Sommerville Chapter 4

Requirements Basics The Document and The Requirements

Announcements

- You should have formed your Team of 2 for the coursework and informed the TA by email
- Correction - No C++ knowledge required for this course
- Homework1 is up on the course webpage -
  - http://www.inf.ed.ac.uk/teaching/courses/inf2c-se/

Today's Goals (and next time)

- Understand the requirements problem
  - Why are requirements so important
- Get a feel for the structure of a requirements document
  - What goes in there?
- Learn how to write "good" requirements
  - Clear and testable
  - Exit criteria

We Need a Software Process

- Structured set of activities required to develop a software system
  - Specification
  - Design
  - Validation
  - Evolution
- Activities vary depending on the organization and the type of system being developed
- Must be explicitly modeled if it is to be managed

Process of Building a House

Same Life Cycle

Different Process

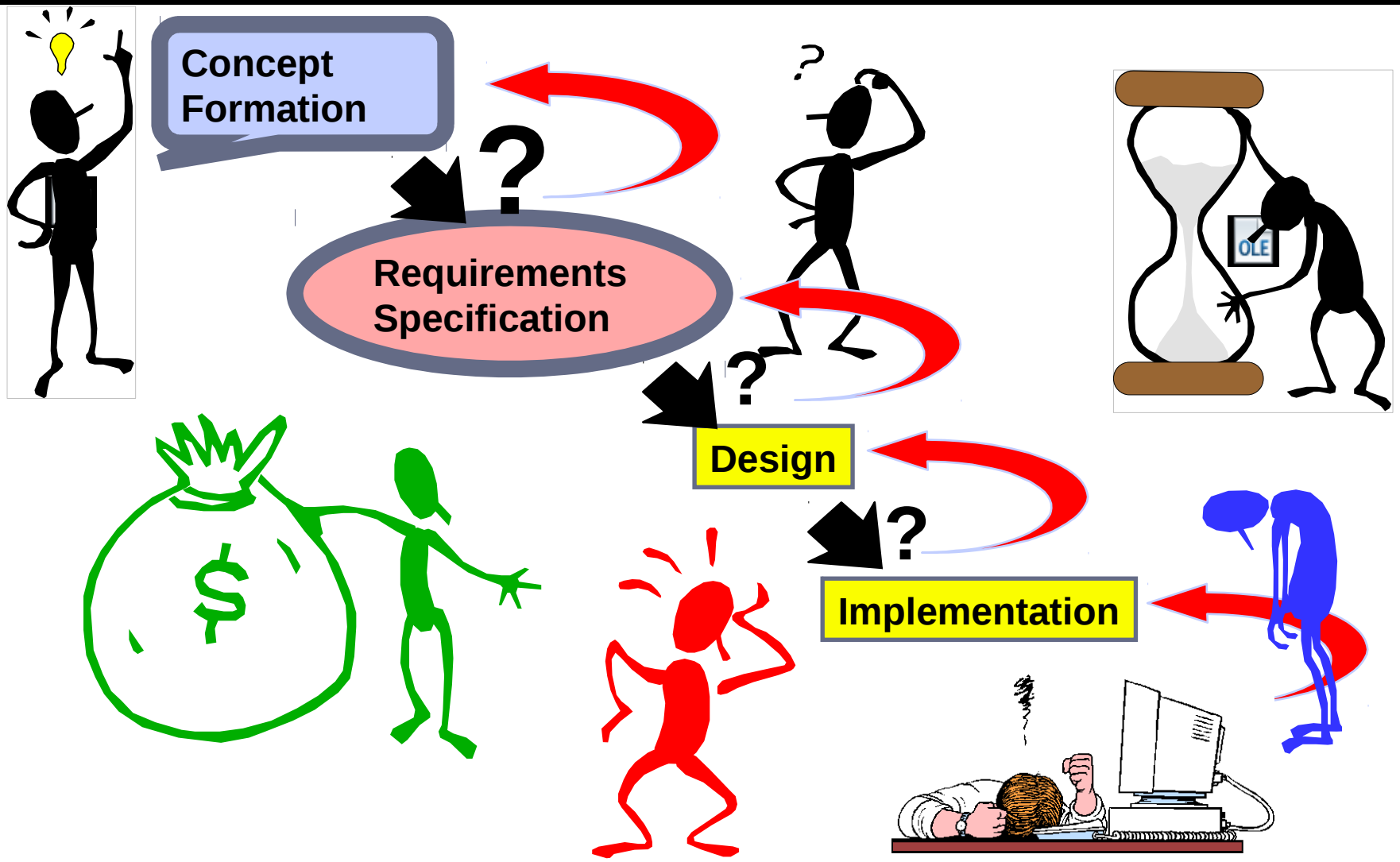
Life Cycle and Process

- Life cycle
  - Phases necessary to keep the product in existence
- Software process
  - Define human activities required to build software
  - Who is doing what, when, and how
- Procedure
  - A sequential series of steps to be followed by a single individual to accomplish a task or make a decision
- Process
  - A flow of events that describes how something works

