Software Development Process

- A structure imposed on the development of a software product.
- Defines **Who, What, When and How** of Developing Software.
- There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process.

http://en.wikipedia.org/wiki/IBM_Rational_Unified _Process

UCSC

Software Development Process

- Domain Analysis
  - The first step in attempting to design a new piece of software, whether it be an addition to an existing software, a new application, a new subsystem or a whole new system, is, what is generally referred to as "Domain Analysis".
  - Assuming that the developers (including the analysts) are not sufficiently knowledgeable in the subject area of the new software, the first task is to investigate the so-called "domain" of the software.

UCSC

Software Development Process

- Domain Analysis
  - make the analysts who will later try to elicit and gather the requirements from the area experts or professionals, speak with them in the domain's own terminology and to better understand what is being said by these people.
  - An important step to extracting and gathering the requirements.

Software Development Process

- Software Elements Analysis
  - ➢ The most important task in creating a software product is extracting the requirements.
  - ➢ Customers typically know what they want, but not what software should do, while incomplete, ambiguous or contradictory requirements are recognized by skilled and experienced software engineers.
  - ➢ Frequently demonstrating live code may help reduce the risk that the requirements are incorrect.

Software Development Process

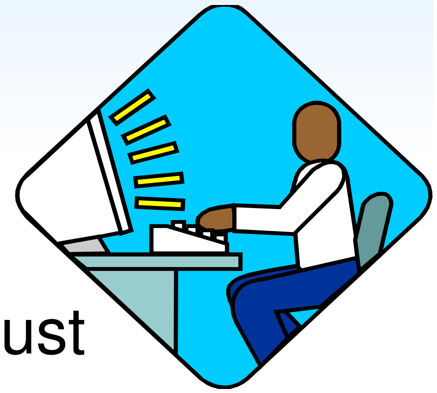
- Specification
  - The task of precisely describing the software to be written, possibly in a rigorous way.

Software Development Process

- Software Architecture
  - An abstract representation of that system
  - Addresses interfaces between the software system and other software products, as well as the underlying hardware or the host operating system

Software Development Process

- Implementation (or coding)
  - A realization of a technical specification or algorithm as a program, software component, or other computer system.
- Testing
  - Testing of parts of software, especially where code by two different engineers must work together, falls to the software engineer.



Software Development Process

- Documentation
  - An important (and often overlooked) task is documenting the internal design of software for the purpose of future maintenance and enhancement.
  - *Documentation is most important for external interfaces.*

UCSC

Software Development Process

- Software Training and Support
  - ➢ People are occasionally resistant to change and avoid venturing into an unfamiliar area so it is very important to have training classes for the most enthusiastic software users (build excitement and confidence)

Software Development Process

- Maintenance
  - Maintaining and enhancing software to cope with newly discovered problems or new requirements can take far more time than the initial development of the software.

The software process

- A structured set of activities required to develop a software system.
- Many different software processes but all involve:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking that it does what the customer wants;
  - Evolution – changing the system in response to changing customer needs.
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

UCSC

Software process descriptions

- When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc. and the ordering of these activities.
- Process descriptions may also include:
  - Products, which are the outcomes of a process activity;
  - Roles, which reflect the responsibilities of the people involved in the process;
  - Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced.

UCSC

Plan-driven and agile processes

- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.
- There are no right or wrong software processes.

Software process models

- The waterfall model
  - Plan-driven model. Separate and distinct phases of specification and development.
- Incremental development
  - Specification, development and validation are interleaved. May be plan-driven or agile.
- Reuse-oriented software engineering
  - The system is assembled from existing components. May be plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of these models.

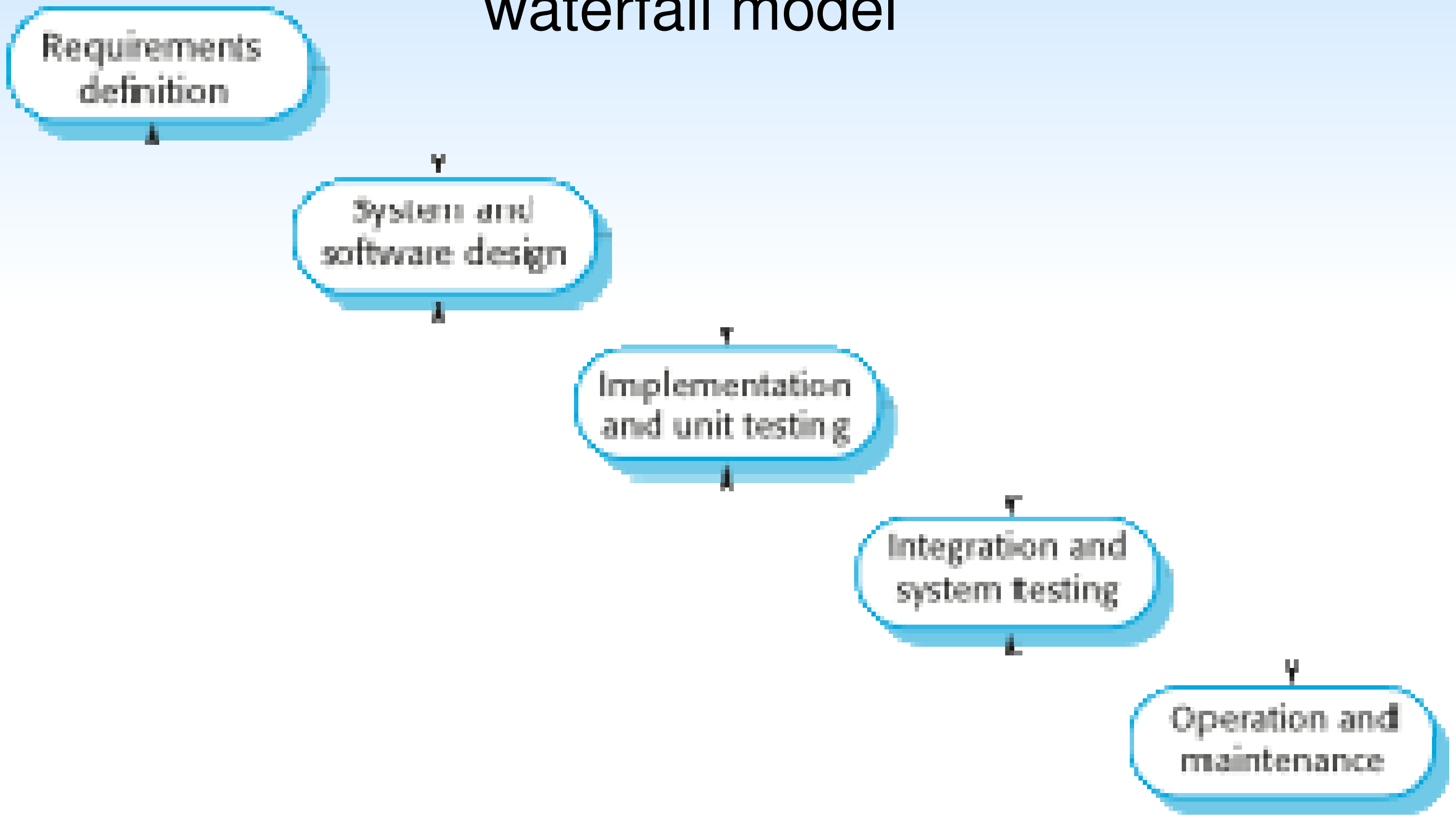
Waterfall Process

- Best known and oldest process
- a sequential software development model in which development is seen as flowing steadily downwards (like a waterfall) through the phases of
  - ***requirements analysis,***
  - ***design,***
  - ***implementation,***
  - ***testing (validation),***
  - ***integration, and***
  - ***maintenance.***

UCSC

The waterfall model

Because of the cascade from one phase to a another, this model is known as the waterfall model


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Waterfall model phases

- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

UCSC

Waterfall model problems

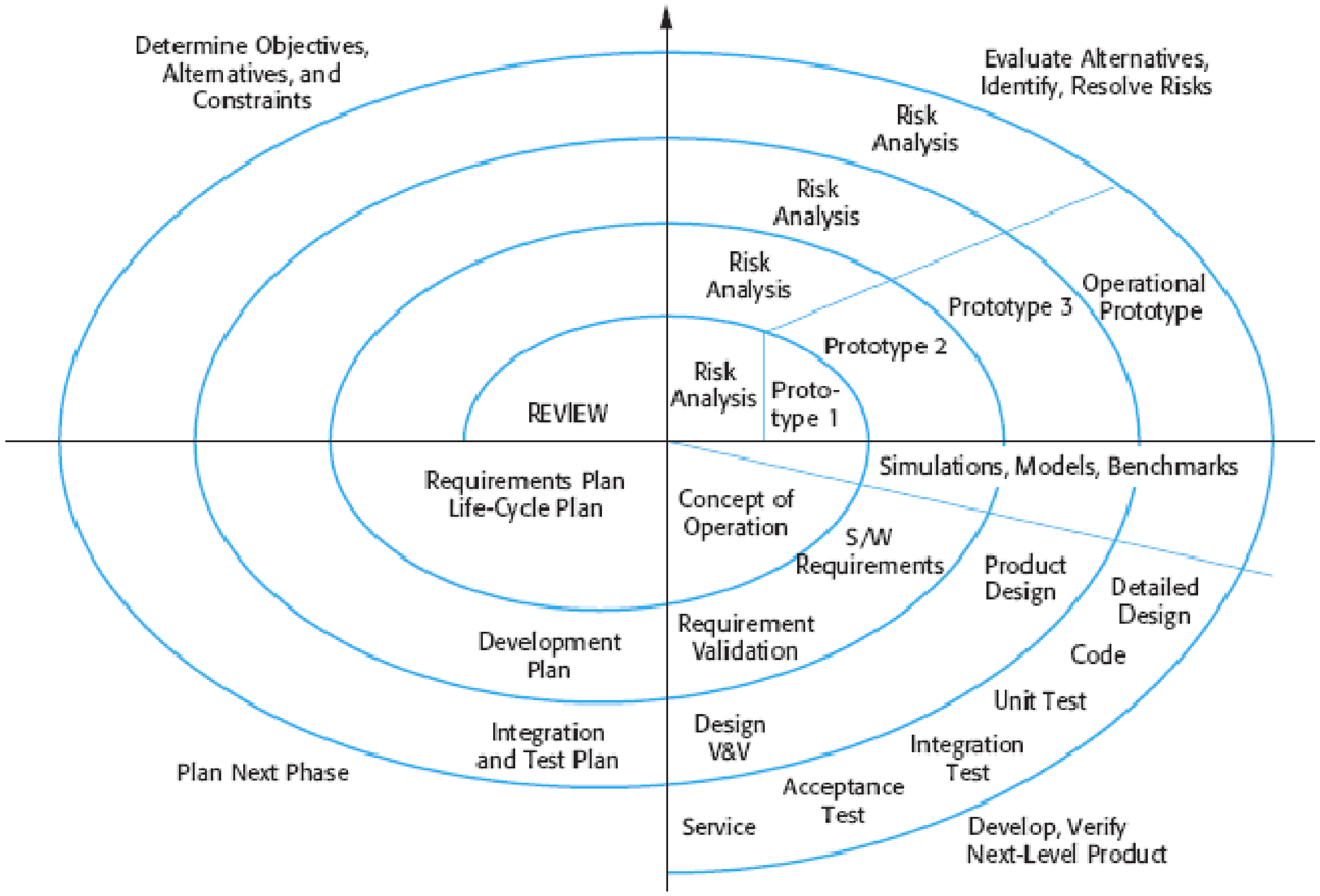
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Spiral development

