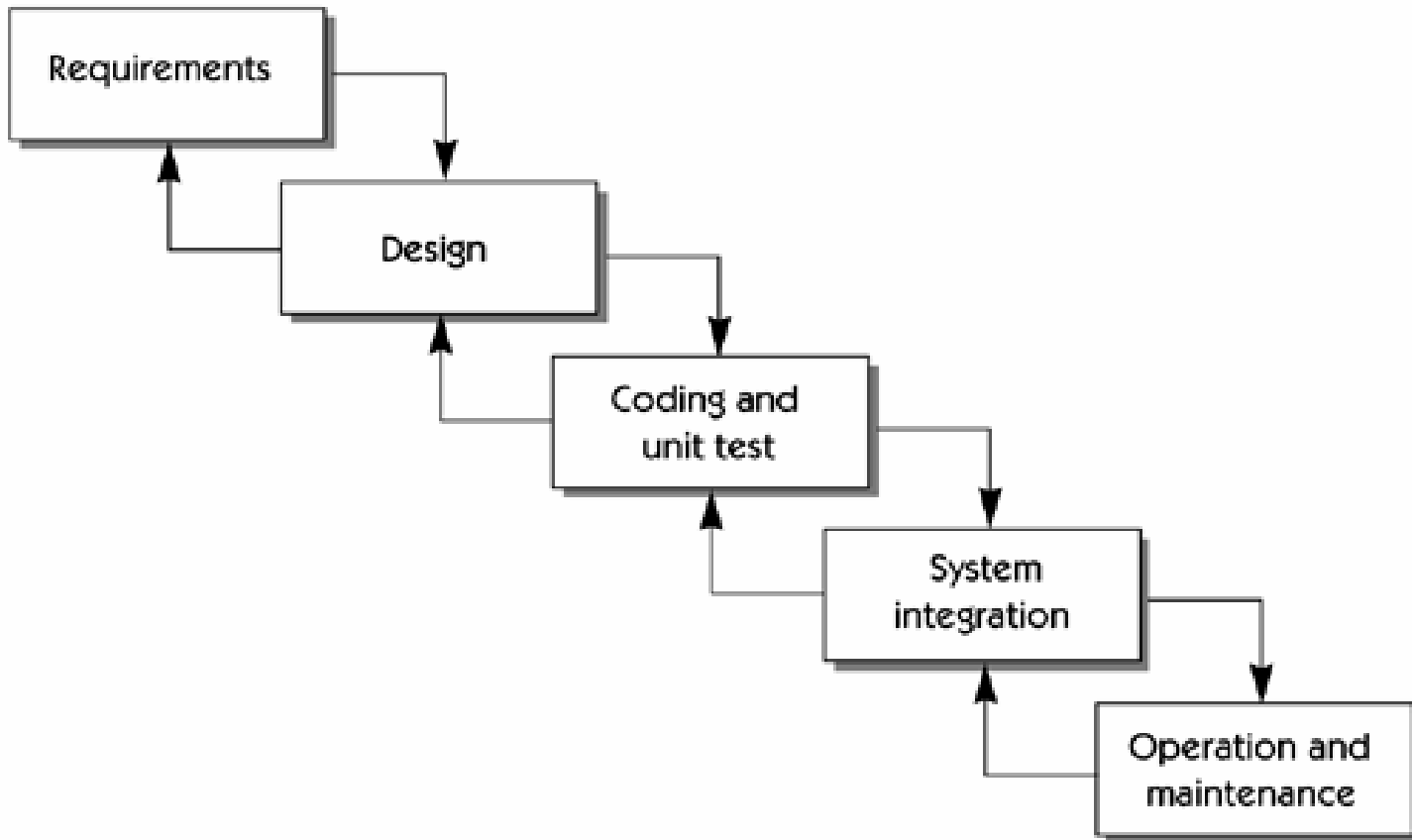
- Effective requirements management can occur only within the context of a reasonably well-defined software process that defines the full set of activities your team must execute to deliver the final software product.
- In order for your team to reach its goal, your team's software development process should define who is doing what, when and how.