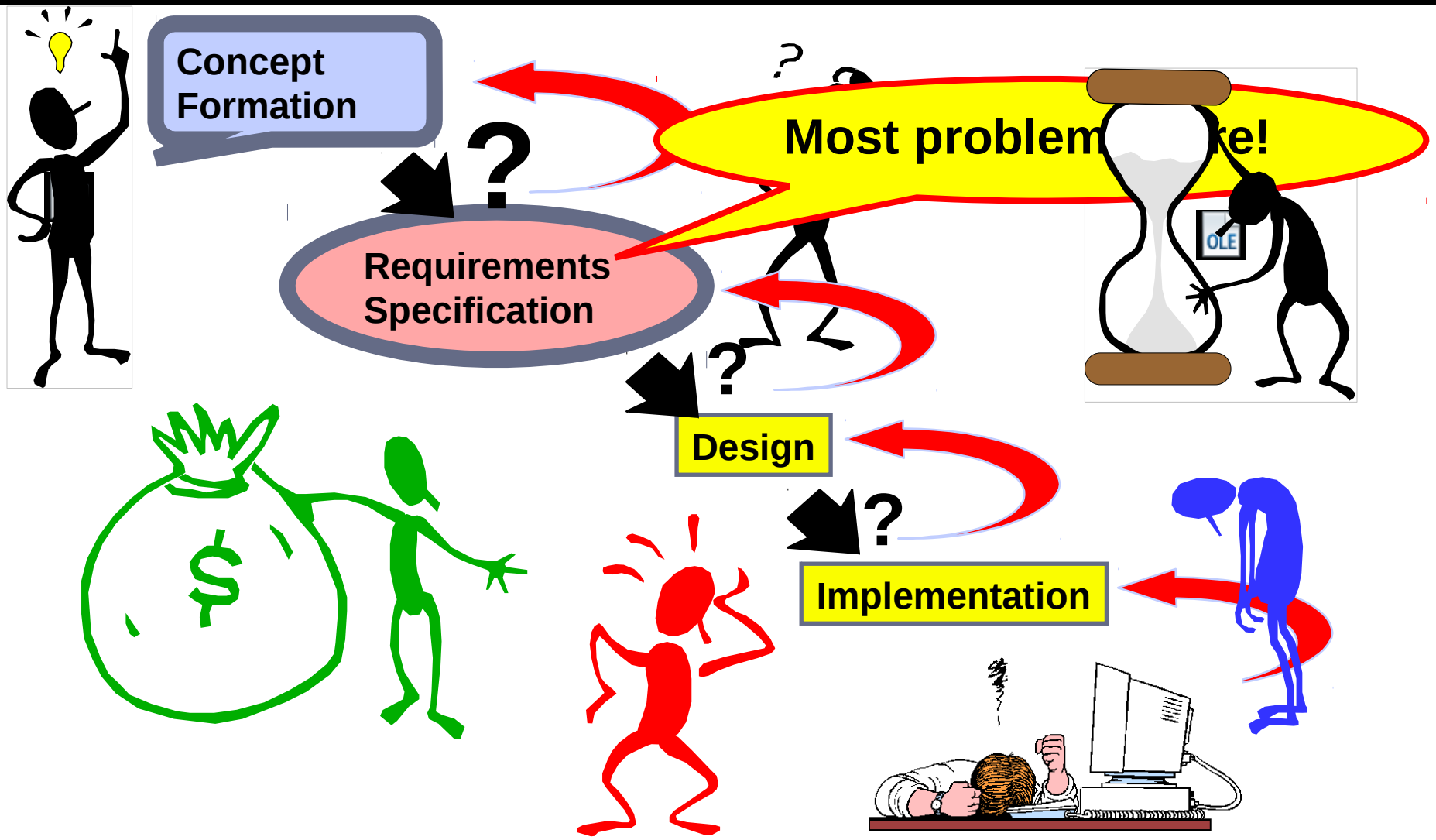
Generic Software Process Models

- The Waterfall Model
  - Separate and distinct phases of specification and development
- Evolutionary Development
  - Specification and development are interleaved
- Spiral Model
  - Let risk analysis drive your process
- Incremental Development
  - Deliver your system in small planned increments
- Agile and eXtreme Programming

The Importance of Good Requirements

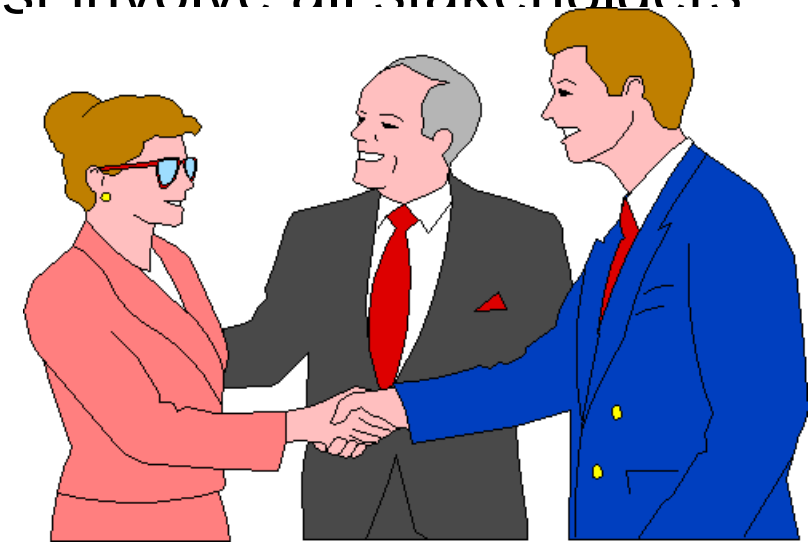


The Importance of Good Requirements



Requirements Specification

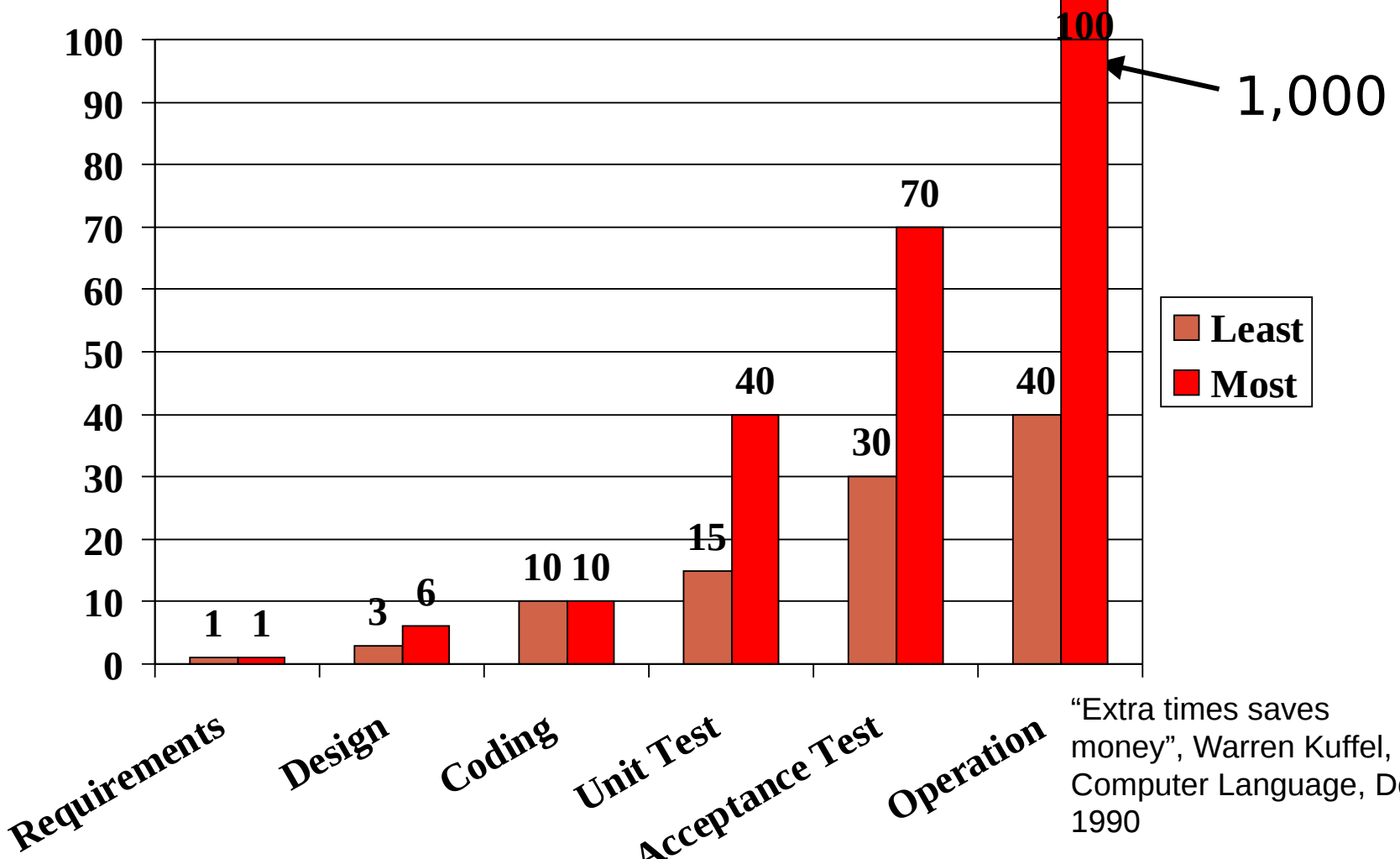
- High-level description of what a system should do
- Must be detailed enough to distinguish between the "right" and the "wrong" system
- Capture the **what** not the **how**
- The specification process must involve all stakeholders
  - Customers
  - Engineers
  - Regulatory agencies
  - Users



