# COMP 3711

# (OOA and OOD)

# Iterative UP – Inception - Eleboration

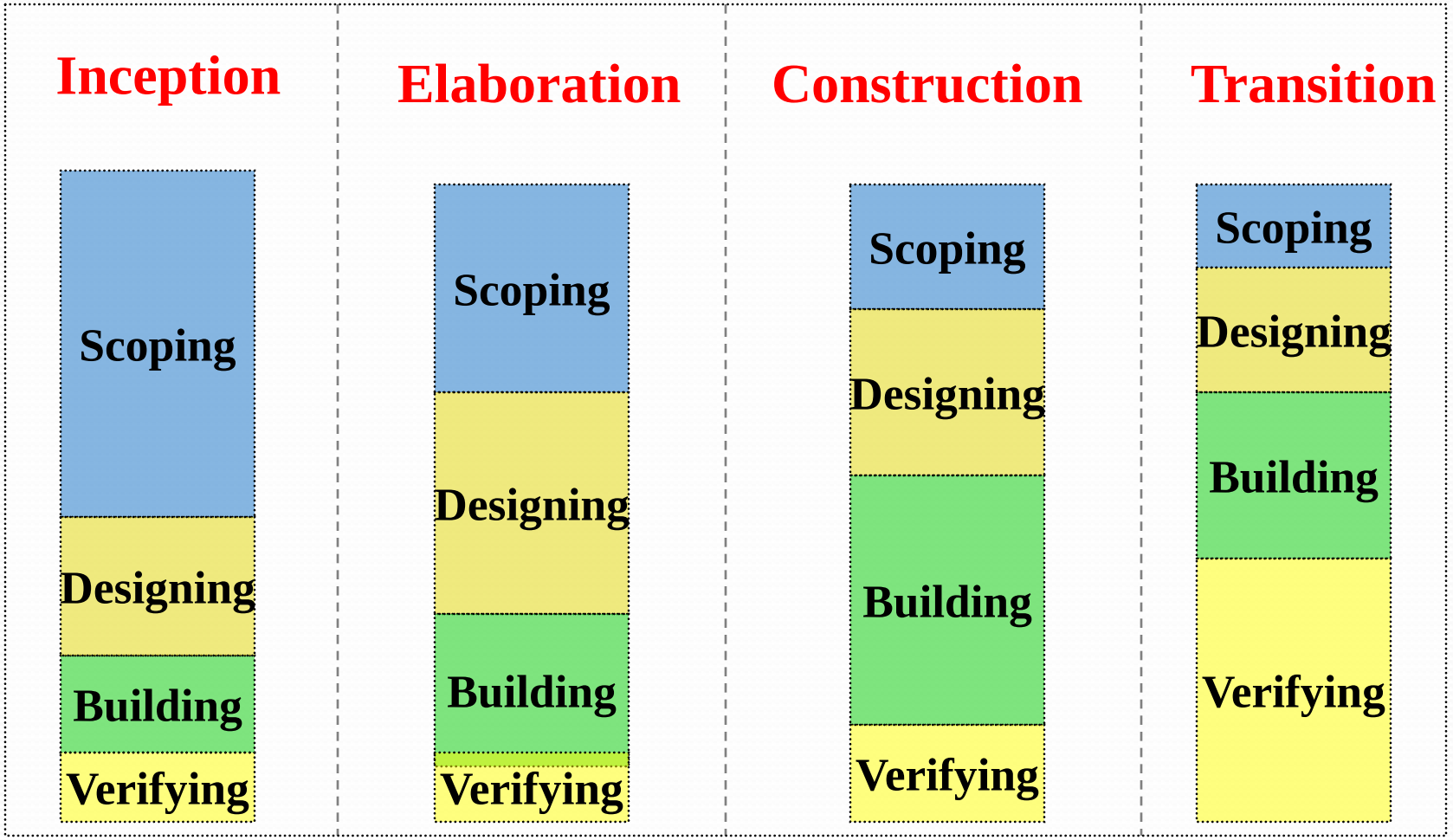
Larman Chapter 4-5, 8

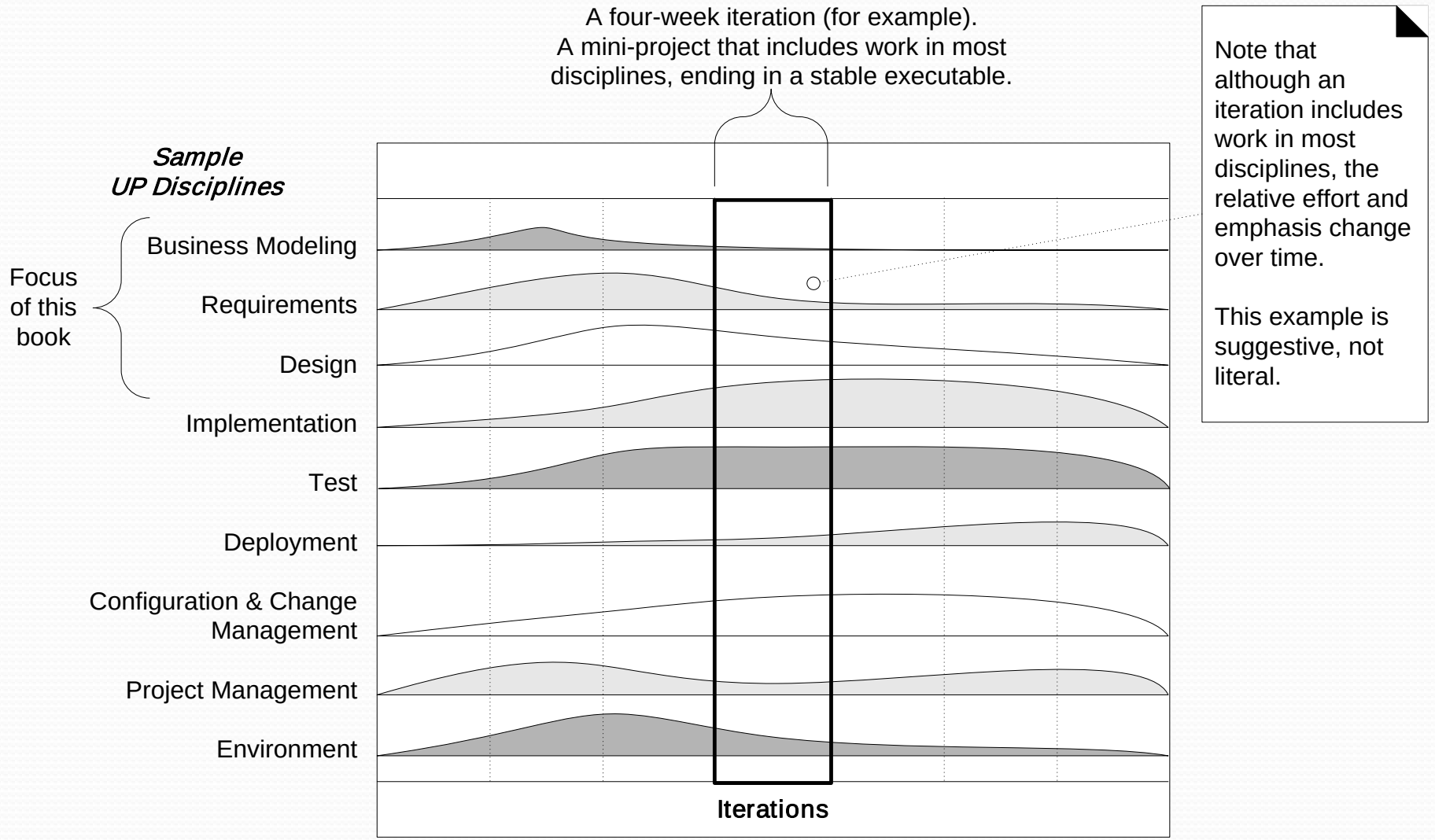
# Unified Process - UP

| Requirement | Design | Develop | Implement |
| --- | --- | --- | --- |
| **Inception** | **Elaboration** | **Construction** | **Transition** |
| Approximate vision, business case, scope | Refine vision, core architecture, refine scope | Implementation of low risk core architecture, refine scope | Beta test and deployment |