- 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.

UCSC

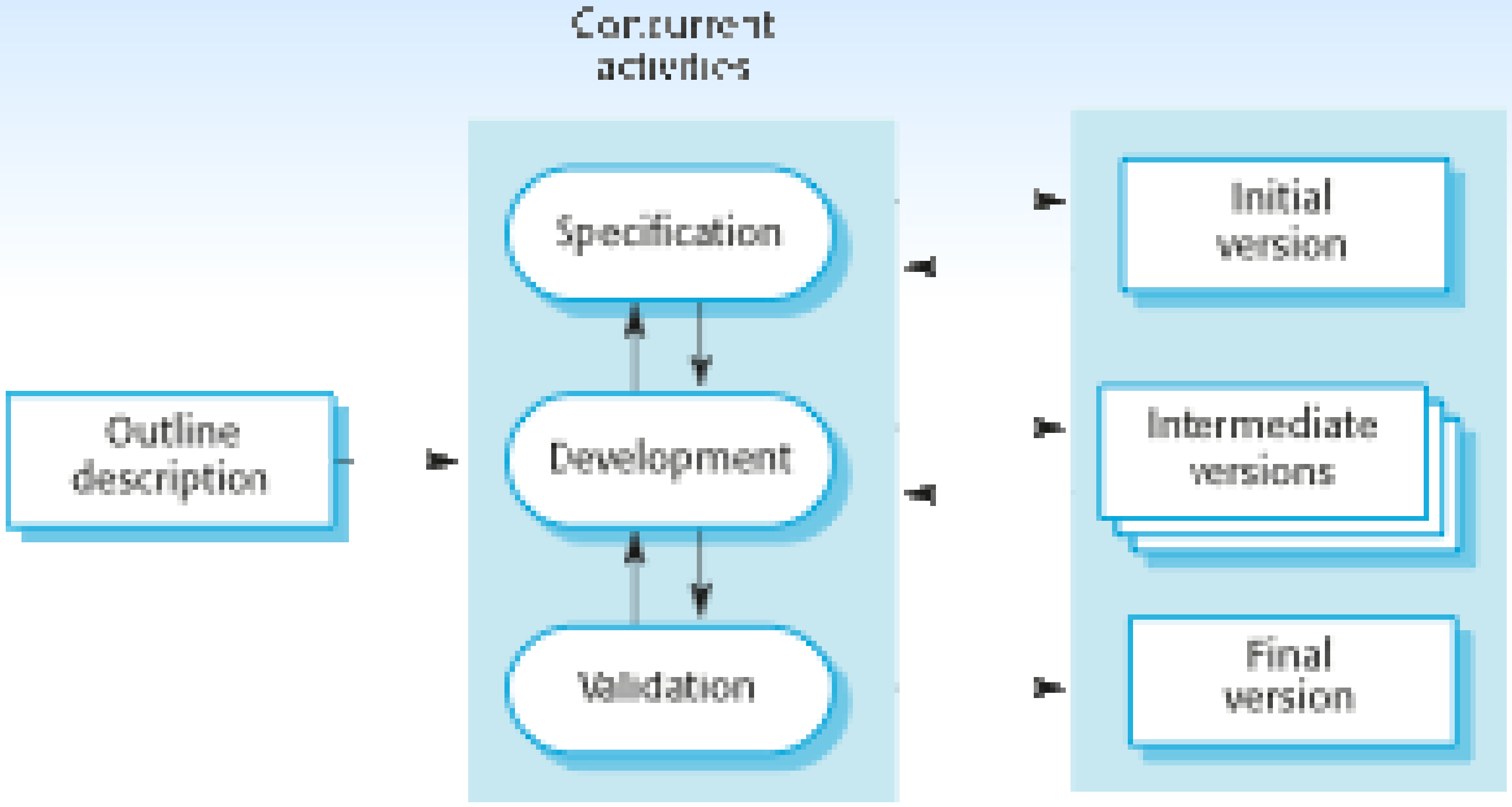
Spiral model of the software process- (a risk driven software process framework)



Spiral model sectors

- Objective setting
  - Specific objectives for the phase are identified.
- Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce the key risks.
- Development and validation
  - A development model for the system is chosen which can be any of the generic models.
- Planning
  - The project is reviewed and the next phase of the spiral is planned.

Incremental development



Incremental development benefits

- The cost of accommodating changing customer requirements is reduced.
  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
  - Customers can comment on demonstrations of the software and see how much has been implemented.
- More rapid delivery and deployment of useful software to the customer is possible.
  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

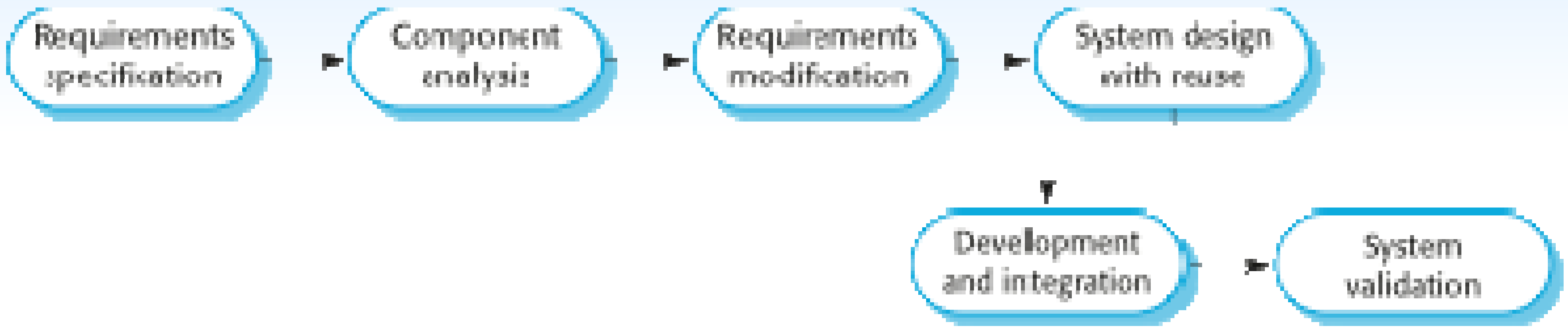
Incremental development problems

- The process is not visible.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

Reuse-oriented software engineering

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Process stages
  - Component analysis;
  - Requirements modification;
  - System design with reuse;
  - Development and integration.
- Reuse is now the standard approach for building many types of business system
  - Reuse covered in more depth in Chapter 16.

Reuse-oriented software engineering


Requirements specification
Component analysis
Requirements modification
System design with reuse
Development and integration
System validation

Rapid software development

- Rapid development and delivery is now often the most important requirement for software systems
  - Businesses operate in a fast –changing requirement and it is practically impossible to produce a set of stable software requirements
  - Software has to evolve quickly to reflect changing business needs.
- Rapid software development
  - Specification, design and implementation are inter-leaved
  - System is developed as a series of versions with stakeholders involved in version evaluation
  - User interfaces are often developed using an IDE and graphical toolset.

Agile methods

- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods. These methods:
  - Focus on the code rather than the design
  - Are based on an iterative approach to software development
  - Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.
- The aim of agile methods is to reduce overheads in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework.

Agile manifesto

- *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
  - *Individuals and interactions over processes and tools*
    Working software over comprehensive documentation
    Customer collaboration over contract negotiation
    Responding to change over following a plan
- *That is, while there is value in the items on the right, we value the items on the left more.*

The principles of agile methods

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

Agile method applicability

- Product development where a software company is developing a small or medium-sized product for sale.
- Custom system development within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are not a lot of external rules and regulations that affect the software.
- Because of their focus on small, tightly-integrated teams, there are problems in scaling agile methods to large systems.

UCSC

Problems with agile methods

- It can be difficult to keep the interest of customers who are involved in the process.
- Team members may be unsuited to the intense involvement that characterises agile methods.
- Prioritising changes can be difficult where there are multiple stakeholders.
- Maintaining simplicity requires extra work.
- Contracts may be a problem as with other approaches to iterative development.