Importance of Requirements

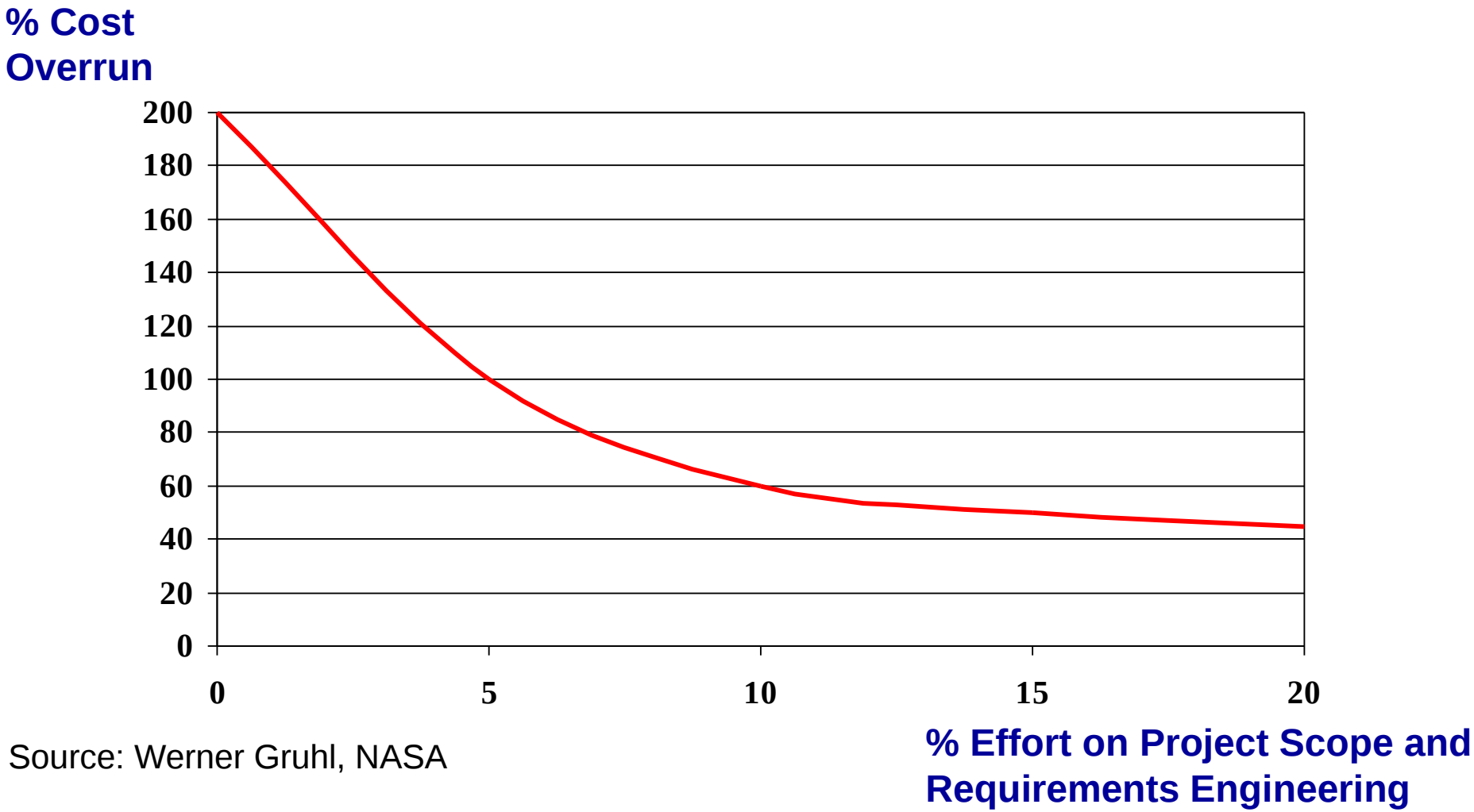
- The Engineering Argument
  - Engineering is about developing solutions to problems
  - A good solution can only be developed if the engineers have a solid understanding of the problem
- The Economic Argument
  - Errors cost more to correct the longer they go undetected
  - Cost of correcting requirements errors is (at least) 100 times more in the maintenance phase than in the requirements phase
- The Empirical Argument
  - Failure to understand and manage requirements is the biggest single cause of cost and schedule over-runs

The Cost of Req. Problems



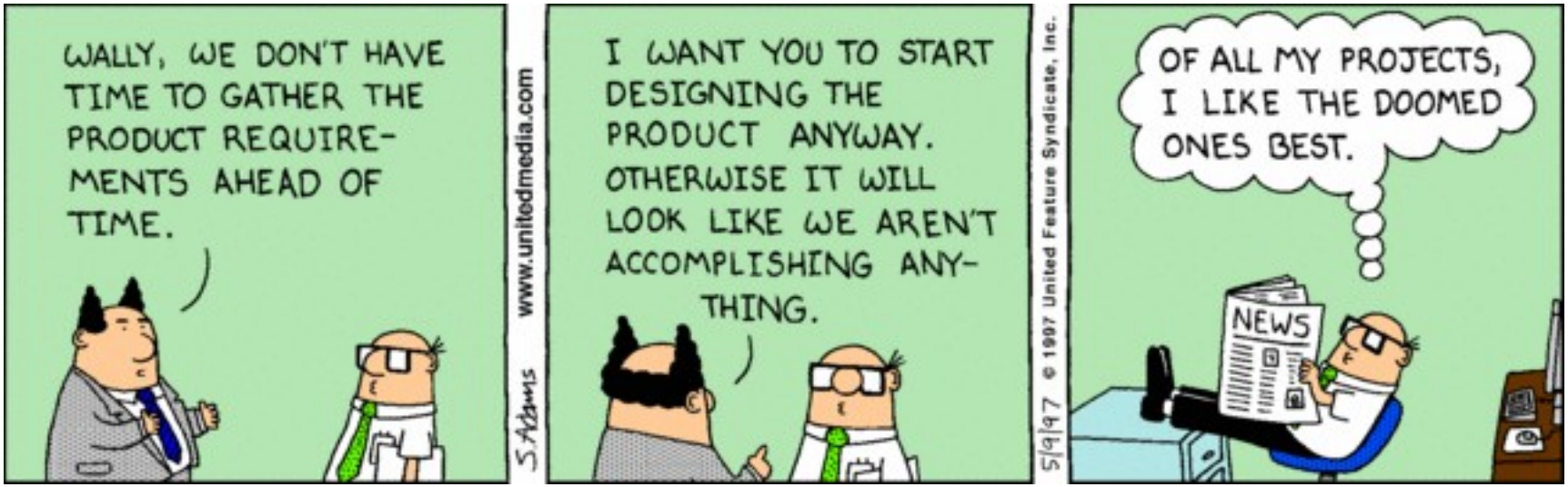
"Extra times saves money", Warren Kuffel, Computer Language, Dec 1990

