CS3500: Object-Oriented Design
Spring 2014

Class 19
3.21.2014

Assignment 9

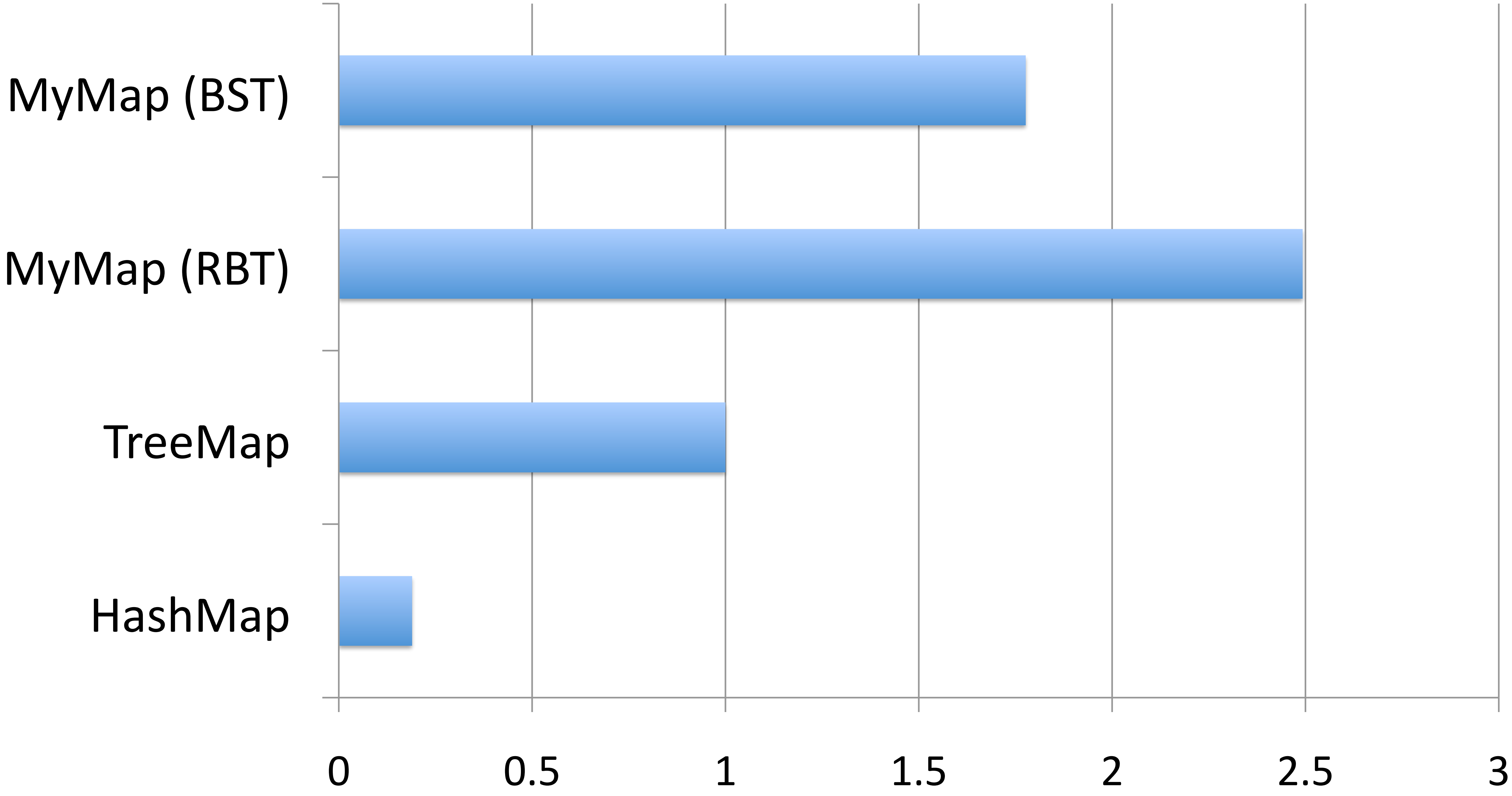
Two parts:

- Part 1 - Benchmarking: paper copy due NOW
- Part 2 - Timing Testing: due TONIGHT at 11:59pm via Web-CAT