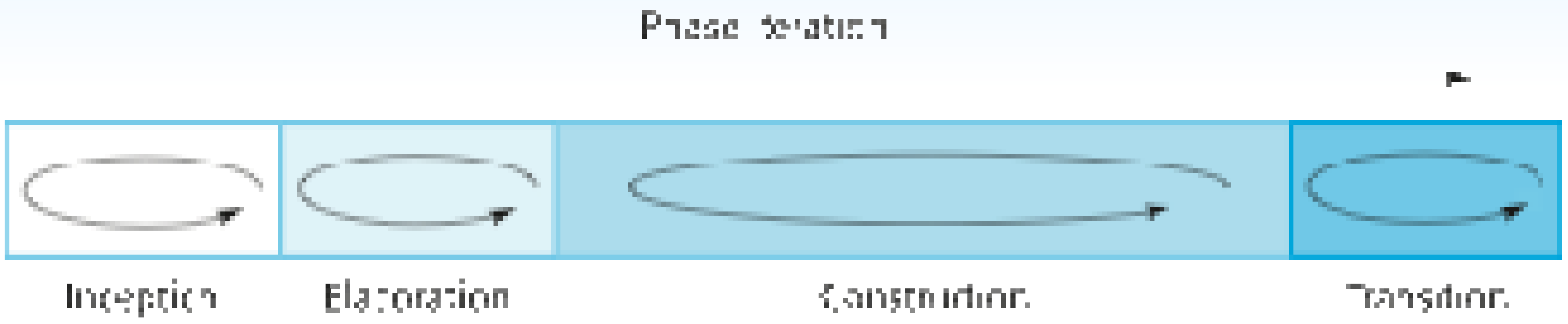
Agile methods and software maintenance

- Most organizations spend more on maintaining existing software than they do on new software development. So, if agile methods are to be successful, they have to support maintenance as well as original development.
- Two key issues:
  - Are systems that are developed using an agile approach maintainable, given the emphasis in the development process of minimizing formal documentation?
  - Can agile methods be used effectively for evolving a system in response to customer change requests?
- Problems may arise if original development team cannot be maintained.

The Rational Unified Process

- A modern generic process derived from the work on the UML and associated process.
- Brings together aspects of the generic process models discussed previously.
- Normally described from 3 perspectives
  - A dynamic perspective that shows phases over time;
  - A static perspective that shows process activities;
  - A practive perspective that suggests good practice.

Phases in the Rational Unified Process



RUP phases

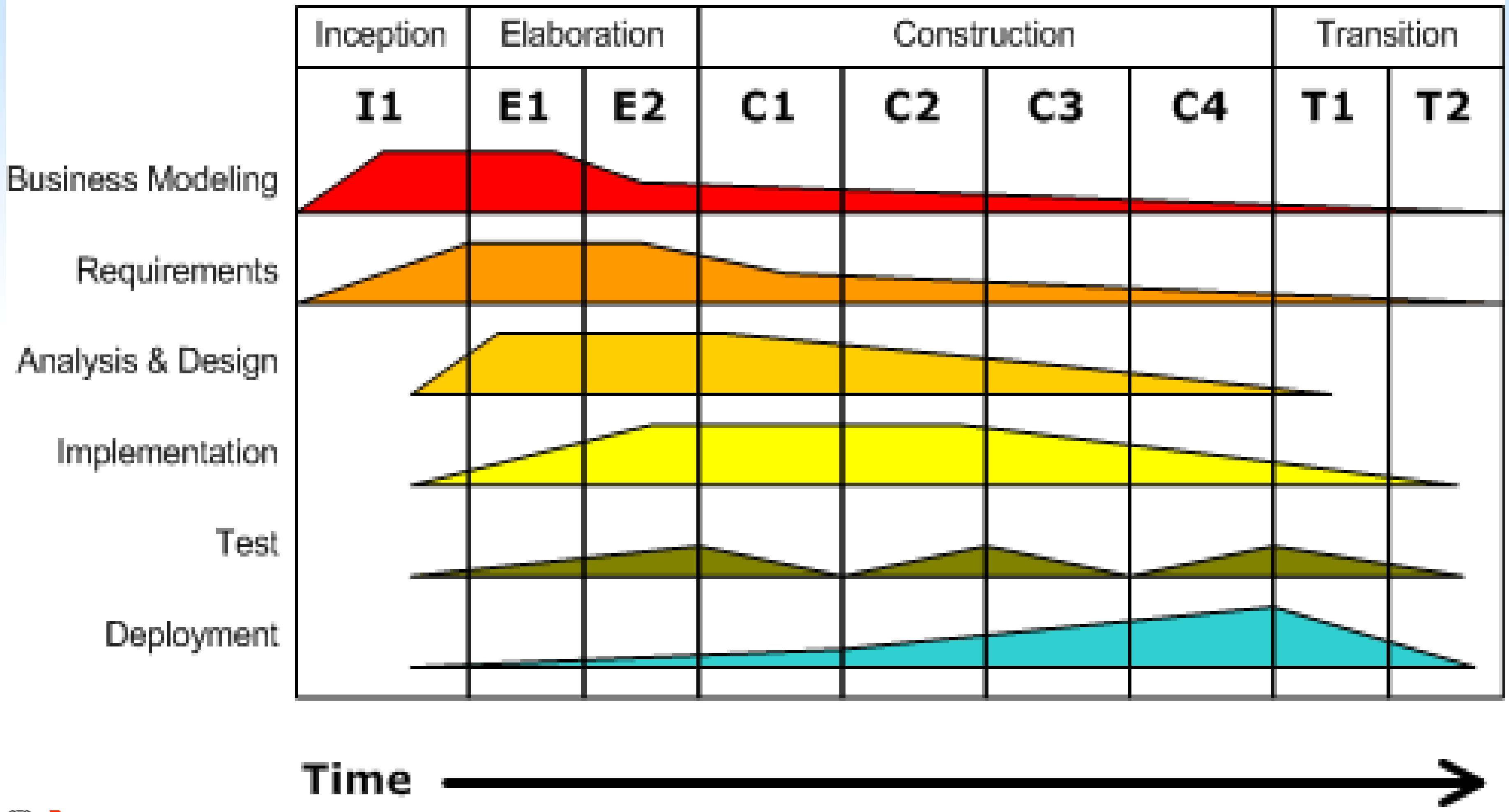
- Inception
  - Establish the business case for the system.
- Elaboration
  - Develop an understanding of the problem domain and the system architecture.
- Construction
  - System design, programming and testing.
- Transition
  - Deploy the system in its operating environment.

RUP iteration

- In-phase iteration
  - Each phase is iterative with results developed incrementally.
- Cross-phase iteration
  - As shown by the loop in the RUP model, the whole set of phases may be enacted incrementally.

Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.



Static workflows in the Rational Unified Process

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.

Static workflows in the Rational Unified Process

Workflow	Description
Testing	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system (see Chapter 25).
Project management	This supporting workflow manages the system development (see Chapters 22 and 23).
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

UCSC

RUP good practice

- Develop software iteratively
  - Plan increments based on customer priorities and deliver highest priority increments first.
- Manage requirements
  - Explicitly document customer requirements and keep track of changes to these requirements.
- Use component-based architectures
  - Organize the system architecture as a set of reusable components.

RUP good practice

- Visually model software
  - Use graphical UML models to present static and dynamic views of the software.
- Verify software quality
  - Ensure that the software meet’s organizational quality standards.
- Control changes to software
  - Manage software changes using a change management system and configuration management tools.

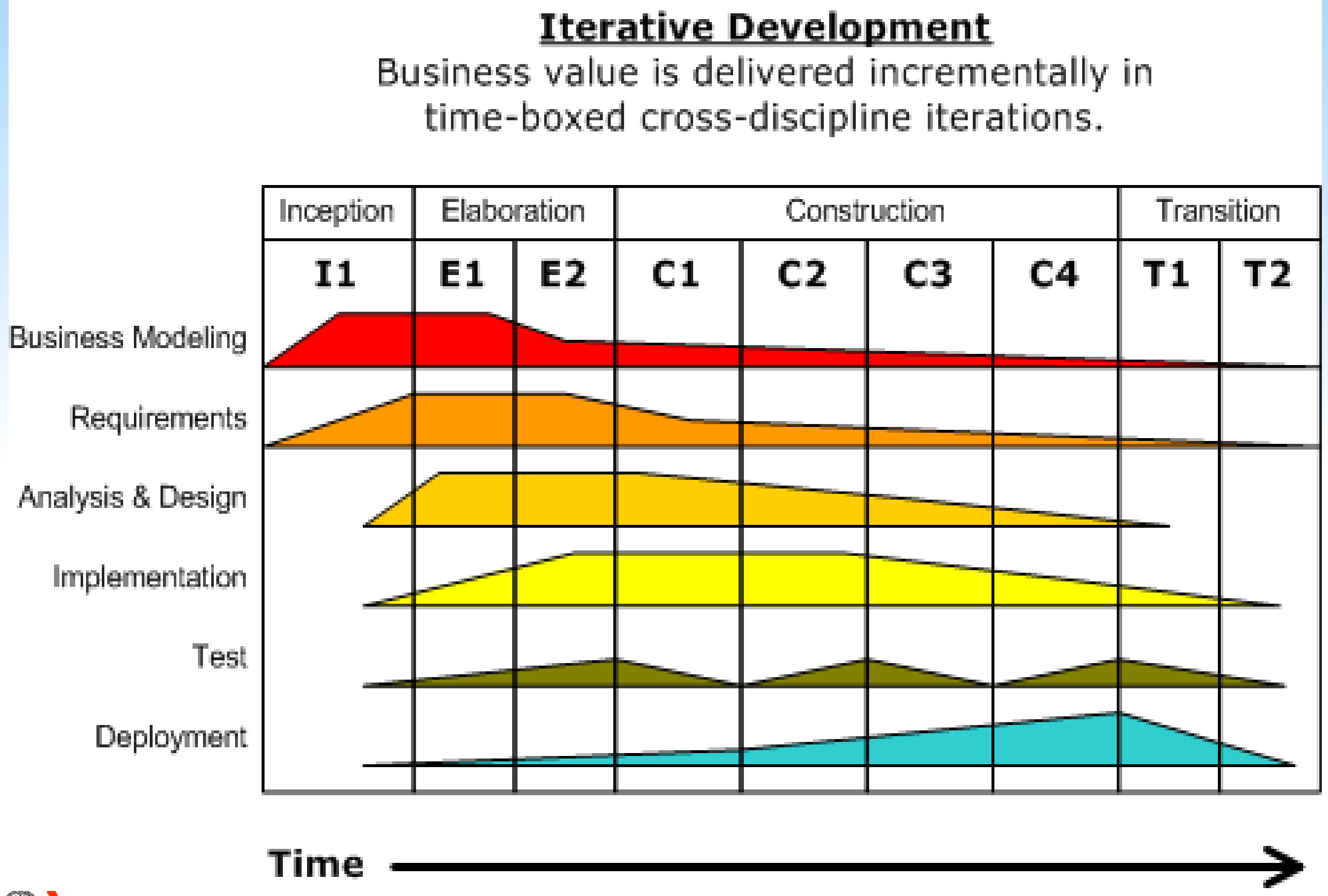
Key points

- Processes should include activities to cope with change. This may involve a prototyping phase that helps avoid poor decisions on requirements and design.
- Processes may be structured for iterative development and delivery so that changes may be made without disrupting the system as a whole.
- The Rational Unified Process is a modern generic process model that is organized into phases (inception, elaboration, construction and transition) but separates activities (requirements, analysis and design, etc.) from these phases.