The Waterfall Model

- The waterfall model is sequential:
 - Software activities proceed logically through a sequence of steps.
 - Each step bases its work on the activities of the previous step.
- The waterfall model was widely followed in the 1970s and '80s and served successfully as a process model for a variety of medium- to large-scale projects.

The Waterfall Model (Royce 1970)





The Waterfall Model

1. Collect major requirements (hardware, software & human parts), define clearly, and document them.
2. Design & plan how to meet the req.
3. Code the system and test units,
4. Link all parts, install and test as a whole
5. Corrective, adaptive & perfective maintenance



Waterfall Model Problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changes in customer requirements.
- Therefore, this model is **only appropriate when**
 - the requirements are well-understood and
 - changes will be fairly limited during the design process.
- **Long process and no intermediate builds**, but
 - Stakeholders need to gain confidence
 - Designers and developers need confirmation that they are building what is needed and wanted.
- **Some team members are idle** (because they are not involved in the current development phase)
 - We need to put people to work on several phases at once



Incremental Development

● Notice that

Successful large system is originally successful small project that grows incrementally.

● Two incremental models:

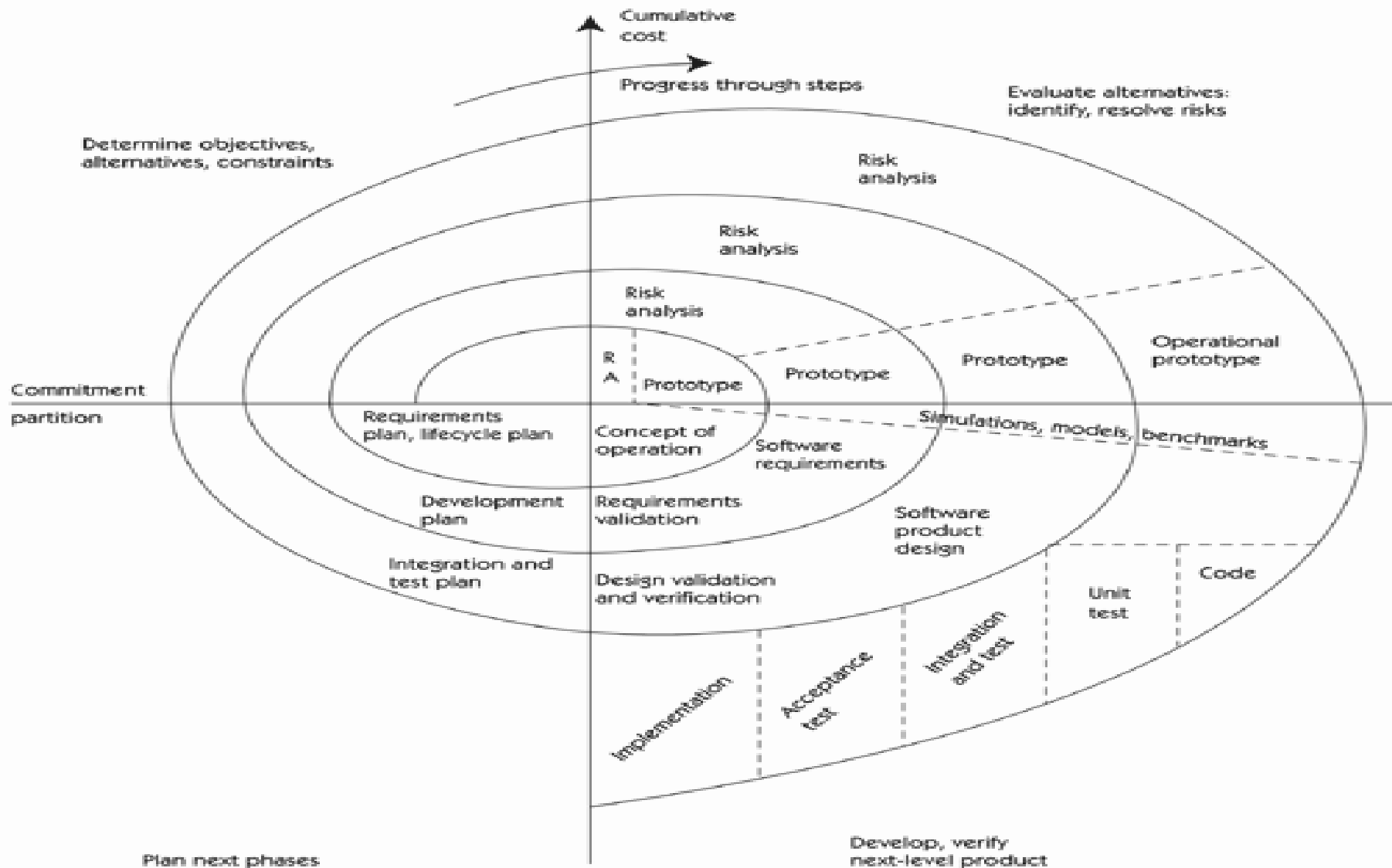
- Spiral Model
- Iterative approach



Prototyping

- **Prototype** is a system or partially complete system that is built quickly to operate some aspects of the requirements.
- Thus, it may have poor performance, less functionality, limited data processing capacity, and low quality.

The Spiral Model (Boehm, 88)





Spiral Model's 4 Sectors

1. **Objectives setting:** specific objectives for the phase are identified.
2. **Risk assessment and reduction:** risks are assessed and activities put in place to reduce the key risks.
