RE Basics :
Purpose and Nature of Requirements

Gregor v. Bochmann, University of Ottawa

Based on Powerpoint slides prepared by Gunter Mussbacher
with material from:

Sommerville & Kotonya 1998, Lethbridge & Laganière 2001-2005, Hooks & Farry 2001, Bray 2002, Pressman 2005, Amyot 2005-2009, Somé 2008

uOttawa

Table of Contents

- Definition and Importance of Requirements

- Types of Requirements
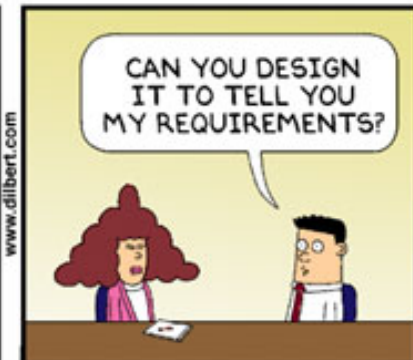
- The beginning is the most important part of the work.[1]

[1] Plato, 4 B.C.

uOttawa

Mars Climate Orbiter

- In 1999, the Mars Climate Orbiter disappears around Mars

- Cost: about $125M US

- Problem caused by a misunderstanding between a team in Colorado and one in California

- One team used the metric system while the other used the English system for a key function…

GIRES

- GIRES[1] (Gestion intégrée des ressources)
  - Integrated management of resources
  - To replace >1000 existing systems
  - In 140 organisations / departments
  - Affecting 68000 employees!
- 8-year project of the Quebec government, started 1998
- $80 million budget
- Could not cope with changes to the requirements…
  - Cost of $400 millions after 5 years, and very late
  - Project cancelled in 2003

[1] http://radio-canada.ca/nouvelles/Index/nouvelles/200303/04/012-GIRES.shtml

uOttawa

Canadian Gun Registry[1,2]

- Law adopted in 1995
- Was supposed to cost $119M, with revenues of $117M (net cost of $2M)
- 30 types of permits, long questionnaires, 90% of errors in requests
- Rising costs ($327M in 2000, $688M in 2002, plus others…)
- Many political and legal issues, and a few scandals…
- Customer asked for over 2000 changes in the computer system!
- ~$1B in 2004, probably ~$2B by the time the system is fully functional
- Tons of unhappy customers and taxpayers…
- Not required to register as of May 17, 2006!

[1] http://en.wikipedia.org/wiki/Canadian_gun_registry
[2] http://radio-canada.ca/actualite/zonelibre/04-02/registre_armes.asp

uOttawa

Definition and Importance of Requirements

You said "Requirements"?

- A requirement is:

  - Capturing the purpose of a system

  - An expression of the ideas to be embodied in the system or application under development

  - A statement about the proposed system that all stakeholders agree must be made true in order for the customer's problem to be adequately solved

    - Short and concise piece of information

    - Says something about the system

    - All the stakeholders have agreed that it is valid

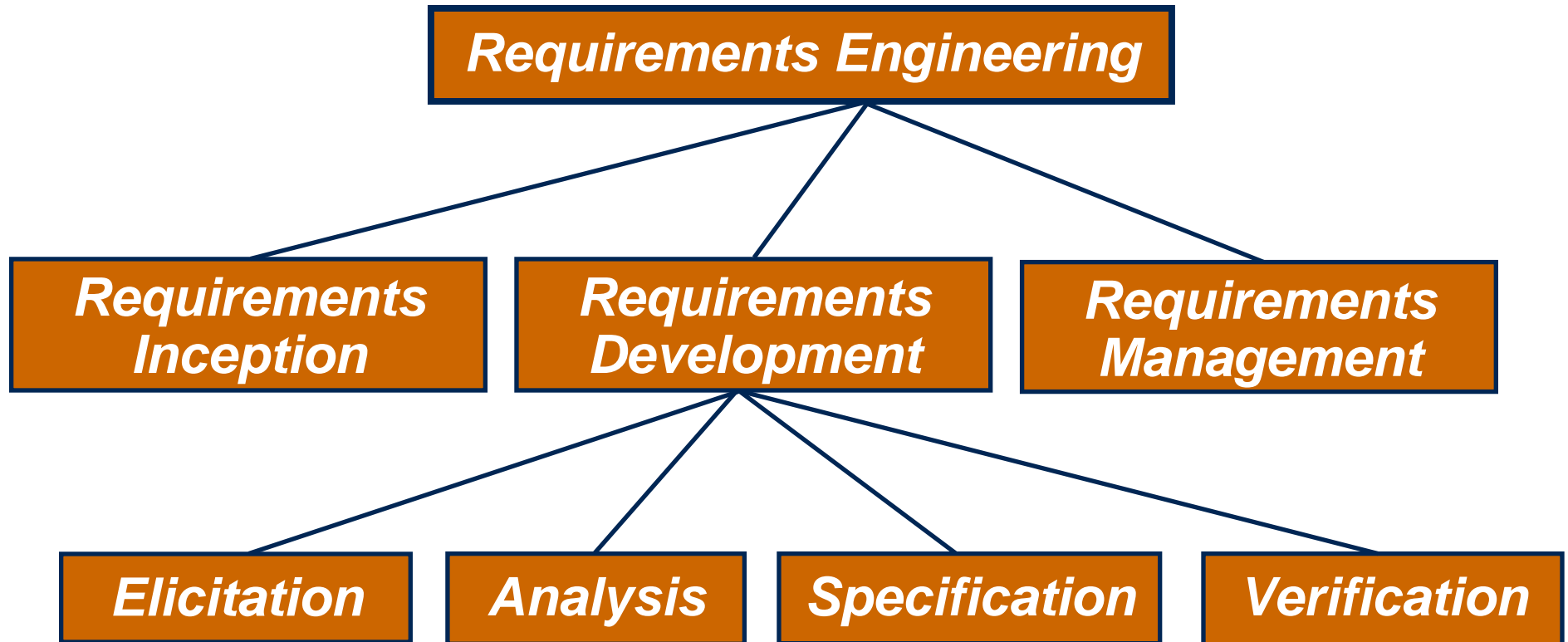   - It helps solve the customer's problem

According to IEEE 830-1993

- A **requirement** is defined as:

  - A condition or capability needed by a user to solve a problem or achieve an objective

  - A condition or a capability that must be met or possessed by a system … to satisfy a contract, standard, specification, or other formally imposed document …

You said "Requirements Engineering"?