Rational Unified Process (RUP)

- An iterative software development process framework created by the Rational Software Corporation (division of IBM since 2003)
- not a single concrete prescriptive process, but rather an adaptable process framework, intended to be tailored by the development organizations and software project teams that will select the elements of the process that are appropriate for their needs.

UCSC


Iterative Development
Business value is delivered incrementally in time-boxed cross-discipline iterations.
Inception
Elaboration
Construction
Transition
I1
E1
E2
C1
C2
C3
C4
T1
T2
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Time

UCSC

Rational Unified Process (RUP)

- Activities emphasize the creation and maintenance of models rather than paper documents.
- Development under RUP is architecture centric. The process focus on the early development and baseline of a software architecture (component based)
- The RUP places strong emphasis on building systems based on a thorough understanding of how the delivered system will be used.

Rational Unified Process (RUP)

- History
  - The roots of Rational Process go back to the original spiral model of Barry Boehm.
  - The Rational Approach was developed at Rational Software in the 1980s and 1990s.

UCSC

Rational Unified Process (RUP)

- History
  - In 1995 Rational Software acquired the Swedish Company Objectory AB.
  - The Rational Unified Process was the result of the merger of the Rational Approach and the Objectory process developed by Objectory founder Ivar Jacobson.
  - The first results of that merger was the Rational Objectory Process, designed to an Objectory-like process, but suitable to wean Objectory users to the Rational Rose tool. When that goal was accomplished, the name was changed.

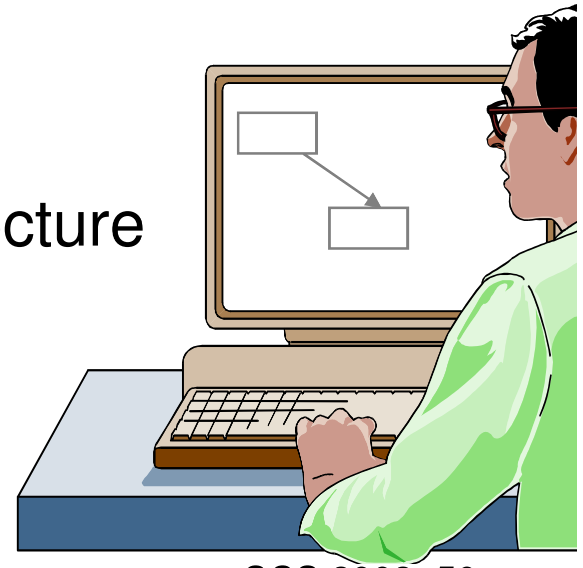
UCSC

Rational Unified Process (RUP)

- History
  - The first version of the Rational Unified Process, version 5.0, was released in 1998.
  - The chief architect was Philippe Kruchten.
  - The version of RUP (7.0) - Jan. 2007

Rational Unified Process (RUP)

- Principles of Business-Driven Development
  - RUP is based on a set of six best practices for modern software engineering.
    - Develop iteratively
    - Manage requirements
    - Employ a component-based architecture
    - Model software visually
    - Continuously verify quality
    - Control changes



UCSC

Rational Unified Process (RUP)

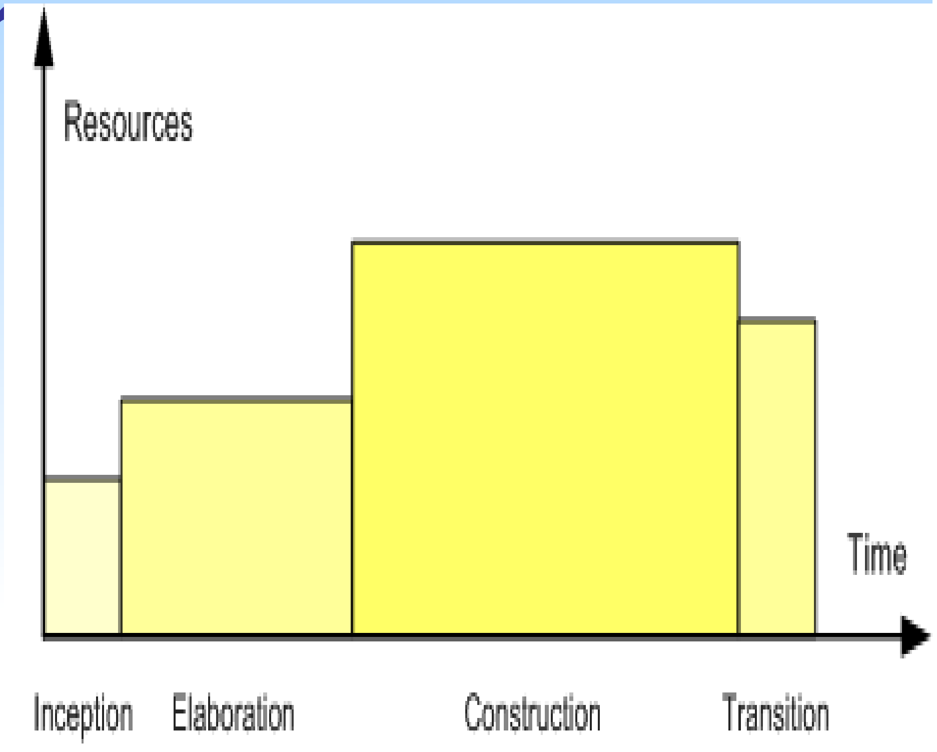
Key Characteristics

- **The Rational Unified Process encourages quality control and risk management.**

  <u>Quality assessment</u> is built into the process, in all activities and involving all participants, using objective measurements and criteria.

  <u>Risk management</u> is also built into the process, so that risks to the success of the project are identified and attacked early in the process, when there is time to react.

Rational Unified Pr



- The RUP lifecycle organizes the tasks into phases and iterations

A project has four phases:

- **Inception phase**
- **Elaboration phase**
- **Construction phase**
- **Transition phase**

Rational Unified Process

The Rational Unified Process consists of the following four phases

- **Inception - Establish the business case for the project**
- **Elaboration - Establish a project plan and a sound architecture**
- **Construction - Grow the system**
- **Transition - Supply the system to its end users**

Inception Phase

- During this phase, you *establish the business case* for the system and *define the project's scope.*
- The business case includes success criteria, risks assessment, estimates of the resources needed and a phase plan showing a schedule of major millstones.

Rational Unified Process

Inception Phase

- During inception, it’s common to create an executable prototype that serves as a proof of concept.
- At the end of the inception phase, you examine the life cycle objectives of the project and decide whether to proceed with full- scale development