Cost Overruns vs. Requirements Effort



Source: Werner Gruhl, NASA

Even Wally Knows



Key Points

- Requirements capture what a proposed system shall do
  - But avoids design detail as much as possible
  - Written in the user's language

- **Poor requirements are the source of all evil**

- Requirements problems are the
  - Most costly
  - Most difficult to correct (they are conceptual)

Requirements: What Are They?

The Requirement

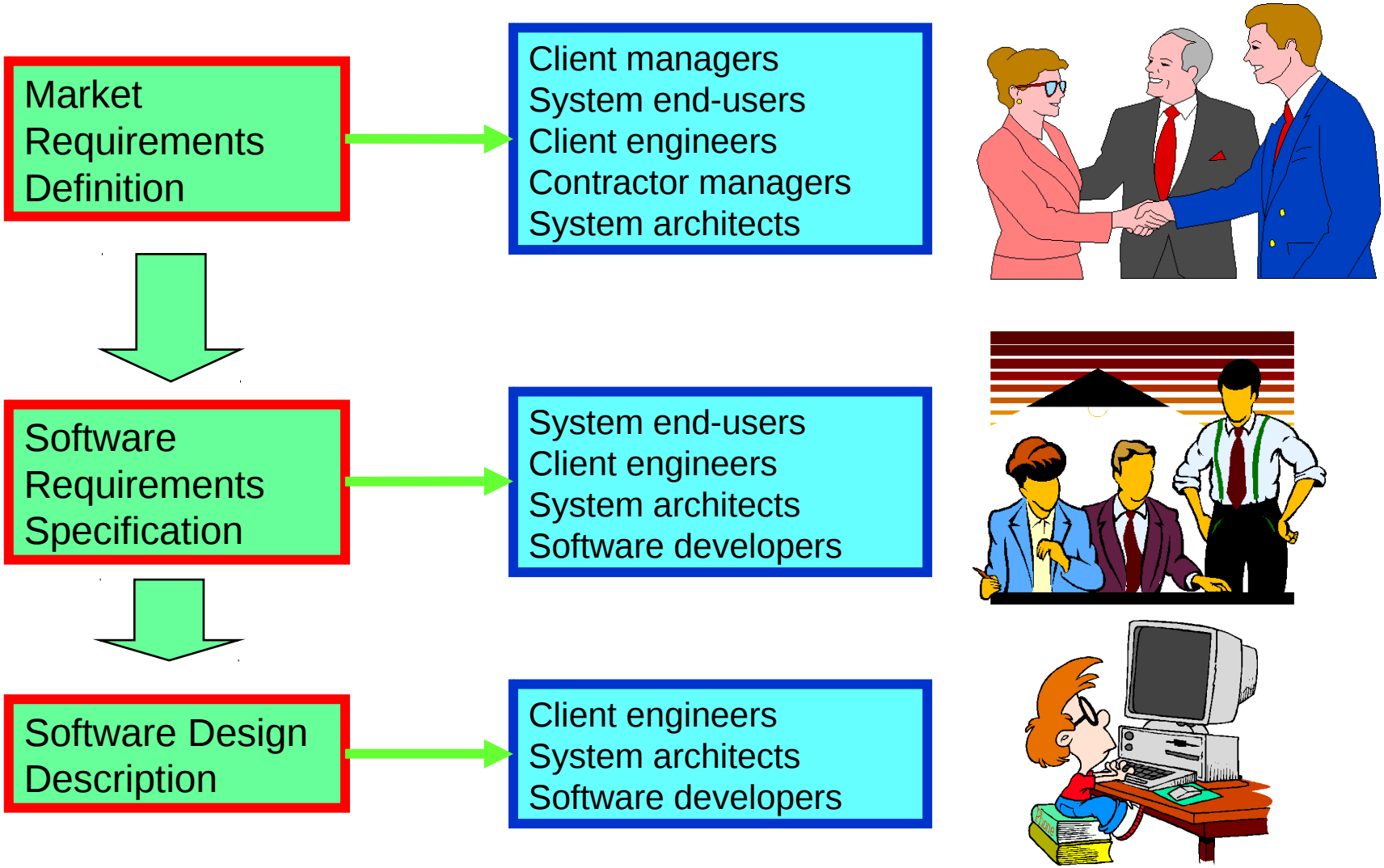
- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification

- This is inevitable as requirements may serve a dual function
  - May be the basis for a bid for a contract
  - Therefore must be open to interpretation
  - May be the basis for the contract itself
  - Therefore must be defined in detail
- Both these statements may be called requirements

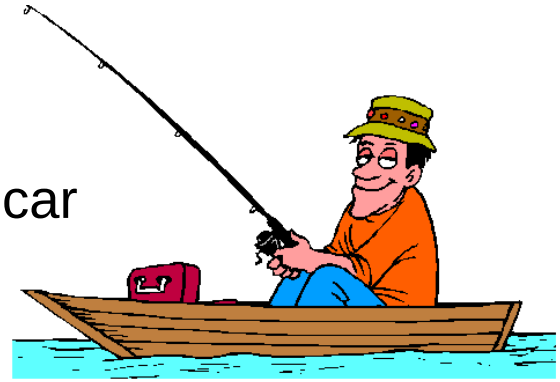
Requirements Definition/Specification

- Market Requirements Definition (MRD)
  - Statements in natural language (plus diagrams) of the services the system provides and its operational constraints
  - Written for customers in their language
- Software Requirements Specification (SRS)
  - A structured document setting out detailed descriptions of the system services
  - Written as a contract between client and contractor
- Software Design Description (high-level design)
  - A detailed software description which can serve as a basis for a design or implementation
  - Written for developers

Requirements Readers



Definitions and Specifications



Market Requirements Definition

1. One person must be able to load the boat on the car rack

(Software) Requirements Specification

1.1 The boat must be lighter than 100 lb.

1.2 The boat must have handles to help one person lift it

1.3 The car rack must be padded so the boat can easily slide into the rack

1.4 Etc.

Capturing Good Requirements

3 Common Problems

- Poorly structured requirements document
- Poorly written individual requirements
- Untestable requirements (future lecture)

The Software Requirements Specification

- The SRS writer shall address the following
  - Functionality
  - What is the software supposed to do?
  - External Interfaces
  - How does the software interact with people, the system's hardware, other hardware, and other software?
  - Performance
  - What is the speed, availability, response time, recovery time of various software functions, etc.?
  - Attributes
  - What are the portability, maintainability, security, etc. considerations?
  - Design constraints imposed on an implementation
  - Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment, etc.?