**Iterative Development means not all requirements are implemented at once**

# Unified Process (UP) - Iterative



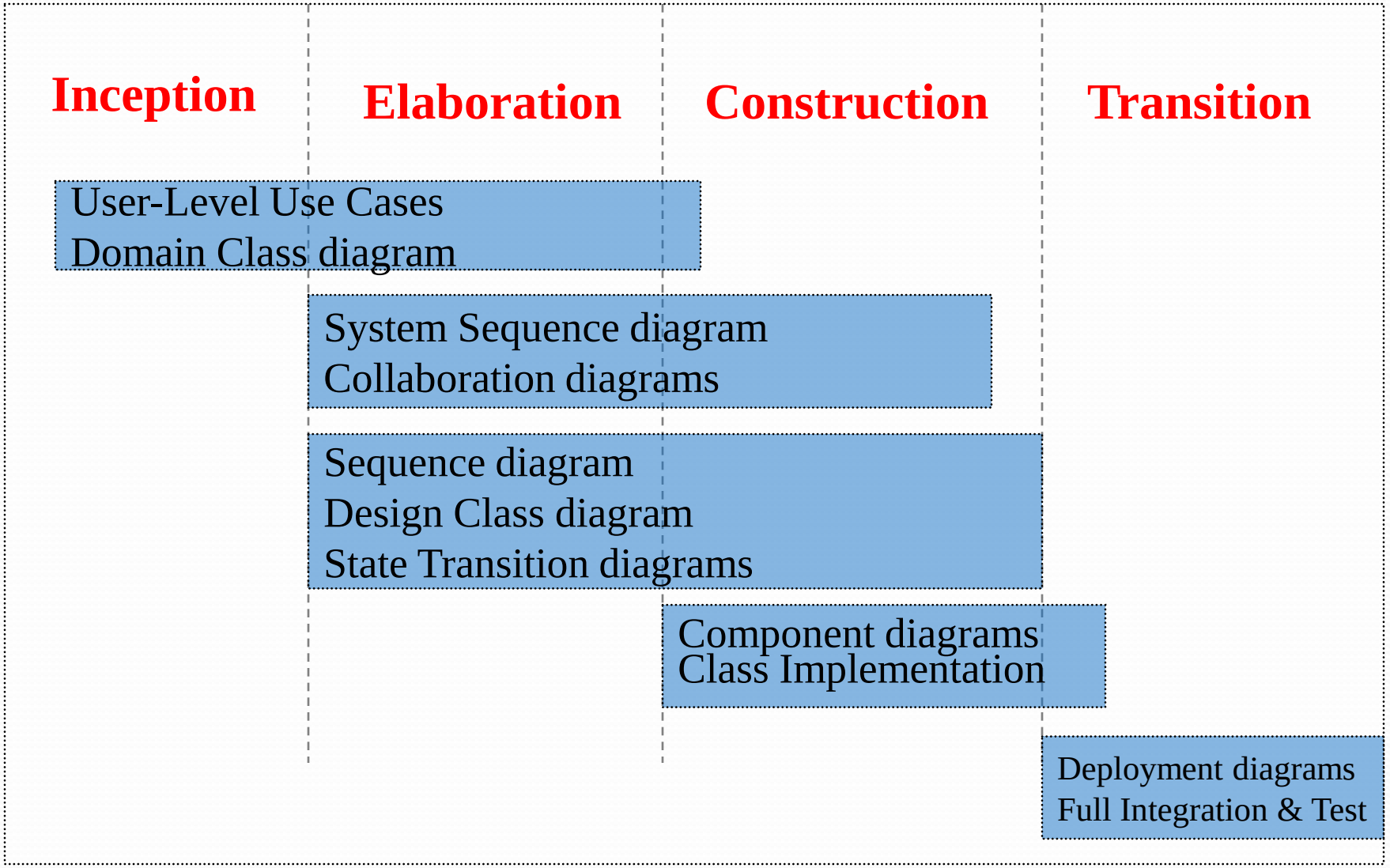



Larman Fig 2.7

# The Unified Process (UP)

- Reinforces six best practices
  - Develop iteratively
  - Define and manage system requirements
  - Use component architectures
  - Create visual models
  - Verify quality
  - Control changes

# UML IN UP (Iteration)



# Purposes of Analysis and Design

- To evolve the requirements and transform them into a design of the system to-be
- To evolve a robust architecture for the system
- To adapt the design to match the implementation environment, designing it for performance

# Analysis Versus Design

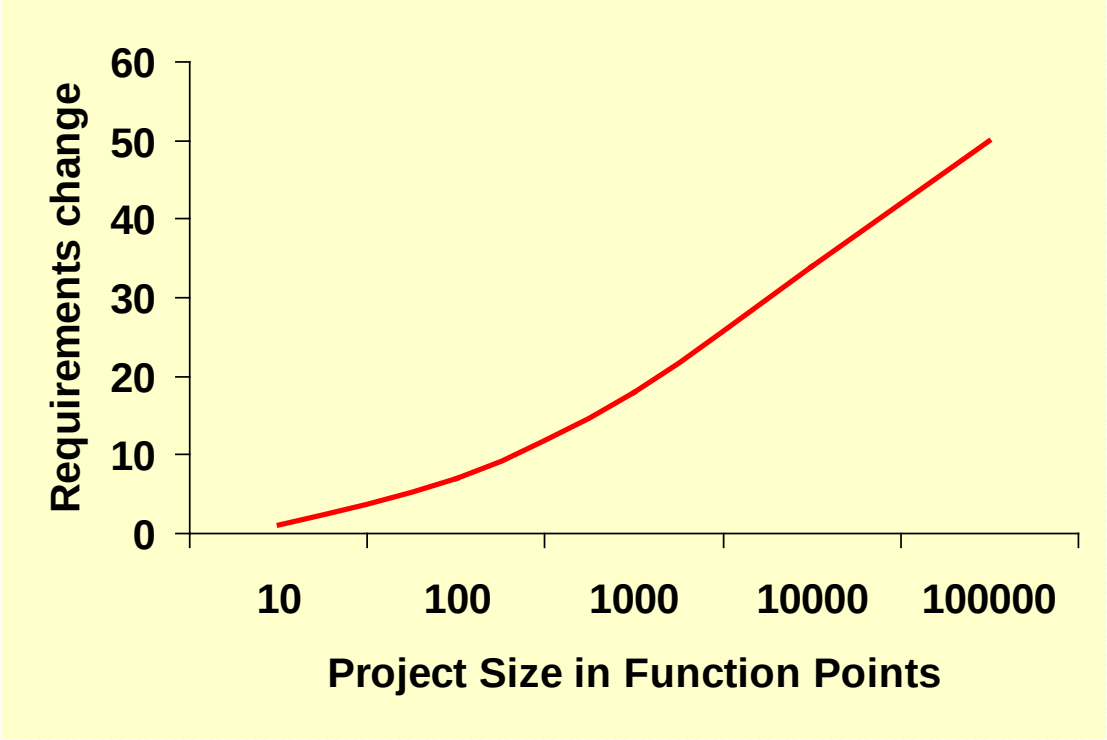
- Analysis
  - Focus on understanding the problem domain
  - Idealized design
  - Behavior
  - Functional requirements
  - System structure
- Design
  - Focus on understanding the solution
  - Operations and Attributes
  - Performance
  - Close to real code
  - Object lifecycles
  - Non-functional requirements
  - A large model