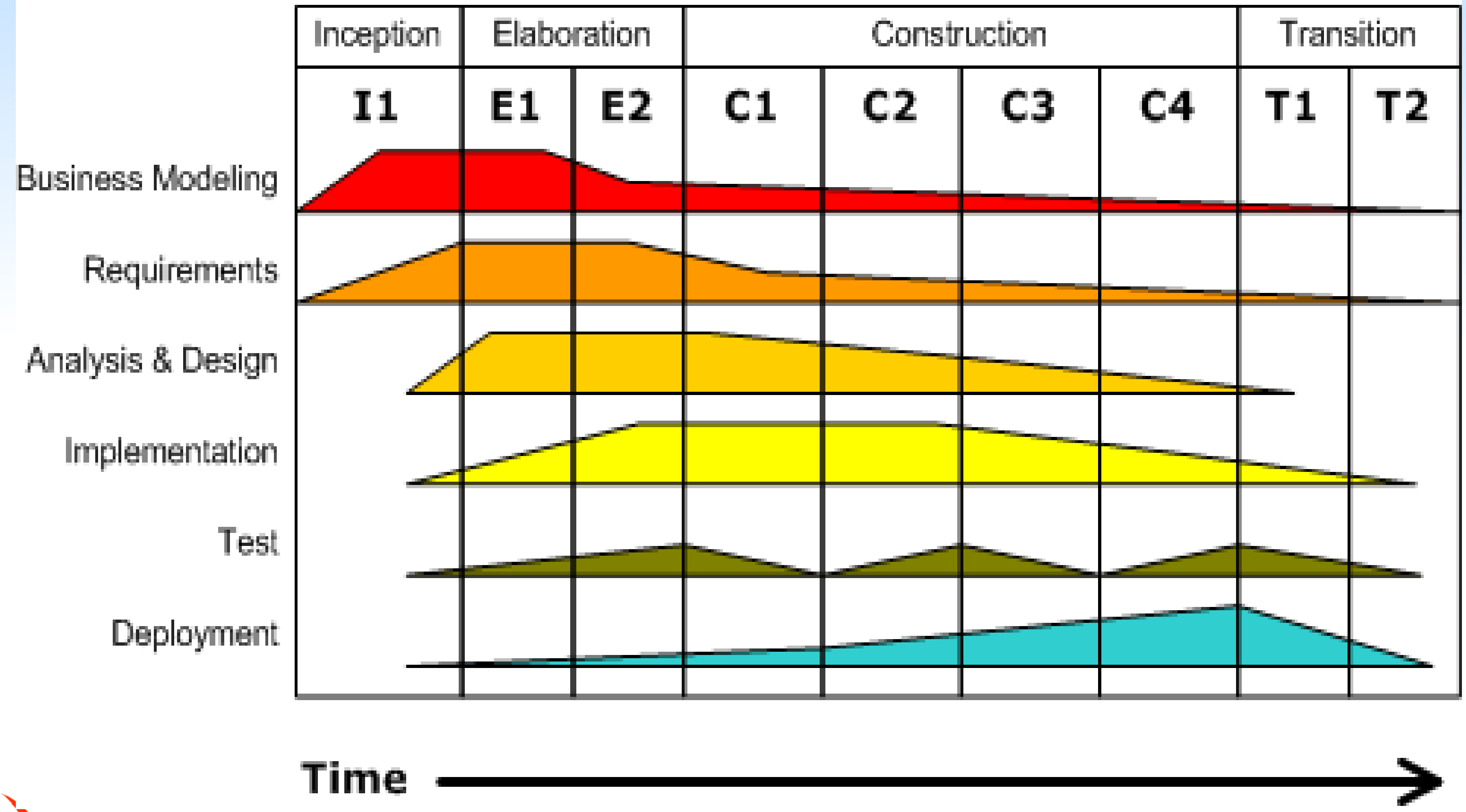
Rational Unified Process

Inception Phase

- Some inspection tasks include determining use cases, actors and draw Use Case diagrams.
- The Use Case diagrams can be presented to users to validate that diagrams are a comprehensive view of the system features.

Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.



Elaboration Phase

- During this phase the problem domain analysis is made and the architecture of the project gets its basic form.

- This phase must pass the Lifecycle Architecture Milestone by meeting the following criteria:
  - A use-case model in which the use-cases and the actors have been identified and most of the use-case descriptions are developed. The use-case model should be 80% complete.

Elaboration Phase

- A description of the software architecture in a software system development process.
- An executable architecture that realizes architecturally significant use cases.
- Business case and risk list which are revised.
- A development plan for the overall project.
- Prototypes that demonstratively mitigate each identified technical risk.

Elaboration Phase

- To verify the architecture, you implement a system that demonstrate the architectural choices and executes significant use cases.
- At the end of the elaboration phase, you examine the detailed system objectives and scope, the choice of architecture, and the resolution of major risks.

Elaboration Phase

- Since elaboration is the detailing of the system requirement, Use Case model might require updating.
- As the flow of processing is detailed, Sequence and Collaboration diagrams help illustrate the flow.
- They also help design objects that will be required for the system.
- Class diagrams and the State Transition diagrams are created during this phase.

Construction Phase

During the construction phase,

- all remaining components and application features are developed and integrated into the product, and
- all features are tested thoroughly.
- Emphasis is placed on managing resources and controlling operations to optimise costs, schedules, and quality.

Rational Unified Process

Construction Phase

- Construction is the stage in which majority of the coding for the project is done.
- Components diagrams are created to show the compile time dependencies between the components.
- After code has been created by the developers, the model can be synchronized with the code through reverse engineering.

UCSC

Transition Phase

➢During the transition phase, you deploy the software to the user community.

➢Once the system has been put into hands of its end users, issues often arise that require additional development in order to adjust the system, correct some undetected problems or finish some features that have been postponed.

Transition Phase

- This phase typically starts with a beta release of the system, which is then replaced with the production system.
- Parallel operation with the legacy system that the prototype is replacing.
- Conversion of operational databases
- Training of users and maintainers.
- Rollout the product to the marketing, distribution, and sales teams.

Disciplines and workflows

Within each phase are a number of iterations.

- An iteration represents a complete development cycle, from requirements capture in analysis to implementation and testing,

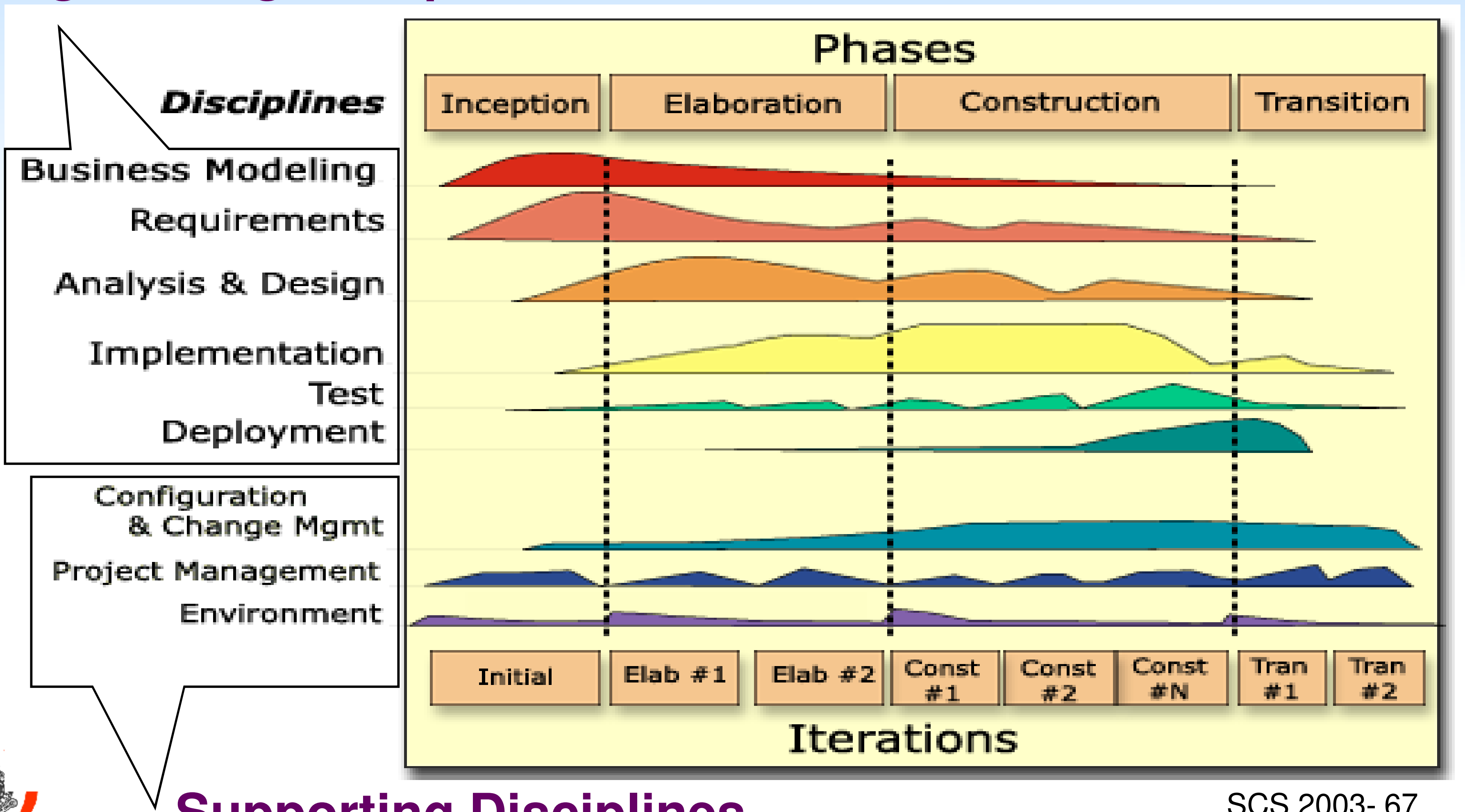
-**that results in the release of an executable product constituting a subset of the final project under development,**

-**which then is grown incrementally from iteration to iteration to become the final system.**

Rational Unified Process

Disciplines and workflows

Engineering Disciplines



Supporting Disciplines

Disciplines and workflows

- RUP is based on a set of
  - building blocks, or
  - content elements, describing
    - what is to be produced,
    - the necessary skills required and
    - the step-by-step explanation describing how specific development goals are achieved.

Rational Unified Process

Disciplines and workflows

- The main building blocks, or content elements,
  - Roles (who) – A Role defines a set of related skills, competences, and responsibilities.
  - Work Products (what) – A Work Product represents something resulting from a task, including all the documents and models produced while working through the process.
  - Tasks (how) – A Task describes a unit of work assigned to a Role that provides a meaningful result.