IEEE Std 830-1998

SRS Should not Include

- Project development plans (cost, staffing, schedules, methods, tools, etc.)
  - Lifetime of SRS is until the end of the operational life of the product
  - Lifetime of development plans is much shorter
- Product assurance plans (CM, V&V, test, QA, etc.)
  - Different audiences
  - Different timelines
- Design
  - Requirements and design have different audiences
  - Analysis and design are different areas of expertise (requirements experts should not do design)
  - Except where the application domain constrains the design, e.g., limited communications bandwidth or security concerns

IEEE Document Standard

- Portrait slides
  - Available on the web page

Requirements Engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed
- Requirements may be functional or non-functional
  - Functional requirements describe system services or functions
  - Non-functional requirements is a constraint on the system or on the development process

Functional Requirements

- A functional requirement is something the system must do.
- A functional requirement is testable
- A general rule is a functional requirement is a “shall statement”
  - The system shall require users to login to access all functions.

Functional Requirements

- These can be high level or low level (generally we're at high level in this class)
- High level: The system shall charge users credit cards for purchases
- Low level: The system shall validate all passwords contain upper and lowercase characters and one number

Example-Functional Requirements

- Requirements must do **ONE THING.**
  - Bad:
    - The system shall accept credit cards and accept pay pal
  - Good:
    - The system shall accept credit cards
    - The system shall accept pay pal.
- • Requirements must be **testable**. Use precise language.
  - Bad:
    - The system shall work with any browser
  - Good:
    - The system shall work with Firefox
    - The system shall work with IE
  - Bad:
    - The system shall respond quickly to user clicks
  - Good:
    - The system shall respond within 10ms to any user click

Non-Functional Requirements

- Define system properties and constraints
  - Reliability, response time and storage requirements
  - Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular CASE system, programming language, or development method

- Non-functional requirements may be more critical than functional requirements
  - If these are not met, the system is useless

Non-Functional Classification

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way
  - Execution speed, reliability, etc.
- Organizational requirements
  - Requirements which are a consequence of organizational policies and procedures
  - Process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are external to the system and its development process
  - Interoperability requirements, legislative requirements, etc.

Non-Functional Requirements Examples

- Product requirement
  - It shall be possible for all necessary communication between the ATM and the user to be expressed in the standard ASCII character set.
- Organizational requirement
  - The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95.
- External requirement
  - The system shall use a file format readable by MS-Word 6.0 and 2000.

Writing Requirements

- Natural language, supplemented by diagrams and tables is the normal way of writing requirements definitions
- This is universally understandable but three types of problem can arise
  - Lack of clarity
  - Precision is difficult without making the document difficult to read
  - Requirements amalgamation
  - Several different requirements may be expressed together
  - Requirements confusion
  - Functional and non-functional requirements tend to be mixed-up

Requirements Samples (NASA)

2.16.3.f

While acting as the bus controller, the C&C MDM CSCI shall set the e, c, w, indicators identified in Table 3.2.16-II for the corresponding RT to “failed” and set the failure status to failed for all RT’s on the bus upon detection of transaction errors of selected messages to RTs whose 1553 FDIR is not inhibited in two consecutive processing frames within 100 milliseconds of detection of the second transaction error if; a backup BC is available, the BC has been switched in the last 20 seconds, the SPD card reset capability is inhibited, or the SPD card has been reset in the last 10 major (10 second) frames, and either

1. the transaction errors are from multiple RT’s, the current channel has been reset within the last major frame, or

2. the transaction errors are from multiple RT’s, the bus channel’s reset capability is inhibited, and the current channel has not been reset within the last major frame.

Templates are Essential

- Define a standard document structure
  - Readers familiar with the document
  - Acts as a checklist so that no sections are forgotten
- Define standard templates for the requirements description
  - Easy to find information
  - Nothing will be “forgotten”

Teller Machine Requirement (bad)

2.6 Withdrawal

If the card is accepted, the user has entered the correct PIN, and there are sufficient funds in the account, the amount of cash shall be dispensed. If the card is invalid (in which case it should be ejected), the PIN does not match the one required for the card (in which case a tone shall sound and the user given the option to try again—the tries shall be limited to 3), or the balance is insufficient (in which case a tone shall sound and the user shall have the opportunity to enter a new amount) cash shall not be dispensed.

Validate PID Definition

2.6: The System shall support cash withdrawals by the user

2.6.1: A withdrawal shall be allowed if and only if:

- The card can be validated (Req. 2.7)
- The PIN is valid for the card (Req. 2.8)
- The funds in the card account exceeds the funds requested in the withdrawal

2.6.2: If a withdrawal is allowed (2.6.1), the exact amount requested shall be dispensed

Train protection system

- The deceleration of the train shall be computed as:
  - $D_{train} = D_{control} + D_{gradient}$

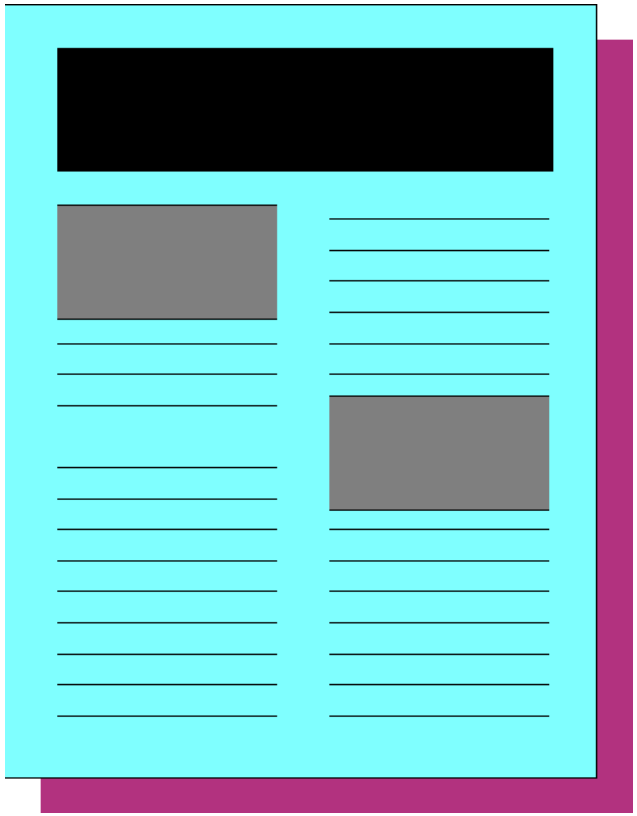
  where $D_{gradient}$ is $9.81ms^2$ * compensated gradient/alpha and where the values of $9.81ms^2$/alpha are known for different types of train.