Larman 1.4

# Evolutionary Requirements

# Faulty Assumption 1: Requirements can be Fairly Accurate

Jones, 1997. Based on 6,700 systems.



# Faulty Assumption 2: Requirements are Stable

- The market changes—constantly.
- The technology changes.
- The goals of the stakeholders change.

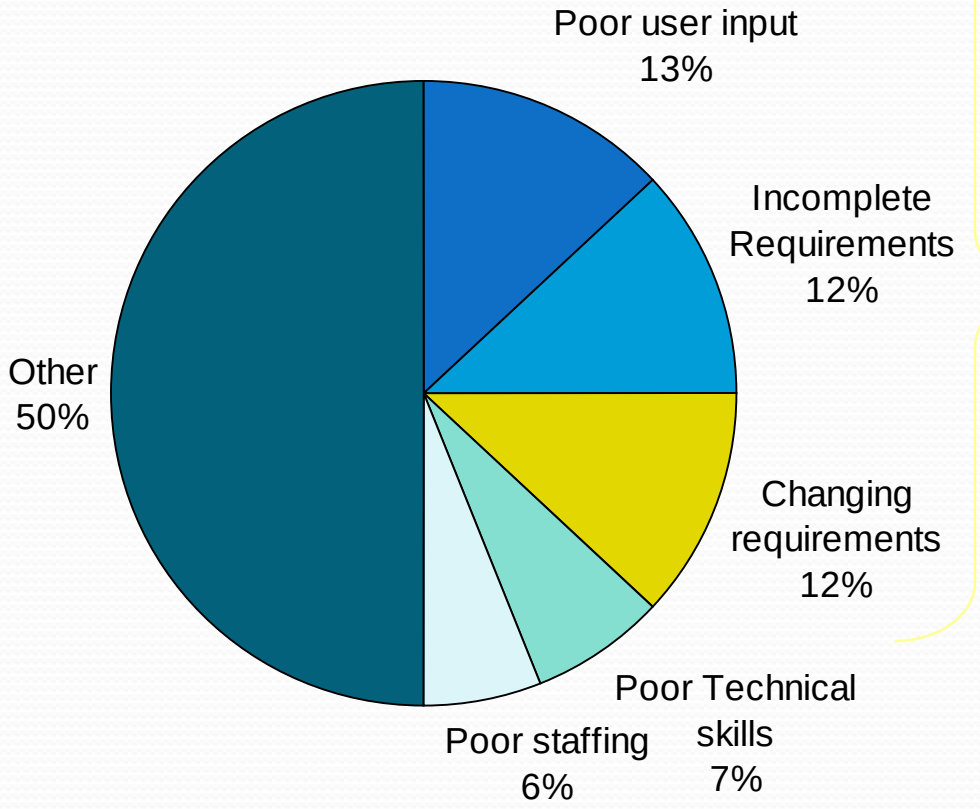
# Faulty Assumption 3: The Design can be Done, before Programming

- Ask a programmer.
- Requirements are incomplete and changing.
- Too many variables, unknowns, and novelties.
- A complete specification must be as detailed as code itself.
- Software is very “hard”.
  - Discover Magazine, 1999: Software characterized as the most complex “machine” humankind builds.

# The Importance of Requirements

- Requirements are often taken for granted.
- Users know what they want, right?
- Well not exactly
- Figuring out exactly what a system should be doing is a major undertaking