Rational Unified Process

Within each iteration, the tasks are categorized into nine Disciplines:

Disciplines and workflows

Engineering Disciplines:

- **Business modeling discipline**
- **Requirements discipline**
- **Analysis and design discipline**
- **Implementation discipline**
- **Test discipline**
- **Deployment discipline**

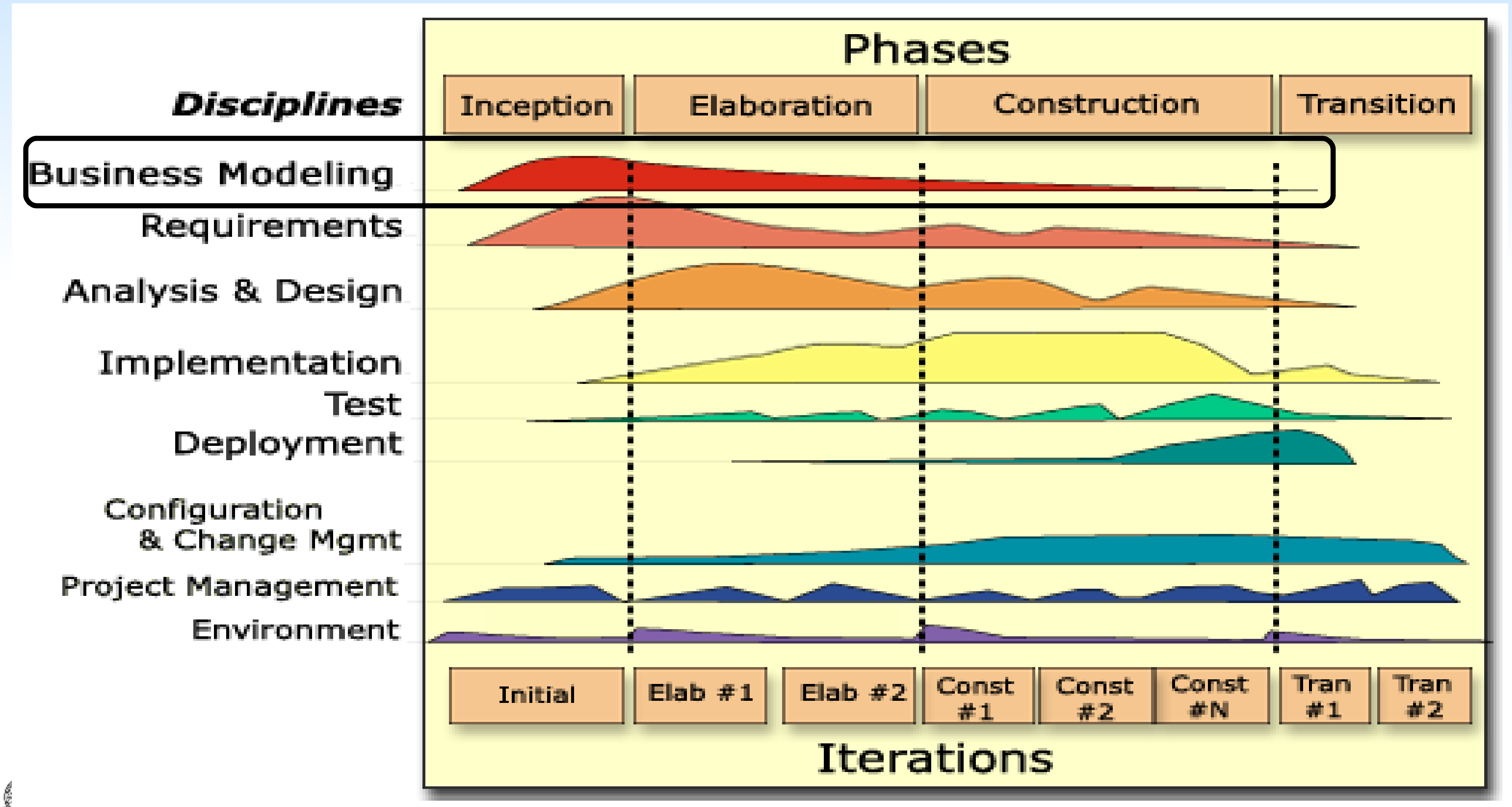
Supporting Disciplines:

- **Configuration and change management discipline**
- **Project management discipline**
- **Environment discipline**

Disciplines and workflows

Rational Unified Process

Business Modeling



Rational Unified Process

Disciplines and workflows

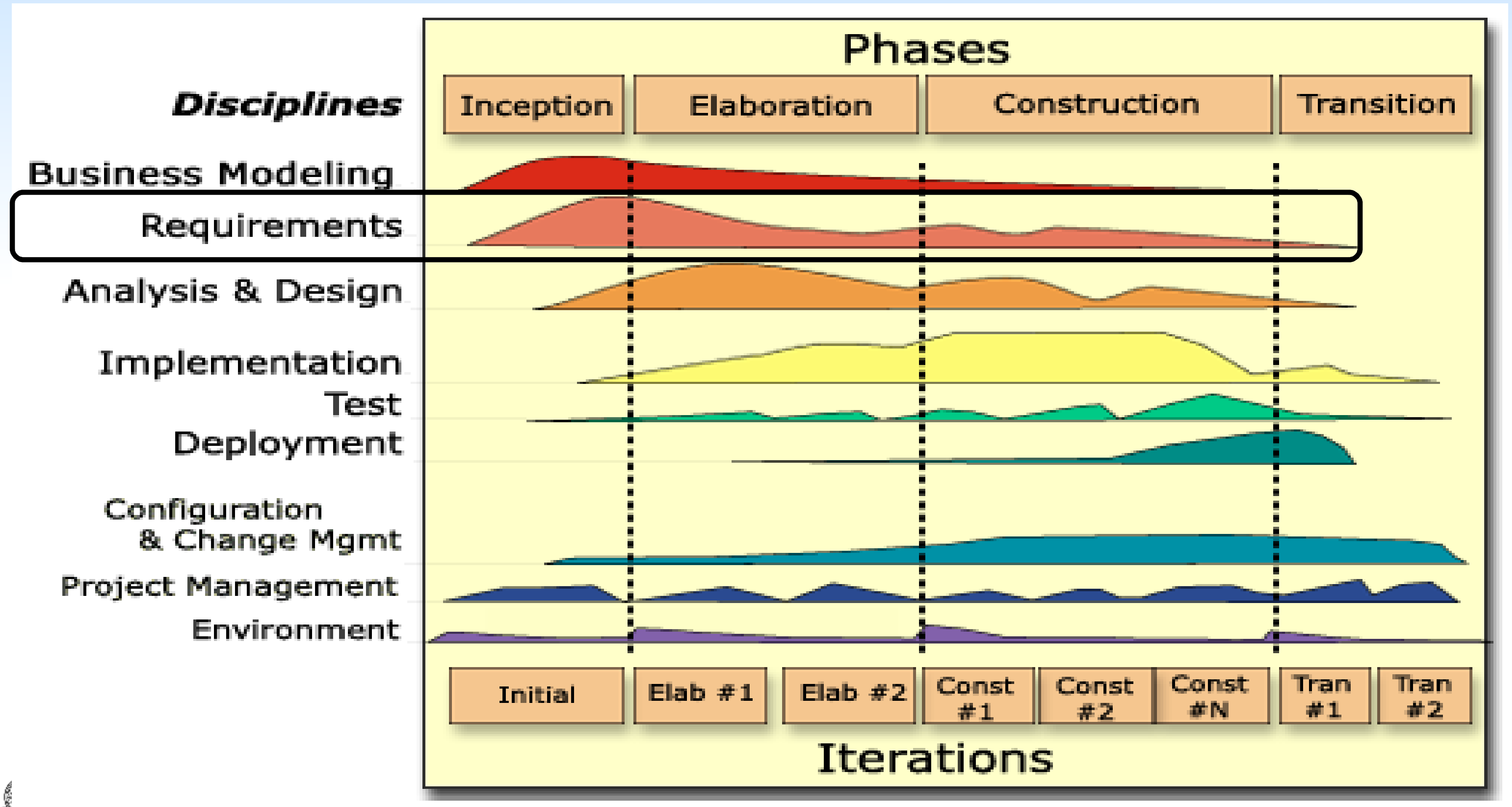
- **Business modeling discipline**
  - **establish a better understanding and communication channel between business engineering and software engineering.**
  - **explains how to describe a vision of the organization in which the system will be deployed and how to then use this vision as a basis to outline the process, roles and responsibilities.**

UCSC

Disciplines and workflows

Rational Unified Process

Requirements Discipline



UCSC

Rational Unified Process

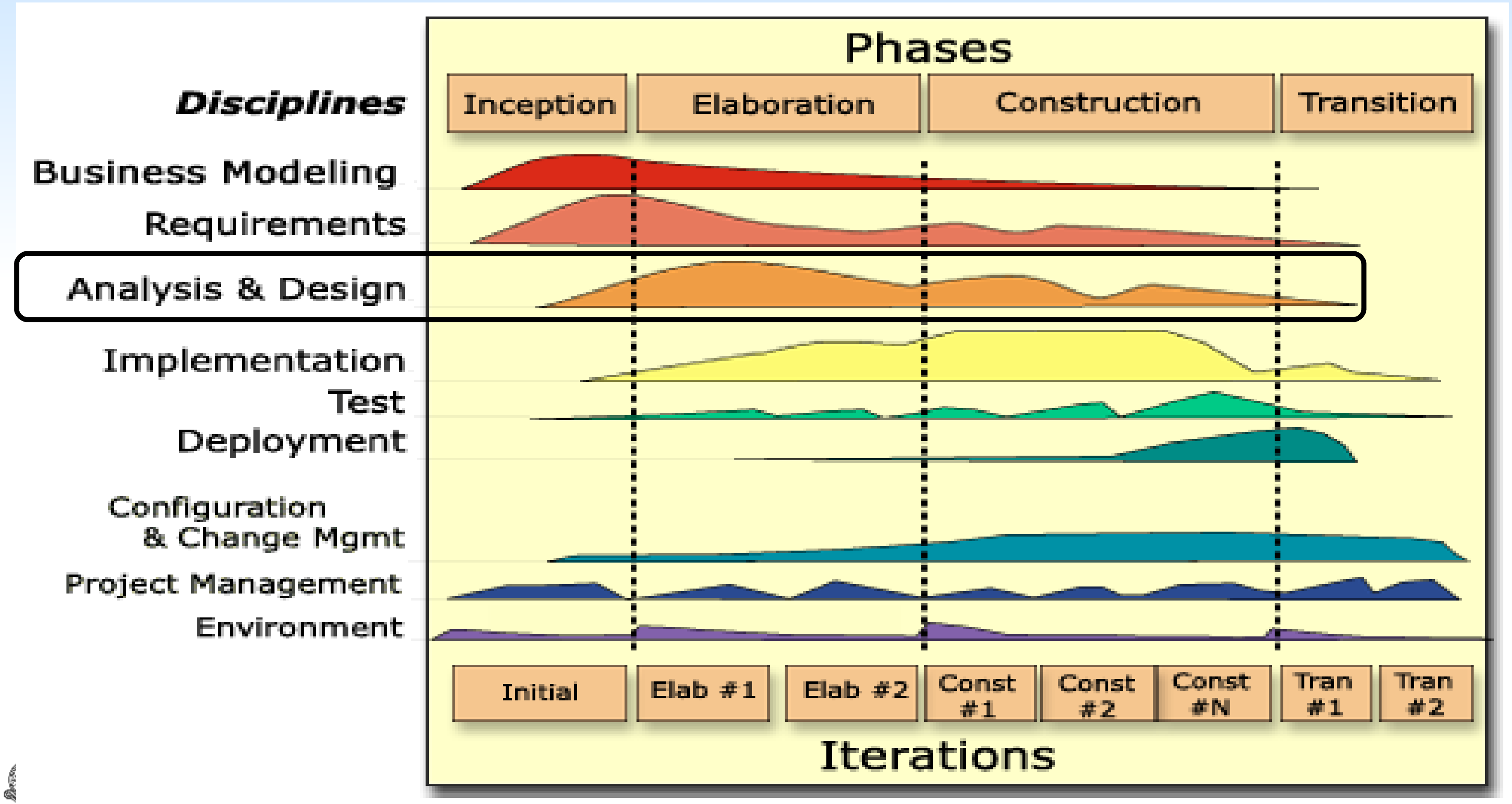
Disciplines and workflows

- Requirements discipline
  - The goal of the Requirements is to describe what the system should do and allows the developers and the customer to agree on that description.