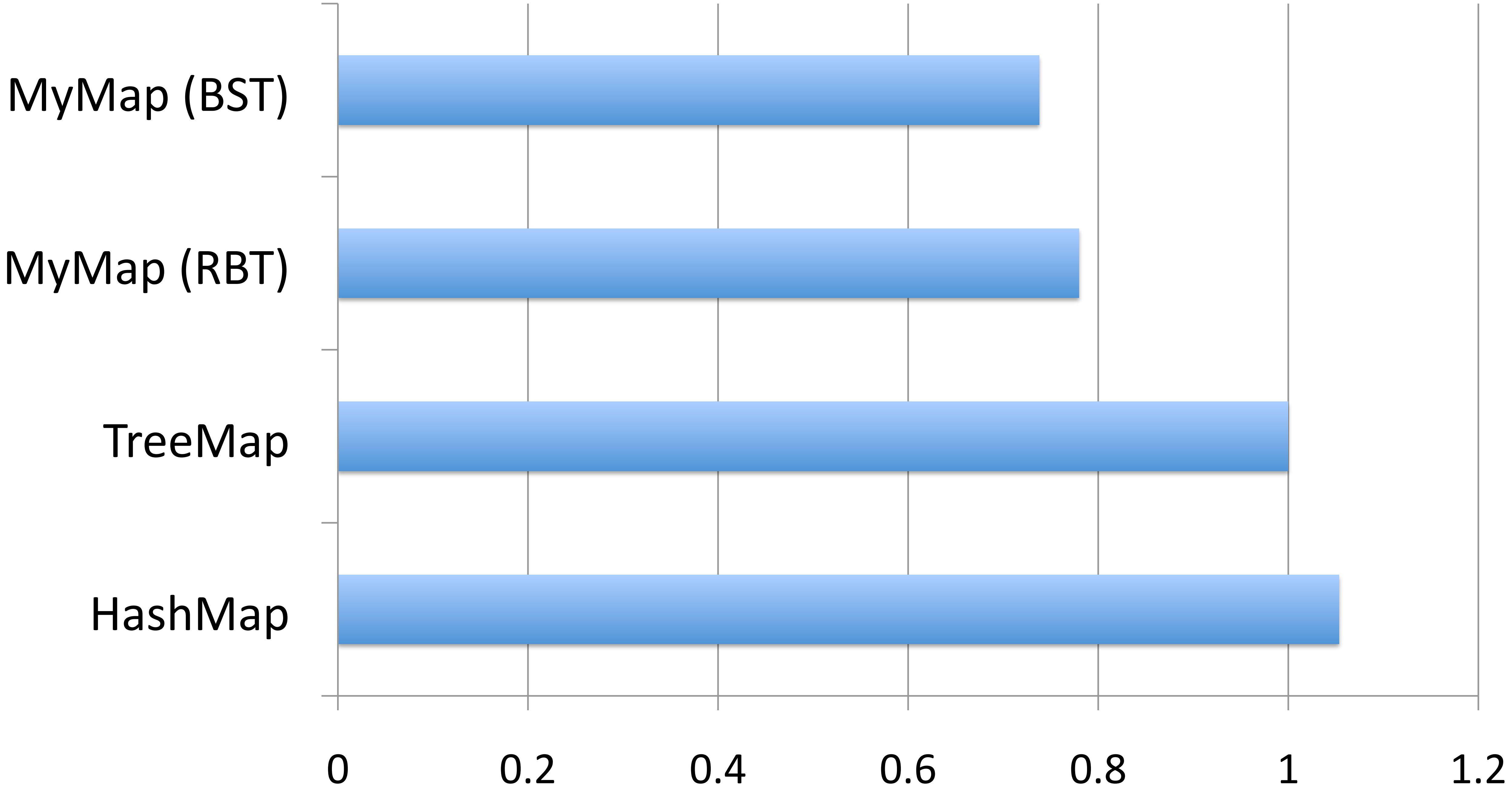
Relative Performance

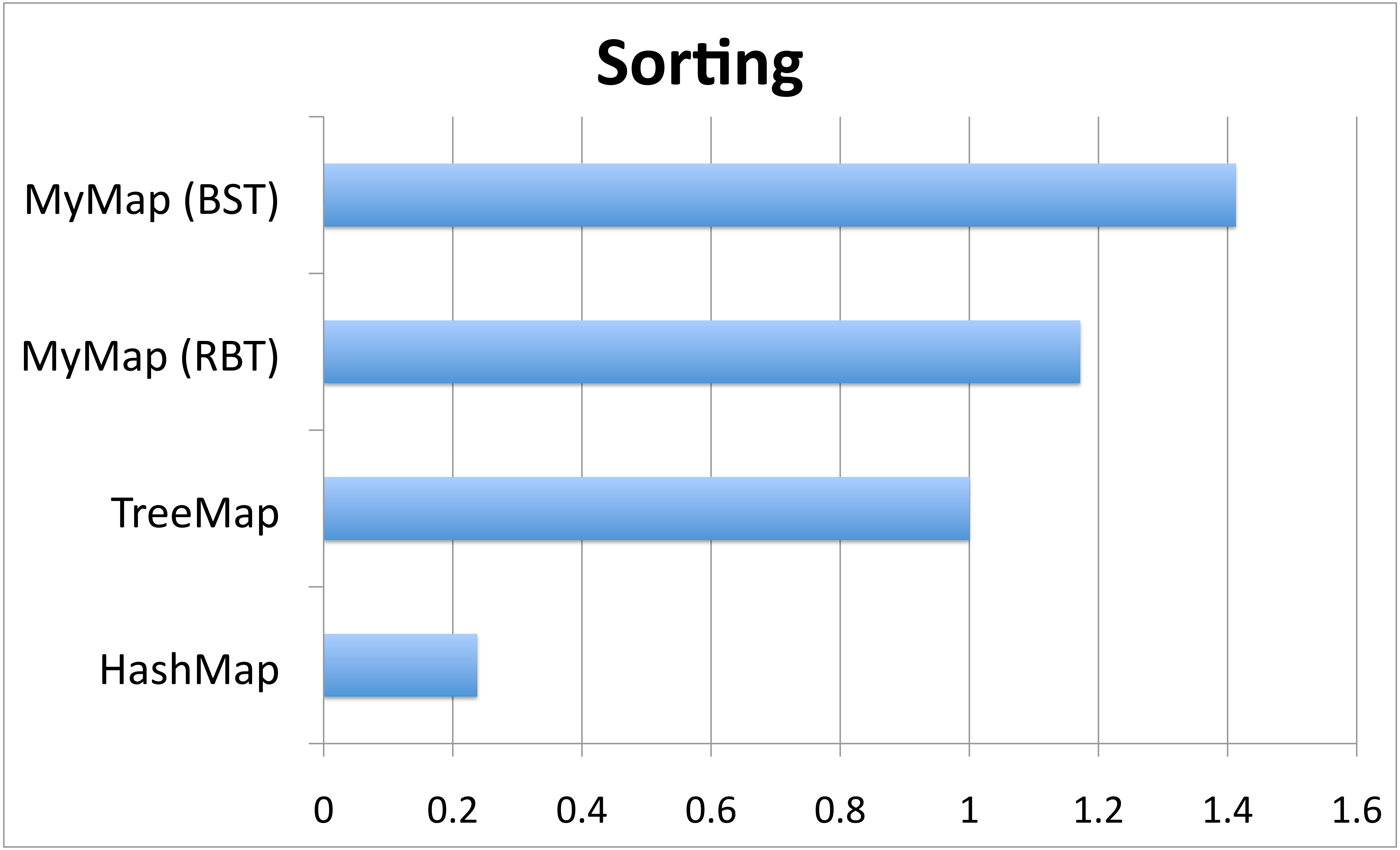
on iliad.txt benchmark

Creating



Extracting




Sorting
MyMap (BST)
MyMap (RBT)
TreeMap
HashMap
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6

Assignment 10

- Requirements and Design Documents
- Group assignment - groups of 3 or 4
- Due Monday, March 31, 2014 at 11:59pm