3. **Development and validation:** a development model for the system is chosen which can be any of the generic models.
4. **Planning:** the project is reviewed and the next phase of the spiral is planned.



The Spiral Model's properties

- Risk-driven and incremental development process
- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design
- loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.
- The main advantage of this process model is the availability of multiple feedback opportunities with the users and customers.

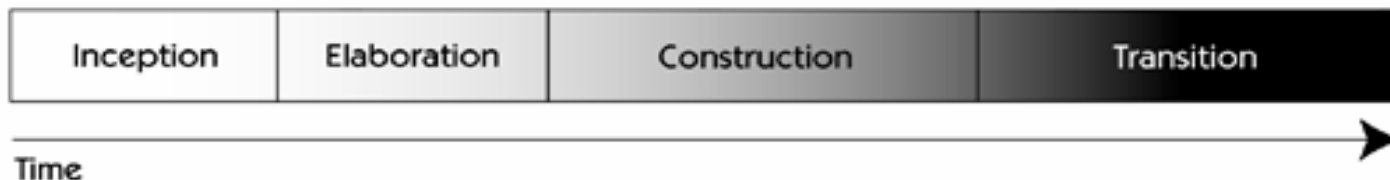


The Iterative Approach (Kruchten, 95)

- In the iterative approach, the lifecycle phases are decoupled from the logical software activities that occur in each phase, allowing us to revisit various activities, such as requirements, design, and implementation, during various iterations of the project.

The Iterative Approach: Lifecycle Phases

- It has 4 phases
 1. **Inception:** scope & purpose of project
 2. **Elaboration:** analysis of the req. and the structure of the system
 3. **Construction:** writing the code
 4. **Transition:** installing the system





The Iterative Approach: Lifecycle Phases

1. Inception phase

- The team is focused on understanding the business case for the project, the scope of the project, and the feasibility of an implementation.
- Problem analysis is performed, the vision for the solution is created, and preliminary estimates of schedule and budget, as well as project risk factors, are defined.



The Iterative Approach: Lifecycle Phases

2. Elaboration phase

- The requirements for the system are refined, an initial, perhaps even executable, architecture is established, and an early feasibility prototype is typically developed and demonstrated.



The Iterative Approach: Lifecycle Phases

3. Construction phase

- The focus is on implementation.
- Most of the coding is done in this phase, and the architecture and design are fully developed.