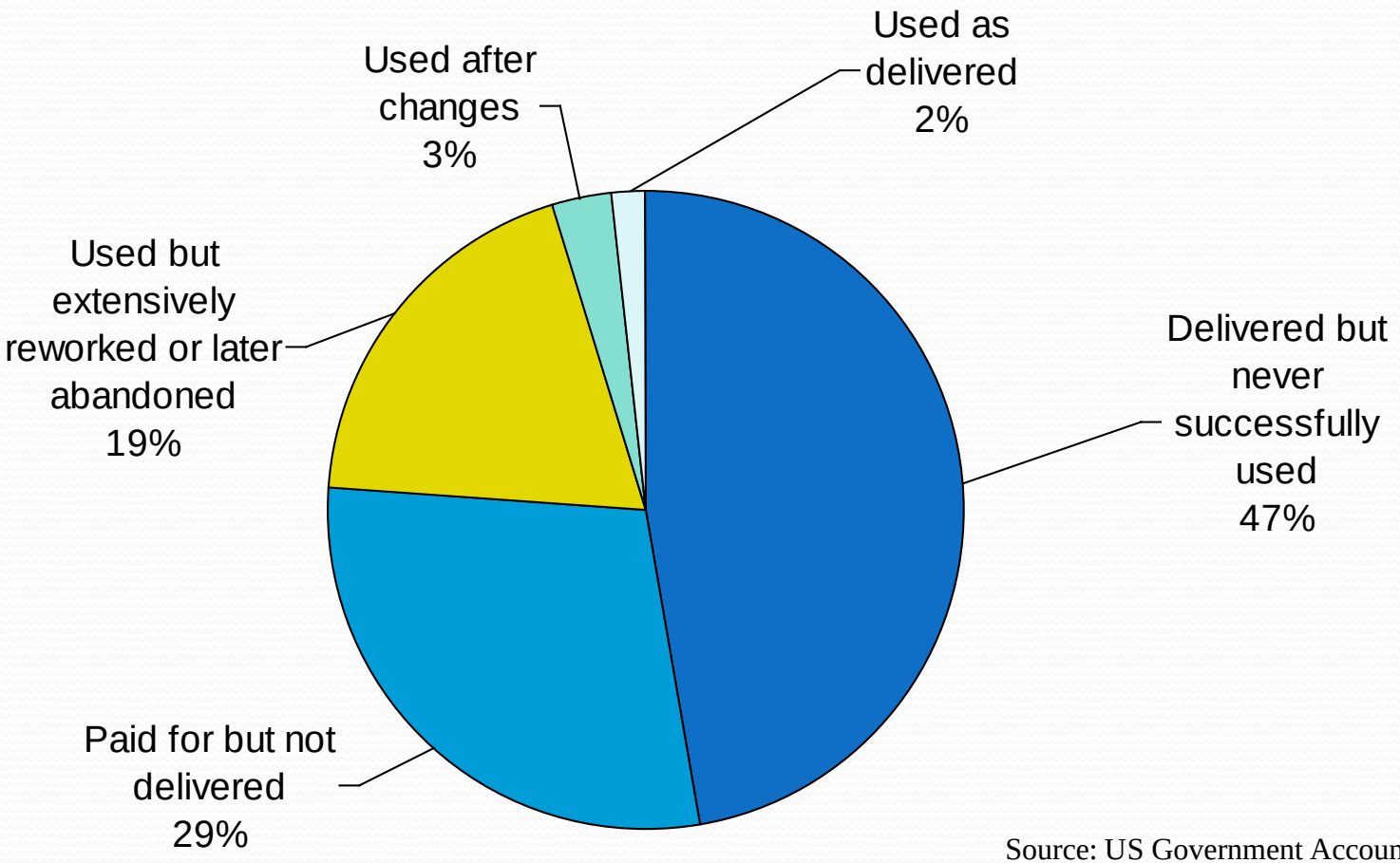
# Factors on challenged software projects




37% relate to problems with requirements

# The Rate of Failure

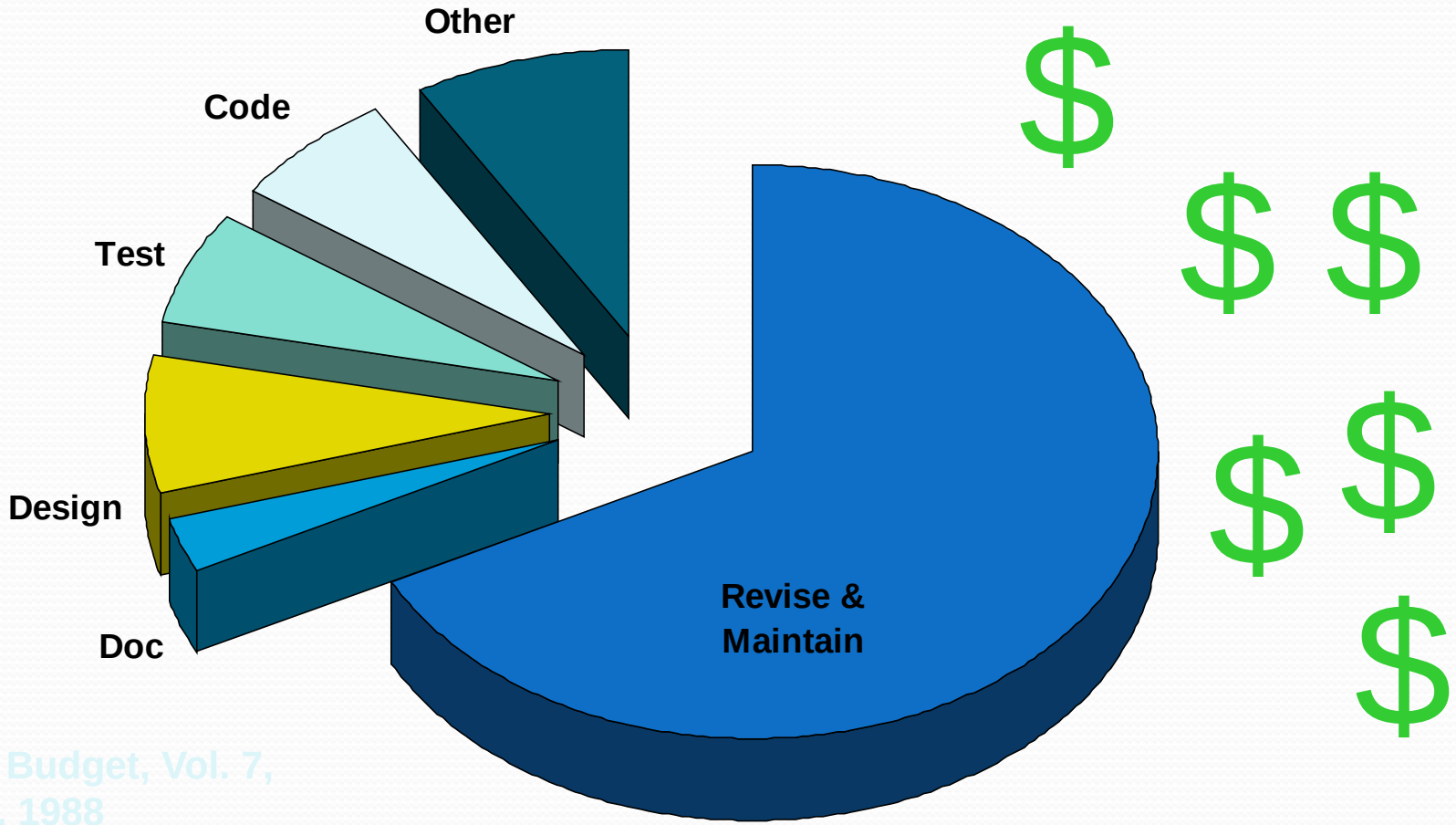
Randomly selected U.S. government software projects:




Source: US Government Accounting Office. Report FGMSD-80-4.

# The Cost of Change

Strategic rational system development plans are based on the complete cost of a system, not solely on development costs.




Source: DP Budget, Vol. 7, No. 12, Dec. 1988

# The Cost of Change

- An AT&T study indicated that, on average

Business Rules change at the rate of 8% per month. This would be quite a large change over a year.

# Types of requirements(FURPS)

- FURPS
  - (F)unctional - features, capabilities, security
  - (U)sability - human factors, help, documentation
  - (R)eliability - frequency of failure, recoverability, predictability
  - (P)erformance - response times, throughput, accuracy, availability, resource usage
  - (S)upportability - adaptability, maintainability, internationalization, configurability

# Types of requirements(FURPS+)

- + indicates ancillary and sub-factors, such as:
  - Implementation - resource limitations, languages, tools, hardware
  - Interface – constraints imposed by interfacing with other systems
  - Packaging
  - Legal – licensing, etc…

# Inception - Analysis Phase

- Gather information
- Define requirements
- Prioritize requirements
- Discovery prototypes
- Evaluate alternatives
- Make recommendation

- Current state, docs?
- Required functions?
- Most important?
- Feasible?
- Best approach?
- Go or No Go?

# Requirement Oriented Artifacts @ Inception

- *JAD sessions (requirement workshop)*
- *Stories*
- *Business Case (with defined Scope)*
- *Activity diagram*
- *Use Case document*
- *Use Case diagram*
- *Domain Class diagram*
- *High-level candidate architecture and components*
- *Proof–of-concept (Discovery) prototypes*

# Activities in the Inception Phase

## During the this phase we:

- establish the business case for the system and define the project's scope.
- create a discovery 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

# Business Case

- The business case includes success criteria, risks assessment, estimates of the resources needed and a phase plan showing a schedule of major millstones.

# Objectives of the Inception Phase – Resolve the System Scope

- Define the scope of the proposed system.
- Draws a line around exactly what is to be within the proposed system and what is outside.
- Defines the external actors, which may be other systems or people which the system is to interact and it specifies at a high level the nature of this interaction.

# Resolve the System Scope – Example Questions

- Is what is to be within the system clear ?
- Are all the actors identified ?
- Is the general nature of the interfaces (user interfaces and communication protocols) to the actors identified ?
- Can what is within the scope stand by itself as a functioning system ?

# Resolve Ambiguities in the Requirements Needed in this Phase

- The requirements at the beginning of the inception phase may range from a broad vision to many pages of textual description.
- However, these initial requirements are likely to contain ambiguities. In the inception phase an effort is made to avoid these ambiguities.

# Resolve Ambiguities - Questions

- Has the limited number of use-case requirements (functional as well as non functional ) needed to reach the objectives of this phase been identified and detailed ?
- Have requirements been identified and detailed in the supplementary requirements?

# Establish a Candidate Architecture

Why should an organization focus on an architecture?

- Lets you gain and retain Intellectual Control over the project.
- Reuse
- Provides a basis for project management.

# Establish a Candidate Architecture –Questions?

- Does it meet user's needs?
- Is it likely to work ?