Software Process

Phases of the Software Process

- Requirements
- Design
- Implementation
- Testing
- Maintenance

Software Process Models

Waterfall Model

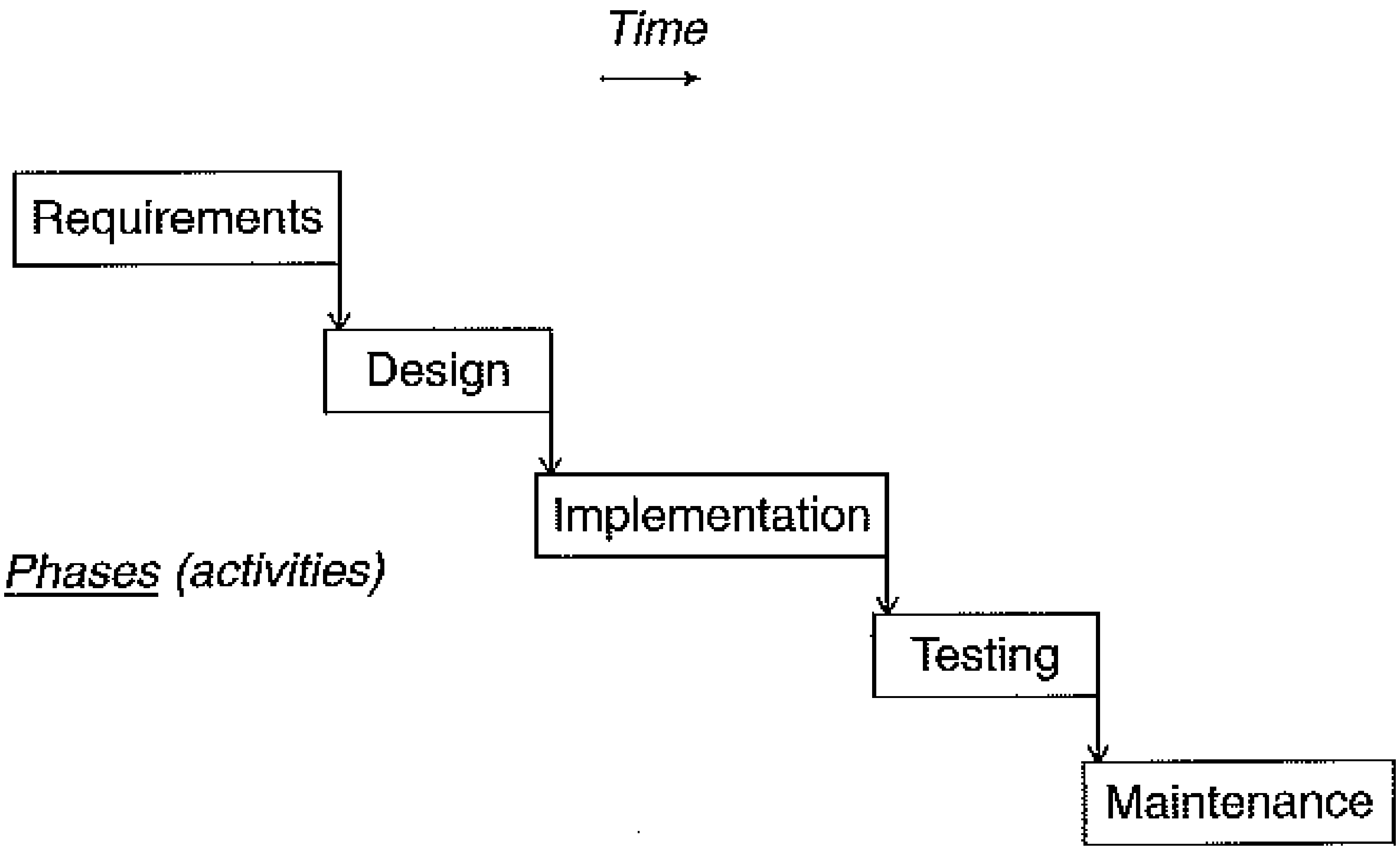


Figure from [Braude & Bernstein]

Waterfall with Feedback

- fi

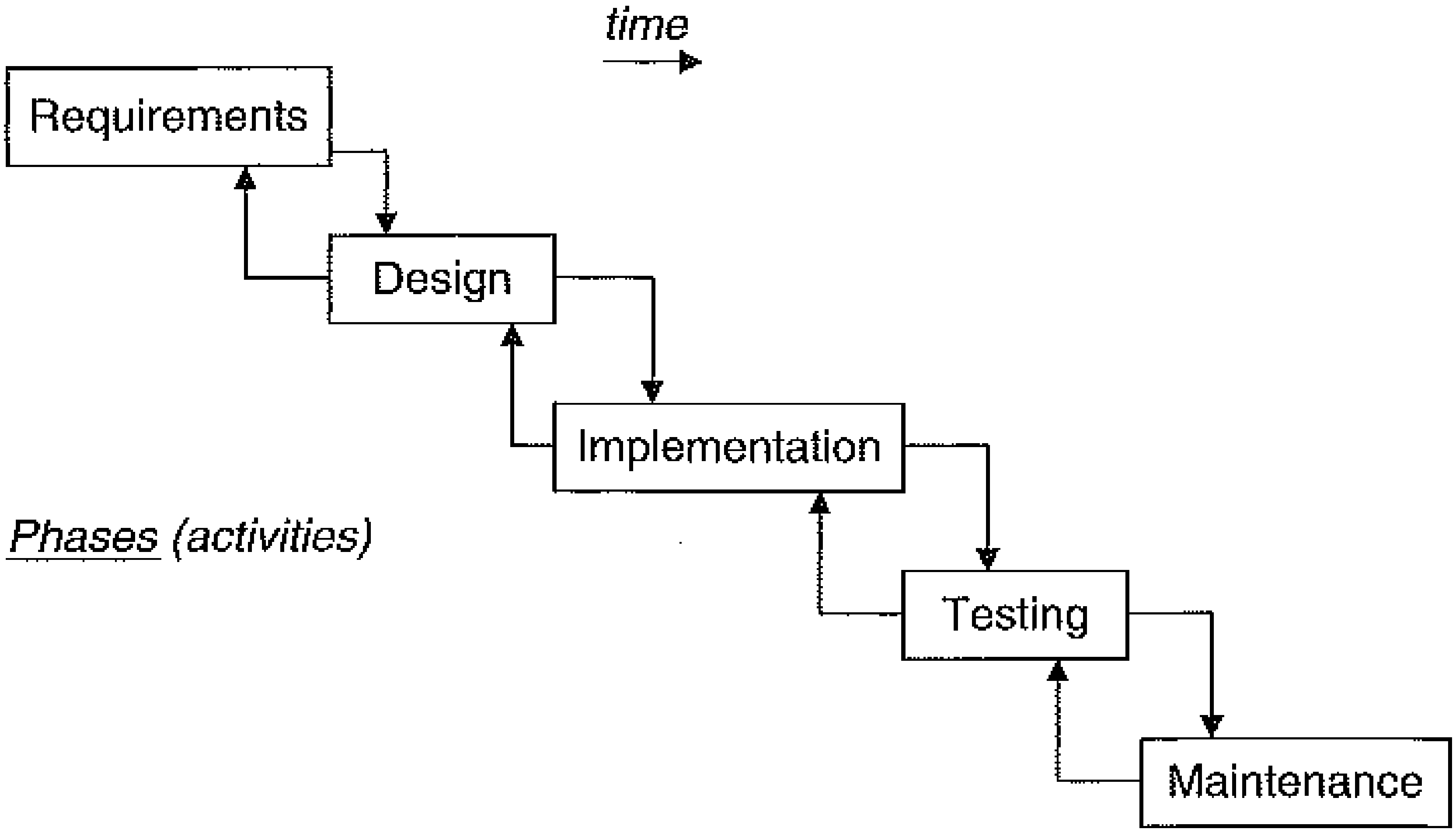


Figure from [Braude & Bernstein]