Requirements Template (suggestion)

Number:	A unique requirements number
Use Case:	Reference to use case using Req.
Introduction*:*	What is the requirement about?
Rationale:	Why is the requirement here?
Source*:*	Who came up with the requirement?
Author*:*	Who wrote it down?
Inputs*:*	What comes in?
Required Function*:*	What is the requirement?
Outputs*:*	What comes out?
Related Reqs*:*	What else is related?
Conflicts:	Requirements in conflict with this one
Support Material:	Docs., Figures, Tables, Etc.
Test Cases*:*	How do we test the requirement?
Date:	When the requirement was modified
Priority:	How important is this requirement

Requirements in Forms

- Example available on the web



Four Easy Requirements Guidelines

- Avoid requirements “fusion”
  - One requirement per requirement specification
- Be precise
  - No vague requirements
- Be rigorous in defining requirements test cases
  - If you cannot define how to test if a requirement is satisfied, you probably have a poor requirement
- Attach a person to each requirement
  - People are much less likely to add “the kitchen sink” if their name is there – no gold plating

Requirements Fusion

2.16.3.f

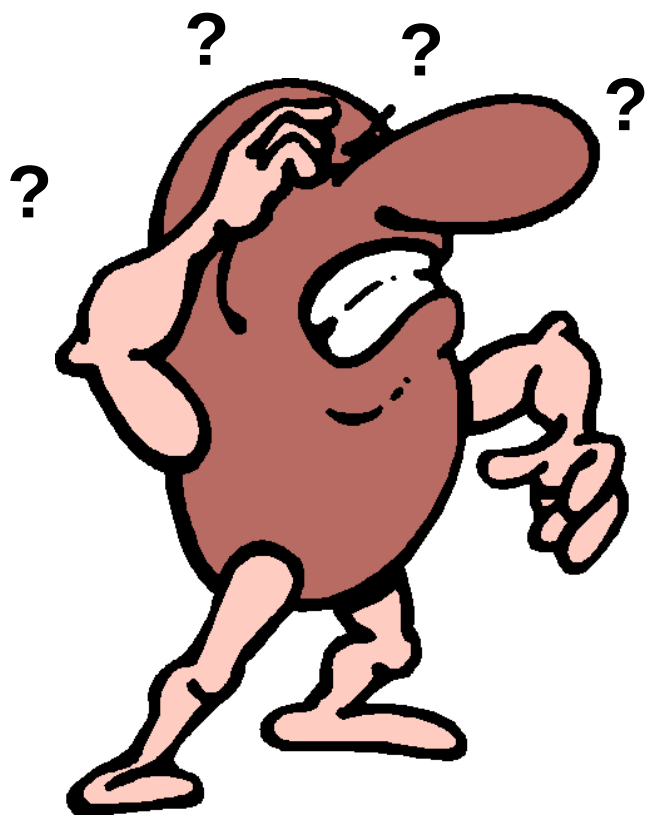
While acting as the bus controller, the C&C MDM CSCI shall set the e, c, w, indicators identified in Table 3.2.16-II for the corresponding RT to “failed” and set the failure status to failed for all RT’s on the bus upon detection of transaction errors of selected messages to RTs whose 1553 FDIR is not inhibited in two consecutive processing frames within 100 milliseconds of detection of the second transaction error if; a backup BC is available, the BC has been switched in the last 20 seconds, the SPD card reset capability is inhibited, or the SPD card has been reset in the last 10 major (10 second) frames, and either

1. the transaction errors are from multiple RT’s, the current channel has been reset within the last major frame, or

2. the transaction errors are form multiple RT’s, the bus channel’s reset capability is inhibited, and the current channel has not been reset within the last major frame.

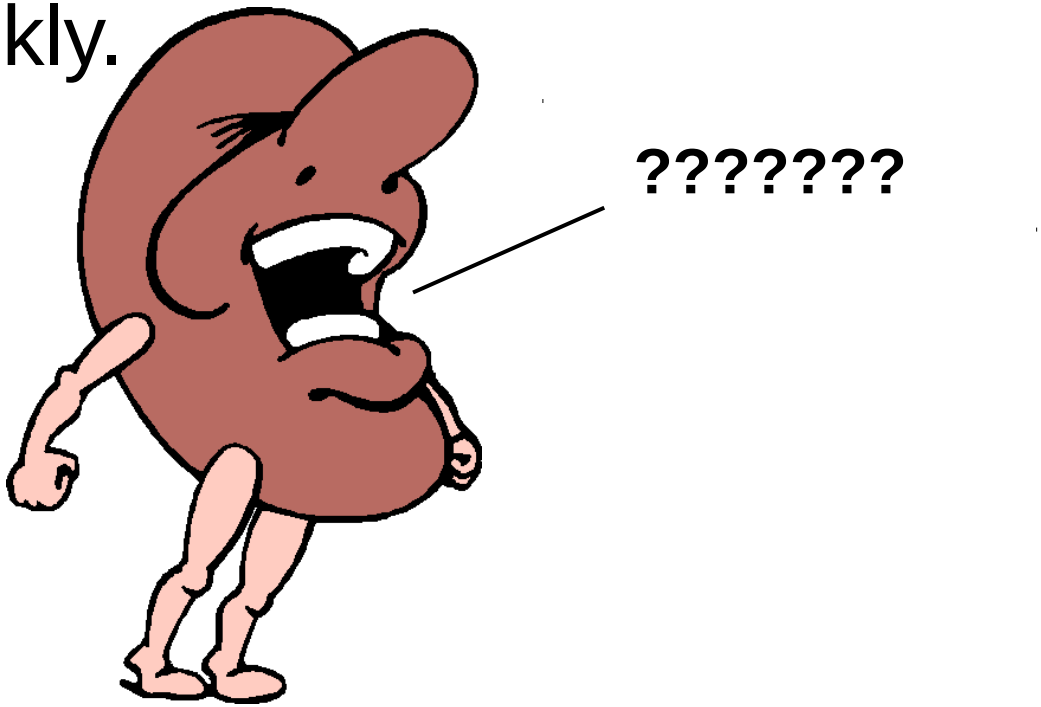
Vague and Ambiguous Requirement

- Charge numbers should be validated on-line against the master corporate charge number list, if possible.