# Objectives of the Inception Phase

- Estimate Risks
  - Many difficult projects have failed because they encountered critical risks.
- Estimate the Cost and Schedule
  - Overall cost and schedule for the entire project.
- Identify the primary scenarios of behavior (use cases) of the system
  - These use cases will drive the systems functionality.

# Estimate Risks - Questions

- Have all the critical risks have been identified ?
- Have the identified risks been mitigated or is there a plan for mitigating them?

# Outcomes of the Inception Phase

- A vision document : general vision of the core project requirements, key features, and main constraints.
- A first version of a business( or domain) model that describes the context of the system .
- A first cut of the models representing a first version of the use-case model and domain model. Of the implementation model and test model, there may be something rudimentary. There is also a first version of the supplementary requirements.

# (conti…)

- A first draft of a candidate architecture description with outlines of views of the use case, analysis, design and implementation models
- Possibly a proof of concept exploratory prototype, demonstrating the use of the new system
- An initial risk list and use case ranking list
- The beginning of a plan for the entire project, including a general plan for the phases
- A first draft of business case, which includes: business context and success criteria (revenue projection, market recognition, project estimate)

# Other Artifacts in Inception

- Vision
- Supplementary Specifications
- Glossary
- Development Case

# Vision

- The vision serves to communicate the big ideas regarding:
  - Why the project was proposed?
  - What the problems are?
  - Who the stakeholders are?
  - What they need?
  - What the proposed solution looks like?
- Vision provides the contractual basis for the more detailed technical requirements.

# System Features

The Vision document should include a list of System Features

- An externally observable service provided by the system which directly fulfills a stakeholder need
- They should pass the linguistic test:

*The system shall do xxxxxx*

# Supplementary Specifications

Document requirements that are not captured in Use Cases.

- FURPS+ requirements - functionality, usability, reliability, performance, and supportability
- Reports
- Hardware and software constraints
- Development constraints
- Other design and implementation constraints
- Internationalization concerns
- Licensing and other legal issues
- Packaging
- Standards (technical, safety, quality)
- Physical environment (e.g. heat or vibration)
- Operational concerns (e.g. error handling or backups)
- Domain or business rules
- Any other domain-related information

# Supplementary Specifications - Constraints

- Constraints are not behaviors, but some other kind of restriction on the design or project.
- Restrictive in nature
- Early constraints are always a bad idea

E.g.
- Must use Oracle
- Must run on Linux

# Supplementary Specifications - Quality Attributes

- Quality of the system other than functionality
- Usability, reliability, etc …
- Two types:
  - Observable at execution (usability, reliability, performance, etc …)
  - Not observable at execution (supportability, testability, …)

# Supplementary Specs - Domain (Business) rules

- Rules that dictate how a domain or business may operate
  - E.g. company policies, physical laws, government laws, etc ….

# Glossary

- Also plays the role of a data dictionary
- Terms attributes could include:
  - Aliases
  - Description
  - Format (type, length, unit)
  - Relationship to other elements
  - Range of values
  - Validation rules
- Glossary for “atomic” and “composite terms

# Iterative Requirement Gathering

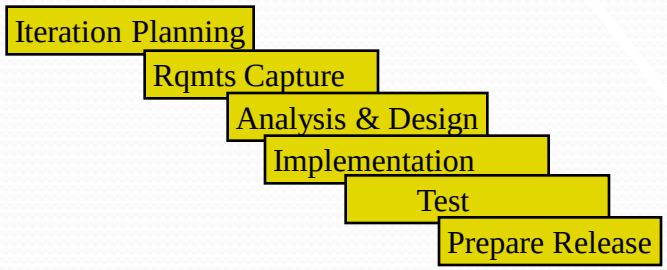
- What happened in Inception?
- What happened in Elaboration?