Waterfall

Pros

- Simple & easy to use
- Practiced for many years
- Easy to manage
- Facilitates allocation of resources
- Works well for smaller projects where requirements are very well understood

Cons

- Requirements must be known up front
- Hard to estimate reliably
- No stakeholder feedback until after testing
- Major problems not discovered until late in process
- Lack of parallelism
- Inefficient use of resources

Iterative & Incremental

Iterative

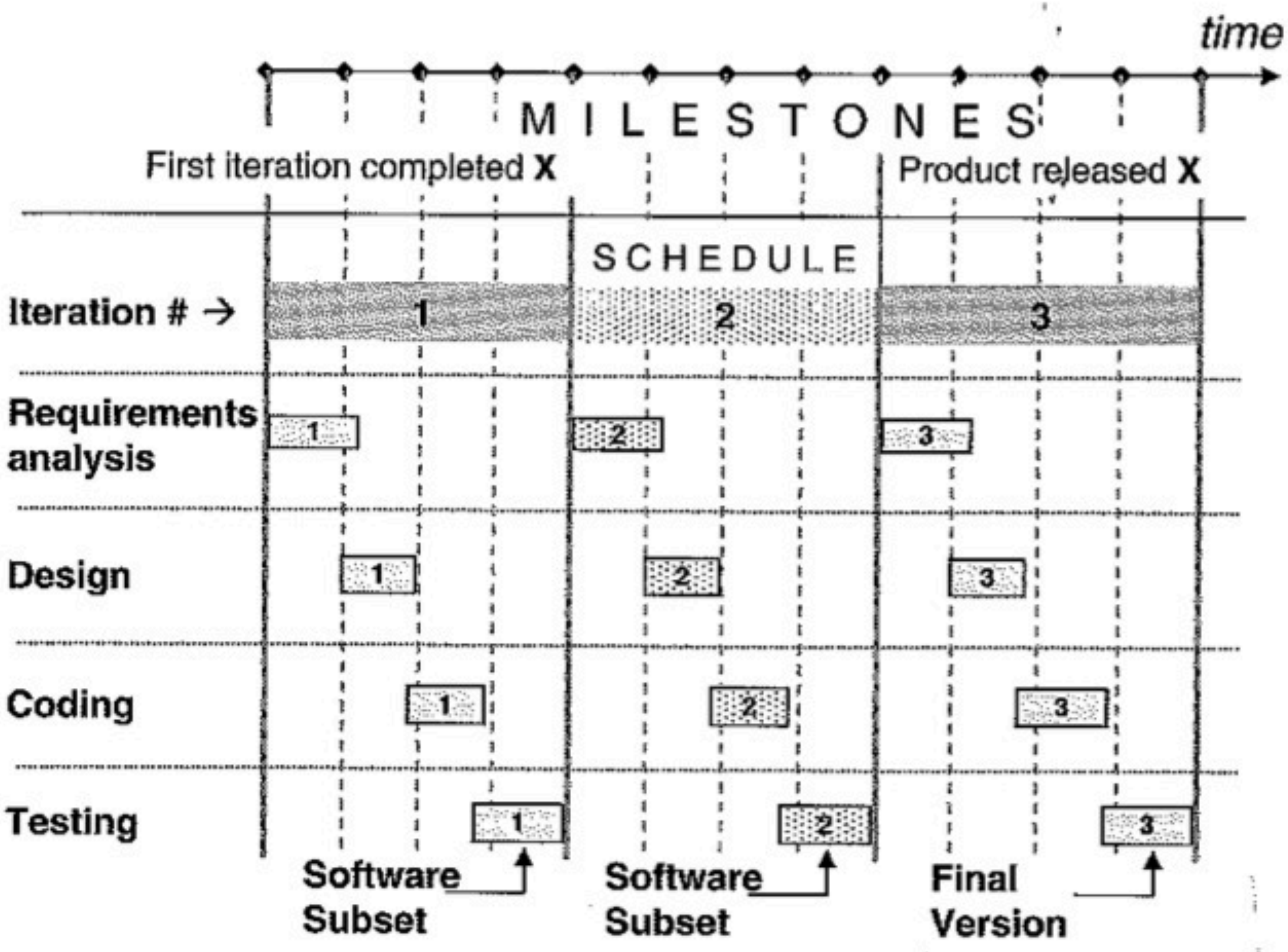


Figure from [Braude & Bernstein]