Disciplines and workflows

Rational Unified Process

Analysis and Design Discipline



UCSC

Rational Unified Process

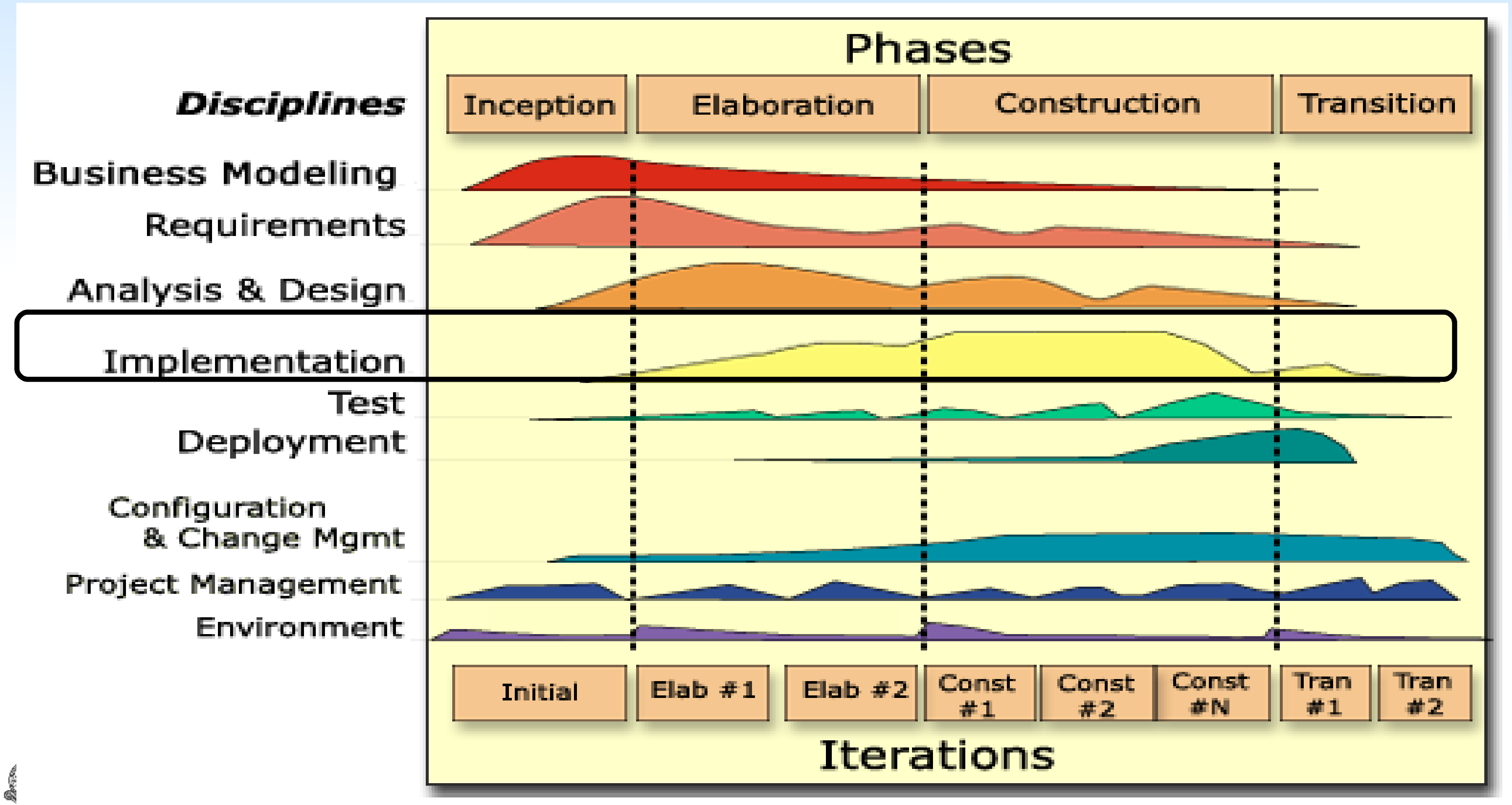
Disciplines and workflows

- Analysis and design discipline
  - The goal of analysis and design is to show how the system will be realized in the implementation phase. The aim is to build a system that:
    - Performs—in a specific implementation environment—the tasks and functions specified in the use-case descriptions.
    - Fulfills all its requirements.
    - Is easy to change when functional requirements change.

Rational Unified Process

Disciplines and workflows

Implementation Discipline



UCSC

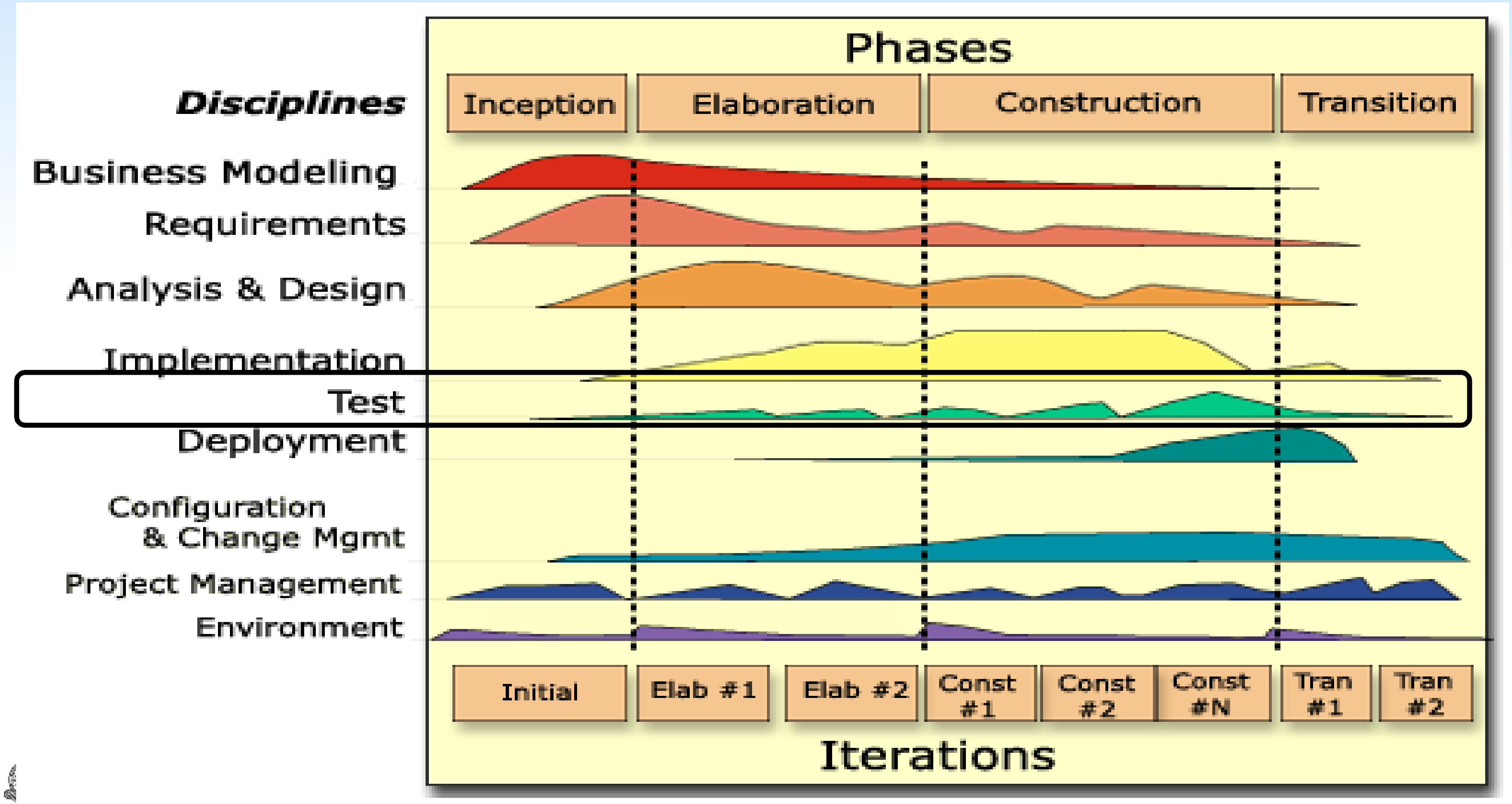
Disciplines and workflows

- Implementation discipline
  - The purposes of implementation are:
    - **To define the organization of the code, in terms of implementation subsystems organized in layers.**
    - **To implement classes and objects in terms of components (source files, binaries, executables, and others).**
    - **To test the developed components as units.**
    - **To integrate the results produced by individual implementers (or teams), into an executable system.**