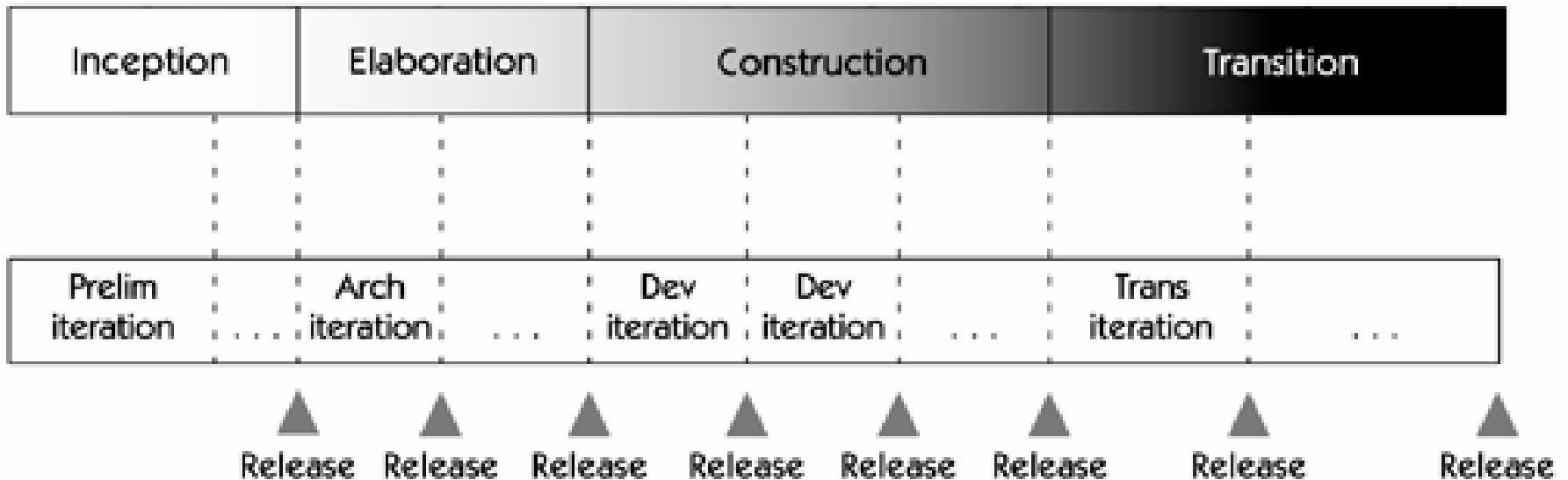


The Iterative Approach: Lifecycle Phases

4. Transition phase

- Beta testing
- The users and maintainers of the system are trained on the application.
- The tested baseline of the application is transitioned to the user community and deployed for use.