Spiral

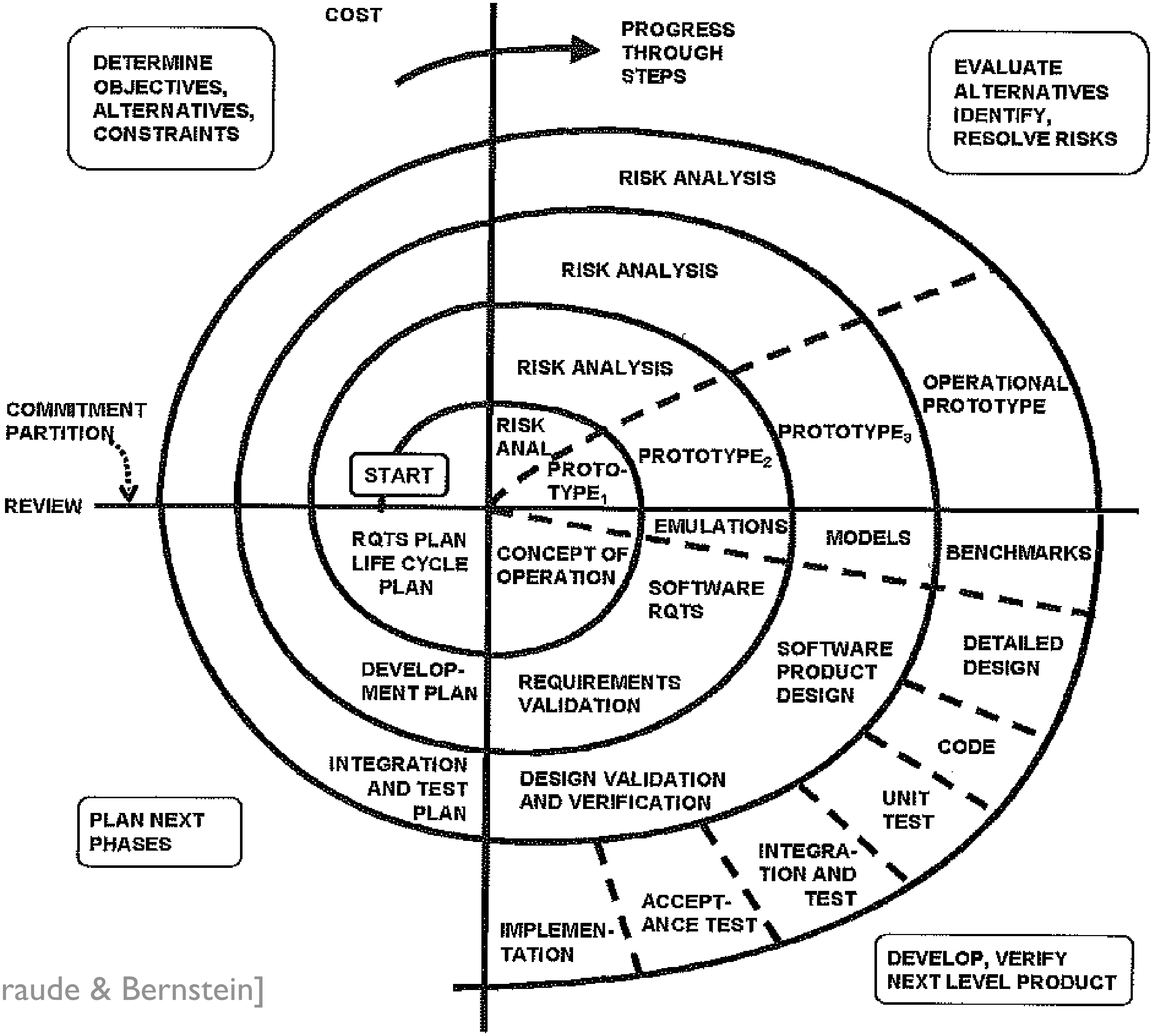


Figure from [Braude & Bernstein]

Incremental

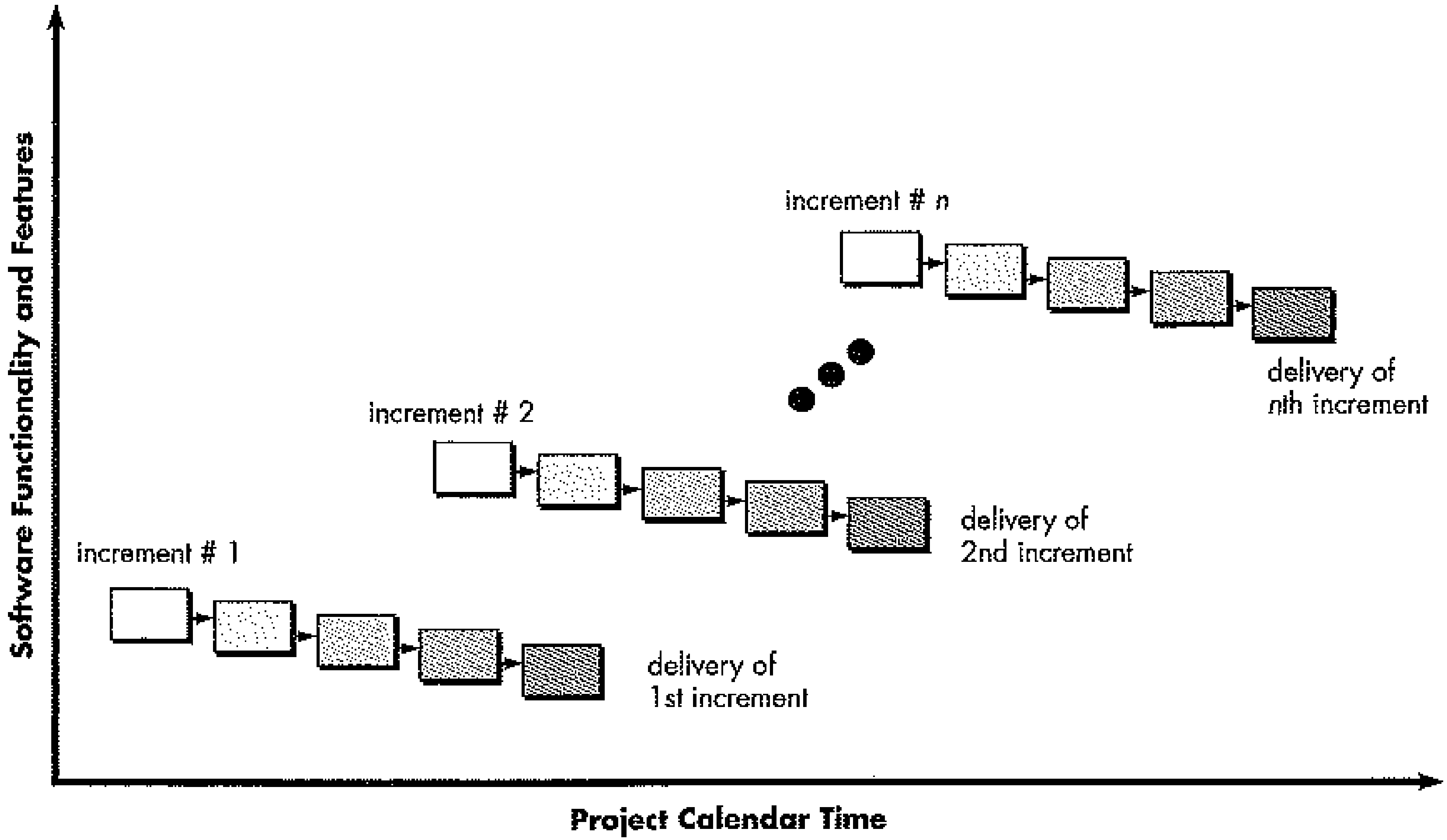


Figure from [Pressman]

Requirements

Why should we care about requirements?