Non-testable Requirement

- If a failure occurs (either internal or external), an easy to interpret alarm must be raised quickly.



Each Requirement Must Be

- Correct
  - The requirement is free from faults.
- Precise, unambiguous, and clear
  - Each item is exact and not vague; there is a single interpretation; the meaning of each item is understood; the specification is easy to read.
- Complete
  - The requirement covers all aspects of the user function.
- Consistent
  - No item conflicts with another item in the specification.

Each Requirement Must Be (Cont.)

- Relevant
  - Each item is pertinent to the problem and its solution.
- Testable
  - During program development and acceptance testing, it will be possible to determine whether the item has been satisfied.
- Traceable
  - Each item can be traced to its origin in the problem environment.
- Feasible
  - Each item can be implemented with the available techniques, tools, resources, and personnel, and within the specified cost and schedule constraints

The SRS (as a document) Must Be

- Complete
  - All user requirements have been included. Do not forget abnormal and boundary cases.
- Consistent
  - No item conflicts with another item in the specification.
- The requirements shall be at a consistent level of detail
- Manageable and Modifiable
  - Things will change and we must be able to accommodate the inevitable requirements evolution.

Example 1

- The product shall provide status messages at regular intervals not less than every 60 seconds

- Is this a good requirement?

Example 1 Rewritten

1. Status Messages

1.1 The Background Task Manager shall display status messages in a designated area in the user interface at intervals of 60 plus or minus 10 seconds.

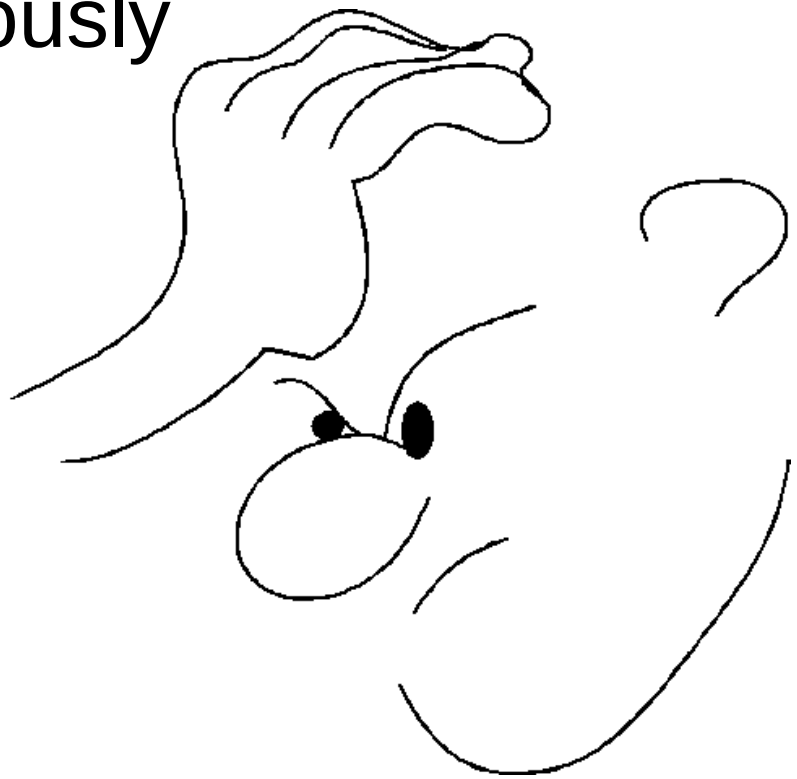
1.2 If background processing is progressing normally, the percentage of the background task processing that has been completed shall be displayed.

1.3 A message shall be displayed when the background task is complete.

1.4 An error message shall be displayed if the background task has stalled or failed.

Example 2

- The product shall switch between displaying and hiding non-printing characters instantaneously



Example 3

- Charge numbers should be validated on-line against the master corporate charge number list, if possible.

- The system shall validate the charge number entered against the on-line master corporate charge number list. If the charge number is not found on the list, an error message shall be displayed and the order shall not be accepted.

Complete?

1. Validate charge card number

1.1 The system shall validate the charge card number entered against the on-line master corporate charge card number list.

1.1.1 If the charge card number is not found on the list, an error message shall be displayed and the order shall not be accepted

Completed?

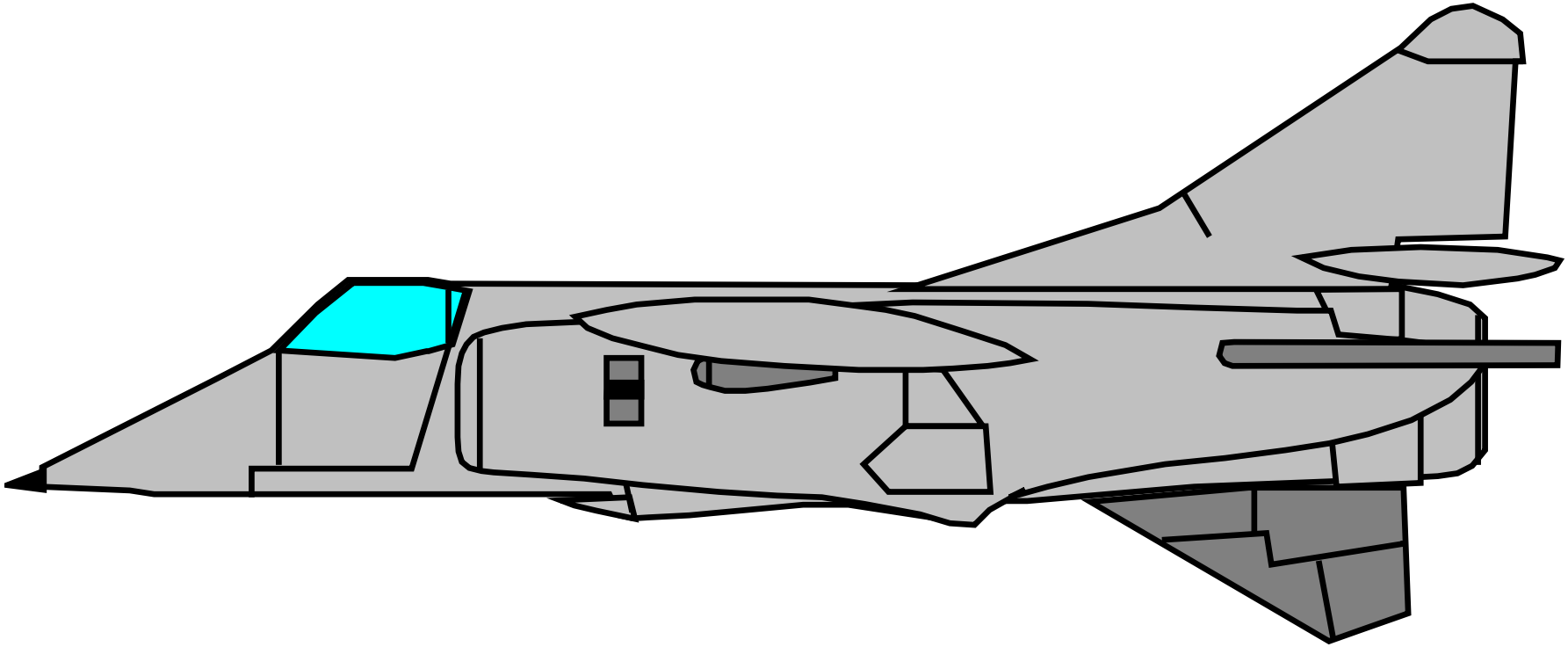
1. Validate charge card number

1.1 The system shall validate the charge card number entered against the on-line master corporate charge card number list.