The Iterative Approach: Iterations

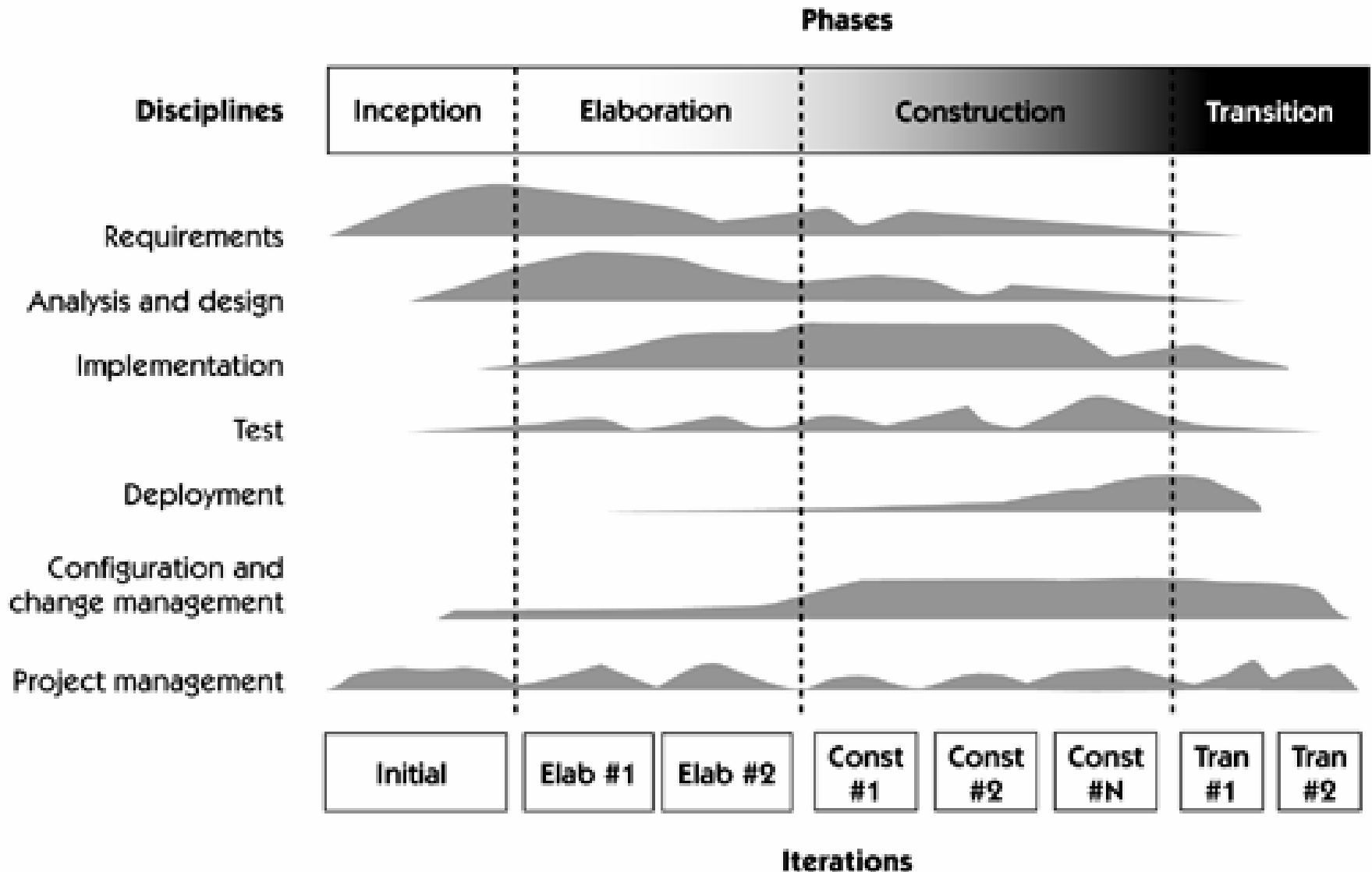




The Iterative Approach: Iterations

- Within each phase, the project typically undergoes multiple iterations.
- An iteration is a sequence of activities with an established plan and evaluation criteria, resulting in an executable of some type.
- Each iteration builds on the functionality of the prior iteration; thus, the project is developed in an "iterative and incremental" fashion.

The Iterative Approach: Disciplines





The Iterative Approach: Disciplines

- In the iterative approach, the activities associated with the development of the software are organized into a set of disciplines.
- Each discipline consists of a logically related set of activities, and each defines how the activities must be sequenced to produce a viable work product (or artifact).
- Although the number and kind of disciplines can vary, based on the company or project circumstances, there are typically at least six disciplines.



The Iterative Approach: Disciplines

- During each iteration, the team spends as much time as appropriate in each discipline.
 - Thus, an iteration can be regarded as a mini-waterfall through the activities of requirements, analysis and design, and so on, but each mini-waterfall is "tuned" to the specific needs of that iteration.
- The size of the "hump" indicates the relative amount of effort invested in a discipline.
 - For example, in the elaboration phase, significant time is spent on "refining" the requirements and in defining the architecture that will support the functionality.
- The activities can be sequential (a true mini-waterfall) or may execute concurrently, as is appropriate to the project.



Requirements in the Iterative Model

- From the requirements management perspective, the iterative approach provides two major advantages:
 1. Better adaptability to requirements change.
 2. Better scope management.



Key Points

- The **team's development process** defines who is doing what, when, and how.
- In **the waterfall model**, software activities proceeded through a sequence of steps, and requirements were "fixed" early.
- In **the spiral model**, the first steps were taken to a more risk-driven and incremental approach to development.
- The **iterative approach**, a hybrid of the waterfall and spiral models, decouples the lifecycle phases from the software activities that take place in each phase.
- The **iterative model** is a more robust model and provides for successive refinement of the requirements over time.