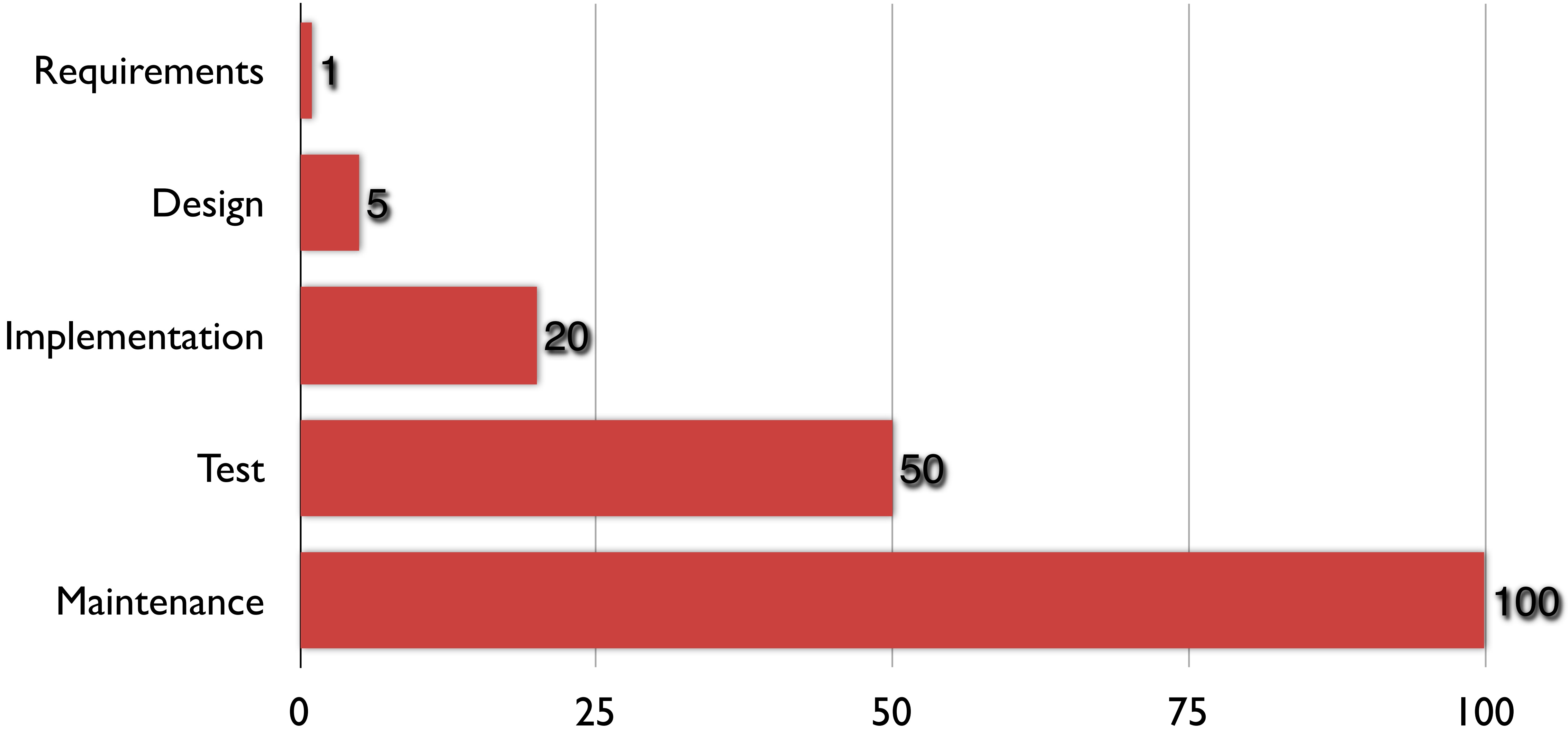
“There’s no sense being exact about something if you don’t even know what you’re talking about.”

- John von Neumann

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is so difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."

- Fred Brooks

Cost of Defects



Reasons for Project Failure

• Incomplete requirements	13.1%
• Lack of user involvement	12.4%
Lack of resources	10.6%
• Unrealistic expectations	9.9%
Lack of executive support	9.3%
• Changing requirements/specification	8.7%
Lack of planning	8.1%
• Didn’t need it any longer	7.5%

Sources: Standish Group, 1995 and 1996; *Scientific American*, September 1994.

Project Success Factors

• User involvement	15.9%
Management support	13.9%
• Clear statement of requirements	13.0%
Proper planning	9.6%
• Realistic expectations	8.2%
Smaller milestones	7.7%
Competent staff	7.2%
• Ownership	5.3%

Sources: Standish Group, 1995 and 1996; *Scientific American*, September 1994.

Sources of requirements

- Stakeholders
- Documents
- Books

Types of Requirements

- Functional
  - expresses a function that an application must perform
- Nonfunctional
  - does not involve specific functionality
  - qualifies services or functionalities
  - should be specific, quantifiable, and testable

Examples

- Functional
  - The application allows clerks to check out DVDs.
- Nonfunctional
  - The application shall display a customer’s account status in less than two seconds.

Analysis & Specification

- Analysis: process of understanding the problem and the requirements for a solution
- Specification: process of describing what a system will do
- **Analysis leads to Specification** – they are not the same

[From NCSU CSC510 Slides]

Output

- Software Requirements Specification (SRS)
- All stakeholders can agree on set of requirements

User of a Requirements Document

[Kotonya & Sommerville]

- **System customers**
  - Specify the requirements & read them to check that they meet their needs.
  - Specify changes to the requirements.
- **Managers**
  - Use the requirements document to plan a bid for the system & to plan the system development process
- **System engineers**
  - Use the requirements to understand what system is to be developed.
- **System test engineers**
  - Use the requirements to develop validation tests for the system.
- **System maintenance engineers**
  - Use the requirements to understand the system & the relationships between its parts

Requirements Document

- The official statement of what is required of the system developers should include both a definition and a specification of requirements
- It is NOT a design document
- As far as possible, it should set of WHAT the system should do rather than HOW it should do it

[From NCSU CSC510 Slides]

Format of Requirements

[Hull, Jackson, and Dick]

Functional Requirements

- The <stakeholder type> shall be able to <capability/function>.
- The system shall allow the <stakeholder type> to <capability/function>.
- The system shall <capability/function>.

Constraints

[Hull, Jackson, and Dick]

- **Performance/capability:** The <system> shall be able to <function> <object> not less than <performance> times per <units>.
- **Performance/capability:** The <system> shall be able to <function> <object> of type <qualification> within <performance> <units>
- **Performance/capacity:** The <system> shall be able to <function> not less than <quantity> <object>.
- **Performance/timeliness:** The <system> shall be able to <function> <object> within <performance> <units> from <event>.
- **Performance/periodicity:** The <system> shall be able to <function> not less than <quantity> <object> within <performance> <units>
- **Interoperability/capacity:** The <system> shall be able to <function> <object> composed of not less than <performance> <units> with <external entity>.
- **Sustainability/periodicity:** The <system> shall be able to <function> <object> for <performance> <units> every <performance> <units>.
- **Environmental/operability:** The <system> shall be able to <function> <object> while <operational condition>.