1.1.1 If the charge card number is not found on the list, an error message shall be displayed and the order shall not be accepted

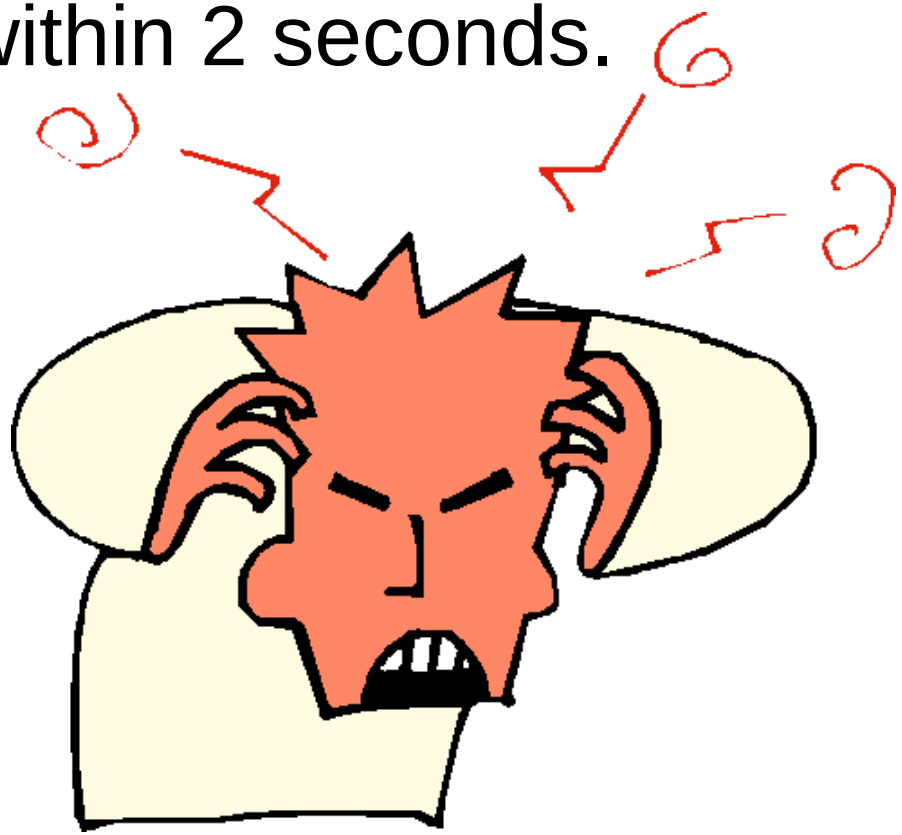
1.2 If the on-line master corporate charge card number list is not available, the order shall not be accepted.

Incomplete Requirements



Example 4

- The system shall respond to all user requests within 2 seconds.



Inconsistent Requirements

- Obvious inconsistencies
  - If the user selects option X, function Y will be performed
  - If the user selects option X, function Z will be performed
- Inconsistency from the driving handbook
  - You must obey the speed limit
  - You must maintain a speed that is consistent with the other traffic

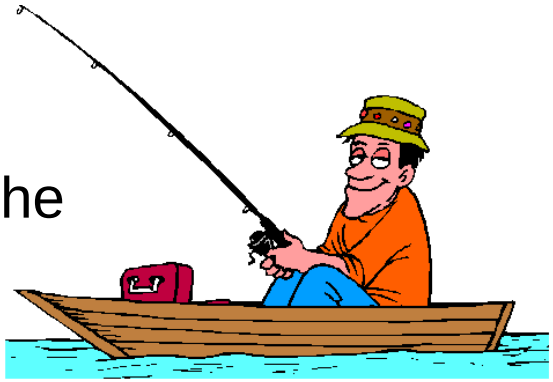
Requirements Rationale

- It is important to provide rationale with requirements
- This helps the developer understand the application domain and why the requirement is stated in its current form
- Particularly important when requirements have to be changed
  - The availability of rationale reduces the chances that change will have unexpected effects

Why Rationale?

Market Requirements Definition

1. One person must be able to load the boat on the car rack



(Software) Requirements Specification

1.1 The boat must be lighter than 100 lb.

1.2 The boat must have handles to help one person lift it

1.3 The car rack must be padded so the boat can easily slide into the rack

1.4 Etc.

Requirements Traceability

- Requirements traceability means that related requirements are linked in some way and that requirements are (perhaps) linked to their source
- Traceability is a property of a requirements specification which reflects the ease of finding related requirements
- Some CASE tools provide traceability support facilities
  - For example, they may be able to find all requirements which use the same terms

Traceability Techniques

- Assign a unique number to all requirements
- Cross-reference related requirements using this unique number
- Produce a cross-reference matrix for each requirements document showing related requirements.
  - Several matrices may be necessary for different types of relationship

Traceability Matrix

ReqID	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1			R					
1.2		U	U			R		U
1.3	R			R				
2.1			R		U			U
2.2								U
2.3		R		U				
3.1								R
3.2							R	

U = “uses the requirement”, **R** = “Some other weaker relationship”

When Are We Done??

- When all the stakeholders are happy
  - Remember the stakeholders are diverse and more numerous that you may think
  - Customer
  - Sales department
  - Engineers and developers
  - Testers
  - Etc.

- The document has passed all inspections and checklists
- All TBD have been closed out

The Use of TBD

- Any SRS using the term “to be determined” (TBD) is not complete

- TBD is often needed, however
  - A description of the condition causing the TBD (why the issue cannot be resolved)
  - A description of what must be done to eliminate the TBD, who is responsible, and by when it must be done

HW1 Discussion

Cruise Control System

- Requirements
- Use Cases
- Use Case Diagram (Papyrus Tool)

We Have Learned

- Use a standard document structure and forms for the individual requirements
- Use checklists to make sure the individual requirements are “good”
- Use checklists to make sure the document is “good”
- Keep an eye towards the future – things will change
  - What requirements are likely to change
  - Structure the your requirements accordingly
  - Provide rationale and traceability
- Make sure all stakeholders agree on the requirements document