Disciplines and workflows

Rational Unified Process

Test Discipline



UCSC

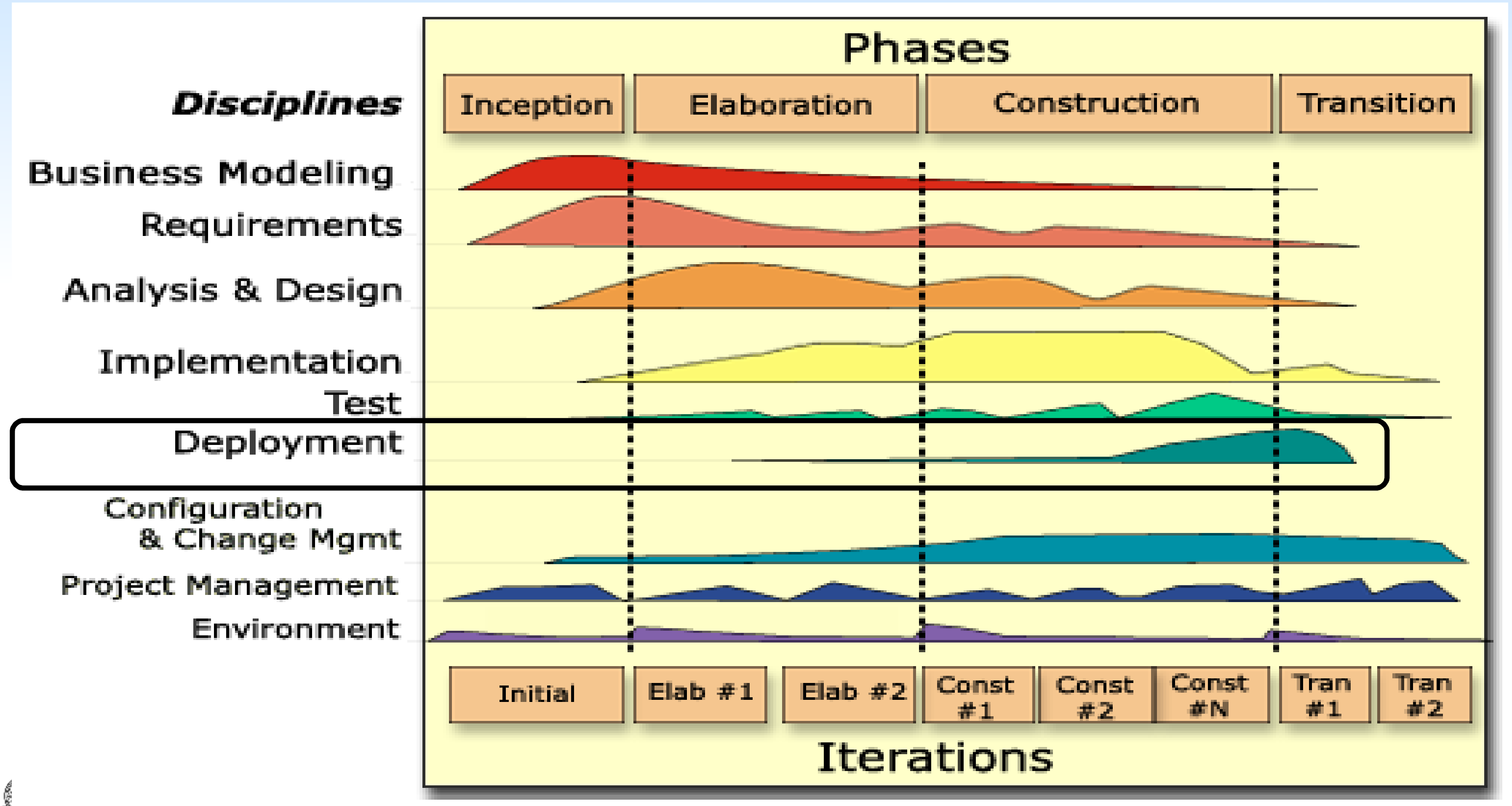
Disciplines and workflows

- Test discipline
  - The purposes of the Test discipline are:
    - To verify the interaction between objects.
    - To verify the proper integration of all components of the software.
    - To verify that all requirements have been correctly implemented.
    - To identify and ensure that defects are addressed prior to the deployment of the software

Disciplines and workflows

Rational Unified Process

Deployment Discipline

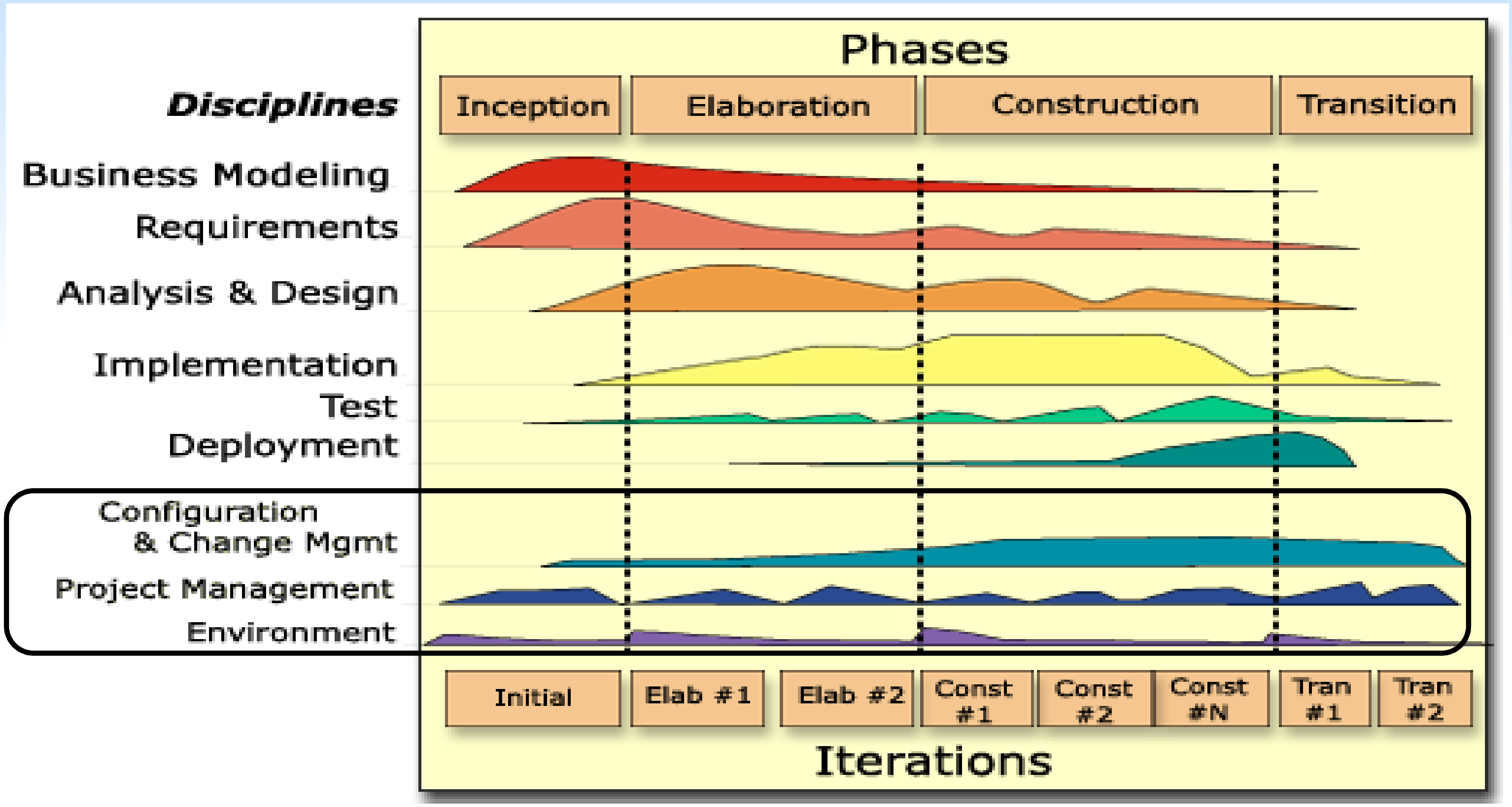


UCSC

Disciplines and workflows

- Deployment discipline
  - The purpose of deployment is to successfully produce product releases, and deliver the software to its end users. It covers a wide range of activities including:
    - Producing external releases of the software
    - Packaging the software
    - Distributing the software
    - Installing the software
    - Providing help and assistance to users

Supporting Disciplines and workflows



Rational Unified Process

Supporting Disciplines and workflows

- **Configuration and Change management discipline**
  - The Change Management discipline in RUP deals with three specific areas:
    - Configuration management: responsible for the systematic structuring of the products
    - Change request management: keeps track of the proposals for change
    - Status and measurement management:

Supporting Disciplines and workflows

- **Project management discipline**
  - Project planning in the RUP occurs at two levels.
    - a coarse -grained or **Phase** plan : describes the entire project, and
    - a series of fine-grained or **Iteration** plans : describe the iterations.

Rational Unified Process

Supporting Disciplines and workflows

- **Project management discipline**
  - does not attempt to cover all aspects of project management e.g.
    - Managing people: hiring, training, coaching
    - Managing budget: defining, allocating, and so forth
    - Managing contracts, with suppliers and customers

Rational Unified Process

Supporting Disciplines and workflows

- **Project management discipline**
  - focuses mainly on the important aspects of an iterative development process
    - Risk management
    - Planning an iterative project, through the lifecycle and for a particular iteration
    - Monitoring progress of an iterative project, metrics

Rational Unified Process

Supporting Disciplines and workflows

- **Environment discipline**
  - focuses on the activities necessary to configure the process for a project.
  - describes the activities required to develop the guidelines in support of a project.
  - provide the software development organization with the software development environment-both processes and tools-that will support the development team.