Quality of Requirements
[Braude & Bernstein]

- Comprehensiveness
  - How **comprehensive** is the SRS?
  - Does the SRS include all of the customer's wants and needs?
- Consistency
  - How **consistent** is the SRS?
  - Are there contradictions in the SRS?
- Self-Completeness
  - How **self-complete** is the SRS?
  - Does the SRS contain all of the necessary parts?
- Understandability
  - How **understandable** is each requirement?
- Unambiguity
  - How **unambiguous** is each requirement?
  - Can the requirement be interpreted in only one way?
- Prioritization
  - How **effectively prioritized** are the requirements?
  - Is there an order for implementation?
- Testability
  - How **testable** is each requirement?
  - Is it possible to validate the requirement in the finished product?
- Traceability
  - How **traceable** is each requirement?
  - Is the requirement traceable forward? Backward?

Traceability

- Track each requirement to design and code that implements it
- Track each part of code back to corresponding elements of design and requirements
- Critical in requirements engineering to link the requirements for the system and other important entities of the software

Traceability

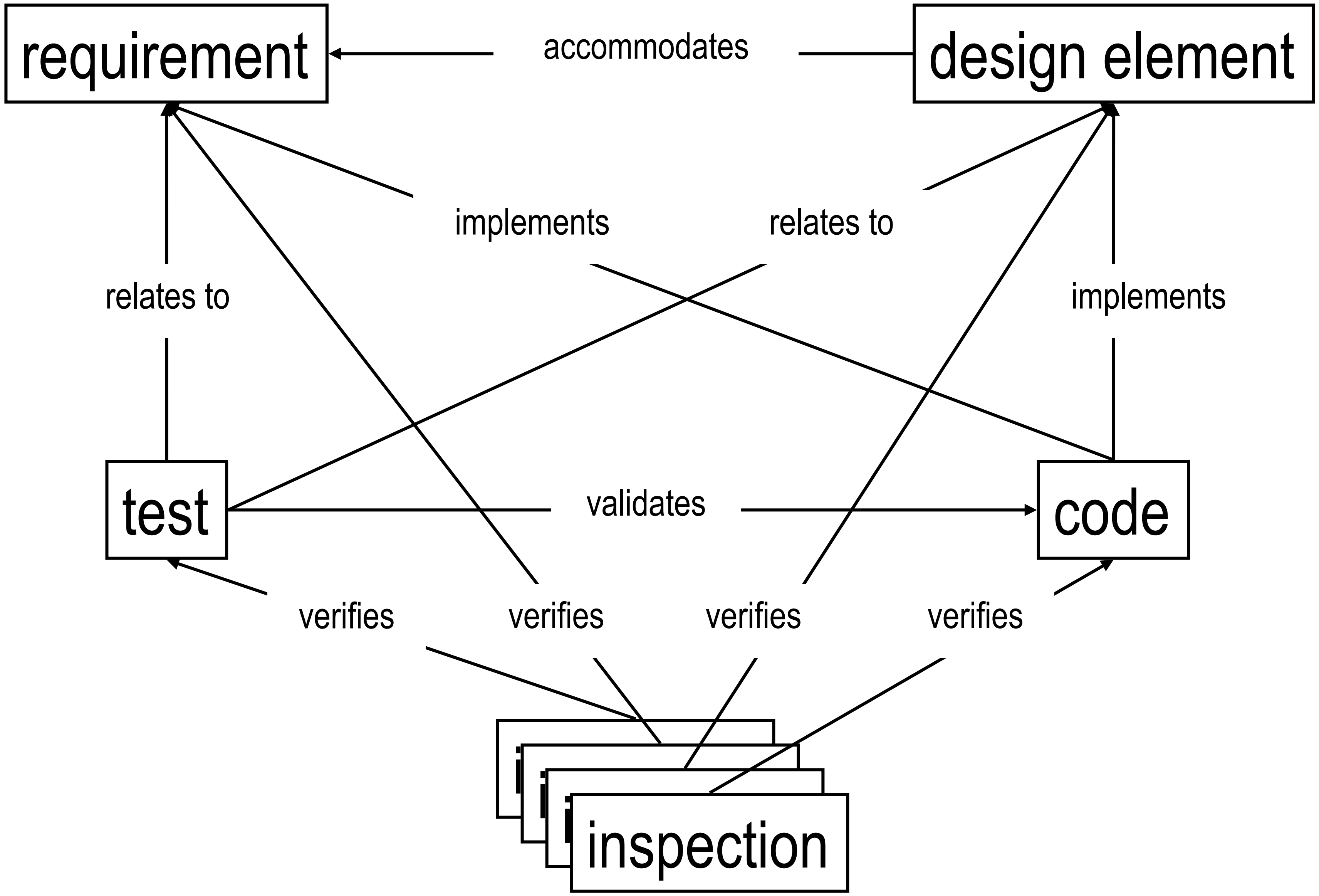


Figure from [Braude & Bernstein]

Benefits of Traceability

[Hull, Jackson, and Dick]

- Greater confidence in meeting objectives.
- Ability to assess the impact of change.
- Improved accountability of subordinate organizations.
- Ability to track progress.
- Ability to balance cost against benefit.

Stories

User Stories

[Cohn]

- Describe software functionality that will be important for the users
- Allow software engineers to gain an understanding of what the software may include

Aspects in User Stories [Cohn]

- A written description of the story used for planning and as a reminder
- Conversations about the story that serve to flesh out the details of the story
- Tests that convey and document details and that can be used to determine when a story is complete

Use Cases

- Use case: sequence of actions taken by the actor and the application
- Actor (person or system)
- Scenario: list of numbered steps
  - Main success scenario
  - Extensions or error conditions

Use Case: Retrieve a File

1. User clicks *File* menu
2. System displays options *New* and *Open*
3. User clicks *Open*
4. System displays file window
5. User enters directory and file name
6. User hits *Open* button
7. System retrieves referenced file into word processor window Extensions

Use Case: Retrieve a File

Main Success Scenario

1. User clicks *File* menu
2. System displays options *New* and *Open*
3. User clicks *Open*
4. System displays file window
5. User enters directory and file name
6. User hits *Open* button
7. System retrieves referenced file into word processor window Extensions