# Use Cases Drive the Iteration Process

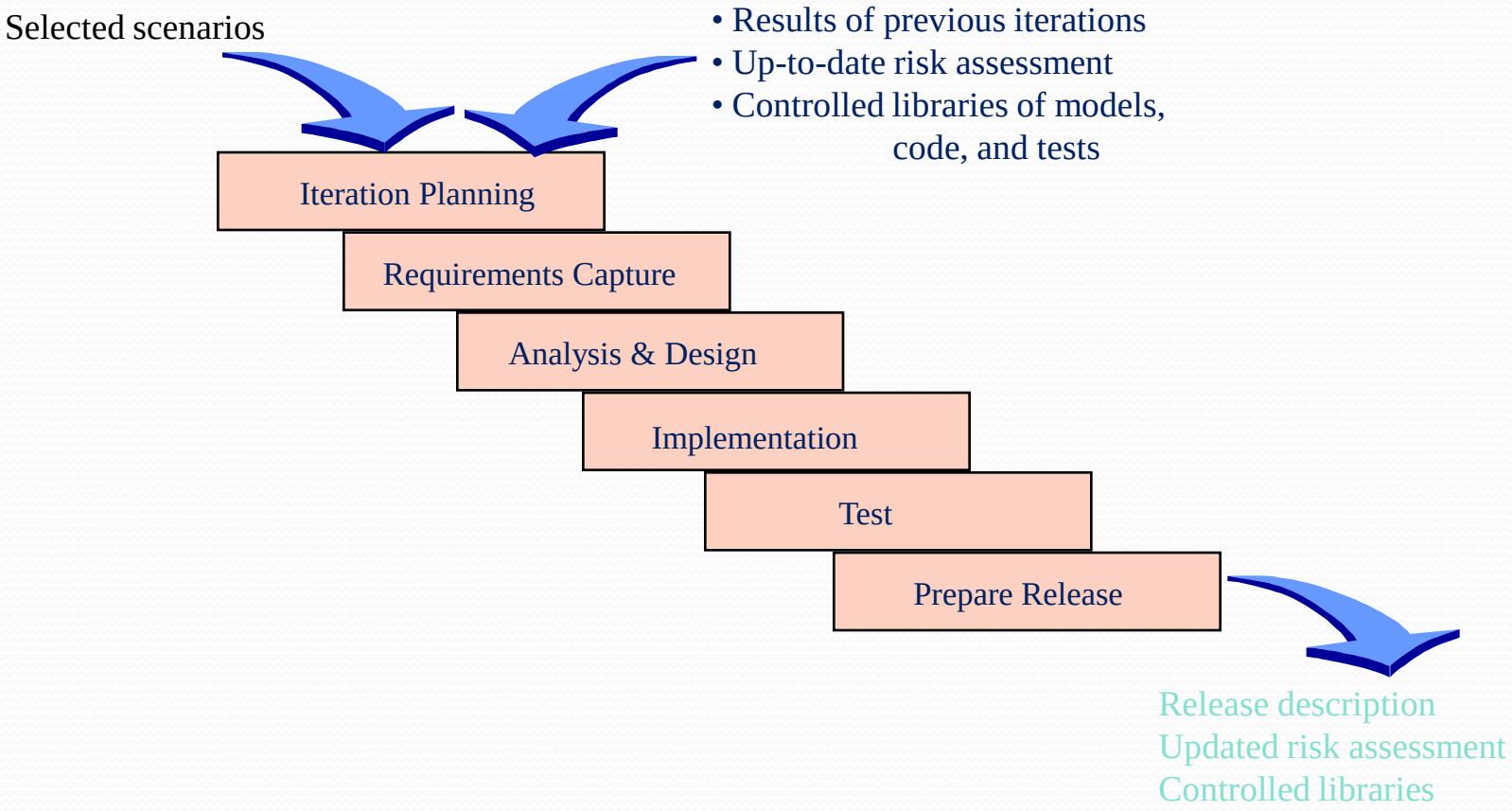
**Inception** → **Elaboration** → **Construction** → **Transition**

***Iteration 1*** → ***Iteration 2*** → ***Iteration 3***

**"Mini-Waterfall" Process**



# The Iteration Life Cycle: A Mini-Waterfall



# What happened in inception

- A short requirements workshop
- Most actors, goals, and use cases named
- 10-20% of use cases written in fully dressed format
- Most influential and risky quality requirements identified
- Version 1.0 of the Vision and Supplementary Specifications written
- Risk list (demo for Comdex show)
- Proof-of-concept prototypes
- UI prototypes
- Recommendations on buy / build / reuse
- High-level architecture and proposed components
- Plan for first elaboration iteration
- Candidate tools list

# Elaboration phase

- Build the core architecture, resolve high-risk elements, define most requirements, and estimate overall schedule and resources.

# Elaboration (continued …)

Initial series of iterations during which:

- Carry out serious investigation
- Implement core architecture
- The majority of requirements are discovered and stabilized
- Major risks are mitigated or retired
- Core architectural elements are implemented and proven

# Elaboration (continued …)

- Elaboration usually 2-4 iterations.
  - Each iteration typically between 2-6 weeks depending on team size.
  - Each iteration is time-boxed (i.e. end date is fixed)
    - If team is unable to meet date, requirement is moved to next iteration
- During this phase, prototypes are not throw-away

# Elaboration - best practices

- Do short time-boxed risk-driven iterations
- Start programming early
- Adaptively design, implement, and test the core and risky parts of the architecture
- Test early, often, and realistically
- Adapt based on feedback from tests, users, and developers
- Write most of the use cases and other requirements in detail, through a series of workshops, once per elaboration iteration

# Planning the next iteration

Organize requirements and iterations by:

- Risk: technical complexity, uncertainty of effort, …
- Coverage: ensure all major parts of the system are at least touched on in early iterations
- Criticality: functions of high business value

# Artifacts that may start in Elaboration

| Artifact | Comment |
|---|---|
| Domain Model | visualization of domain concepts; similar to static information model of domain entities |
| Design Model | Set of diagrams that describe logical design (class diagrams, object interaction diagrams, package diagrams, etc…) |
| Software architecture document | key architectural issues and their resolution in the design |
| Data Model | database schemas |
| Test Model | what will be tested and how |
| Implementation Model | source code, executables, database, etc… |
| Use-case storyboards<br>UI Prototypes | UI, paths of navigation, usability models, etc … |