Extension

7aSystem displays error indicating file could not be opened

Use Case Format

- Name: describes the task
- Associated requirement
- Actor: who is the user
- Priority
- Main success scenario
- Extensions

Use Case Exercise

Rent a Movie

1. User presses “Rent a Movie”
2. System displays movies that are for rent
3. User selects a movie
4. System displays selected movie information
5. User presses “Rent”
6. System prompts for credit card payment
7. User swipes credit card
8. System ejects movie

Rent a Movie

1. User presses "Rent a Movie"
   a. The user shall be able to press "Rent a Movie" from the *Main* screen when she wants to rent a movie.
2. System displays movies that are for rent
   a. The system shall display the list of possible movies to rent when the user presses "Rent a Movie" from the *Main* screen.
3. User selects a movie
   a. From the *Movie Selection screen,* the user shall be able to select a movie to rent.
   b. The system shall only display available movies on the *Movie Selection* screen.
4. System displays selected movie information
   a. The system shall display movie information on the selected movie when the user selects the movie from the *Movie Selection* screen.

Rent a Movie

5. User presses "Rent"
    a. The user shall be able to press "Rent" from the *Movie* screen when she wants to rent the selected movie.
6. System prompts for credit card payment
    a. The system shall prompt the user for credit card payment via the *Payment* screen when the user indicates she wants to rent the selected movie.
7. User swipes credit card
    a. The user shall be able to swipe her credit card to pay for the selected movie.
8. System ejects movie
    a. The system shall process the credit card payment when the user has selected a movie and swiped her credit card.
    b. The system shall eject the movie once the payment has been processed.

Stories for Online Bookstore

- Books can be searched by author, title, or ISBN.
- Books can be viewed detailed information by number of pages, publication date, and a brief description.
- Book can be placed in "shopping cart" and bought.
- Books can be rated and reviewed.

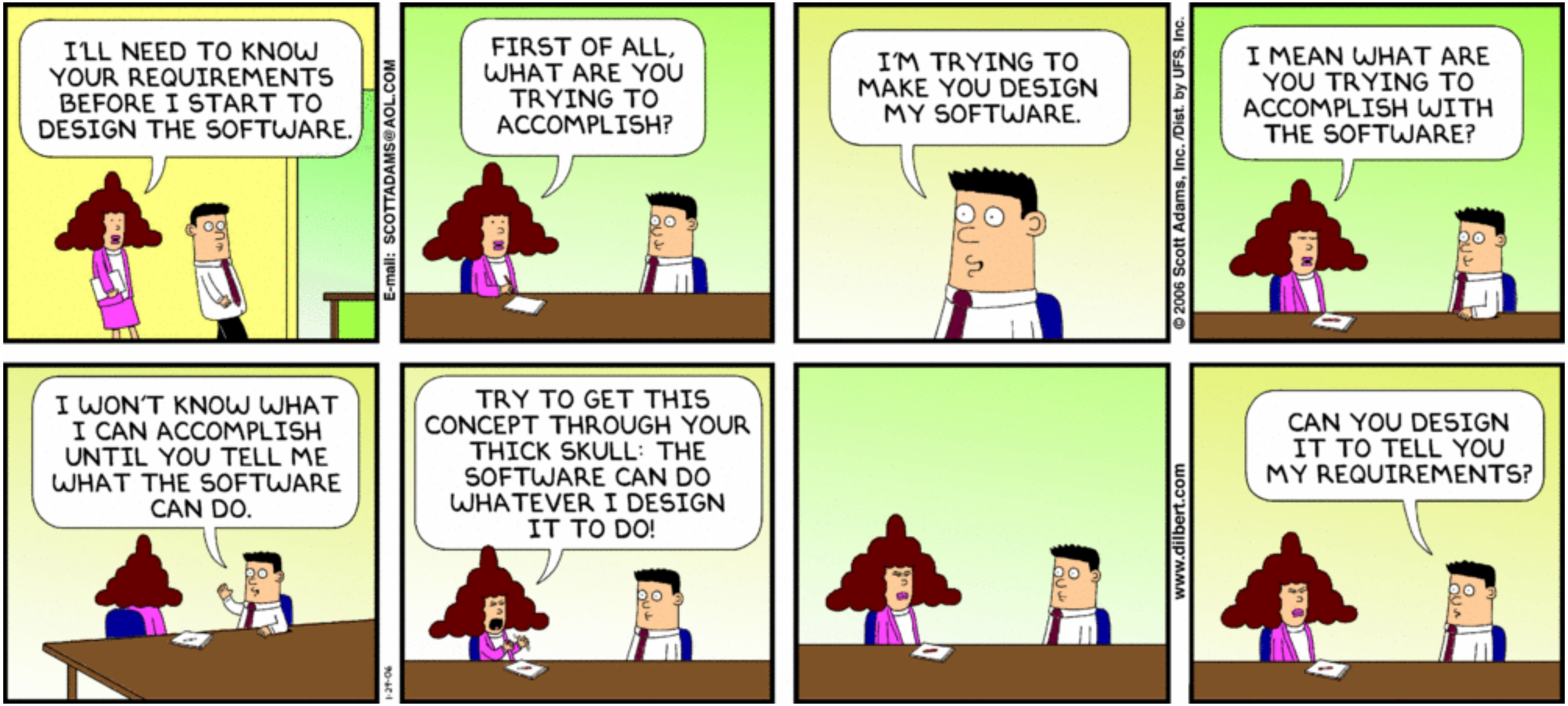
Requirements Engineering is difficult

Despite the clear benefits of getting requirements complete, right, and error-free early, they are the hardest part of the system development lifecycle to do right because:

- we don't always understand everything about the real world that we need to know,
- we may understand a lot, but we cannot express everything that we know
- we may think we understand a lot, but our understanding may be wrong,
- requirements change as client's needs change,
- requirements change as clients and users think of new things they want, and
- requirements of a system change as a direct result of deploying the system.

[From NCSU CSC510 Slides]

YOUR REQUIREMENTS DOCUMENT IS THE BIGGEST I'VE EVER SEEN.
scottadams@aol.com
www.dilbert.com
IT'S TOO BIG TO READ, BUT I CAN GUESS FROM ITS WEIGHT WHAT MUST BE IN THERE.
8/9/99 © 1999 United Feature Syndicate, Inc.
YOU KNOW IT'S A MULTI-USER, GLOBAL SYSTEM, RIGHT?
NO, I'M NOT GETTING THAT.


I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?
E-mail: SCOTTADAMS@AOL.COM
1-29-06
© 2006 Scott Adams, Inc./Dist. by UFS, Inc.
www.dilbert.com