- **Requirements Engineering** (RE) is:
  - The activity of development, elicitation, specification, analysis, and management of the stakeholder requirements, which are to be met by a new or evolving system
  - RE is concerned with identifying the purpose of a software system… and the contexts in which it will be used
    - How/where the system will be used
    - Big picture is important
  - Captures real world needs of stakeholders affected by a software system and expresses them as artifacts that can be implemented by a computing system
    - Bridge to design and construction
    - How to communicate and negotiate?
    - Is anything lost in the translation between different worlds?

Requirements Engineering Activities

```
                    ┌──────────────────────────────────┐
                    │   Requirements Engineering        │
                    └──────────────────────────────────┘
           ┌────────────────┬──────────────────┬────────────────┐
  ┌─────────────────┐  ┌─────────────────┐  ┌─────────────────┐
  │  Requirements    │  │  Requirements    │  │  Requirements    │
  │  Inception       │  │  Development     │  │  Management      │
  └─────────────────┘  └─────────────────┘  └─────────────────┘
                ┌──────────┬─────────┴───────┬──────────┐
        ┌────────────┐ ┌──────────┐ ┌───────────────┐ ┌───────────────┐
        │ Elicitation │ │ Analysis │ │ Specification  │ │ Verification   │
        └────────────┘ └──────────┘ └───────────────┘ └───────────────┘
```

Source: Larry Boldt, Trends in Requirements Engineering   People-Process-Technology, Technology Builders, Inc., 2001

uOttawa

About these RE Activities…

- Inception

  - Start the process (business need, market opportunity, great idea, ...), business case, feasibility study, system scope, risks, etc.

- Requirements elicitation

  - Requirements discovered through consultation with stakeholders

- Requirements analysis and negotiation

  - Requirements are analyzed and conflicts resolved through negotiation

- Requirements specification

  - A precise requirements document is produced

- Requirements validation

  - The requirements document is checked for consistency and completeness

- Requirements management

  - Needs and contexts evolve, and so do requirements!

uOttawa

General Problems with the Requirements Process

- Lack of the right expertise (software engineers, domain experts, etc.)

- Initial ideas are often incomplete, wildly optimistic, and firmly entrenched in the minds of the people leading the acquisition process

- Difficulty of using complex tools and diverse methods associated with requirements gathering may negate the anticipated benefits of a complete and detailed approach
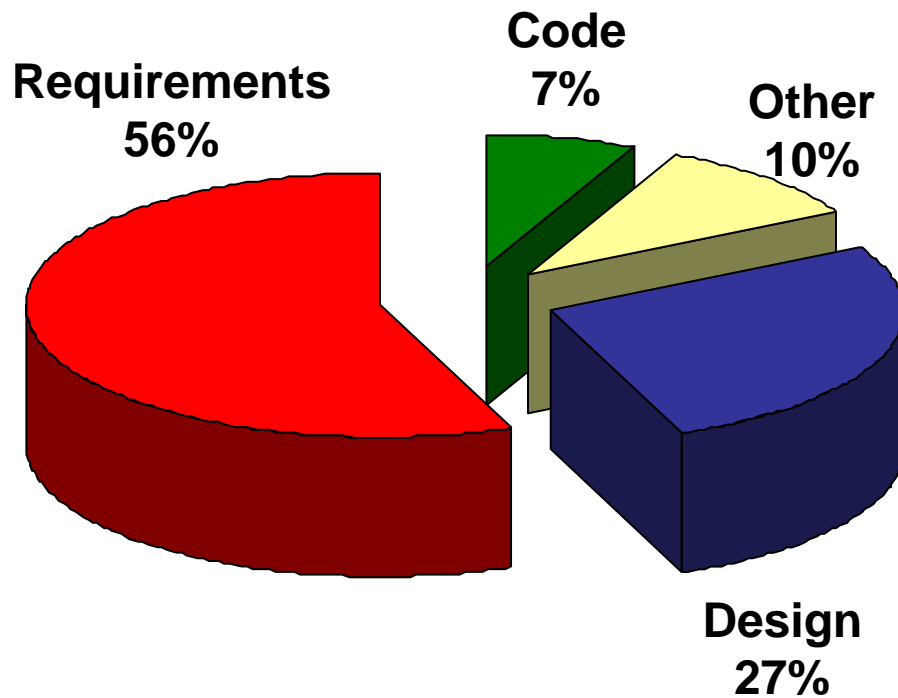
u Ottawa

Statistics from NIST Report

- NIST (National Institute of Standards and Technology) has published a comprehensive (309 pages) and very interesting report on project statistics and experiences based on data from a large number of software projects[1]

  - 70% of the defects are introduced in the specification phase

  - 30% are introduced later in the technical solution process

  - Only 5% of the specification inadequacies are corrected in the specification phase

  - 95% are detected later in the project or after delivery where the cost for correction on average is 22 times higher compared to a correction directly during the specification effort

  - The NIST report concludes that extensive testing is essential, however testing detects the dominating specification errors late in the process
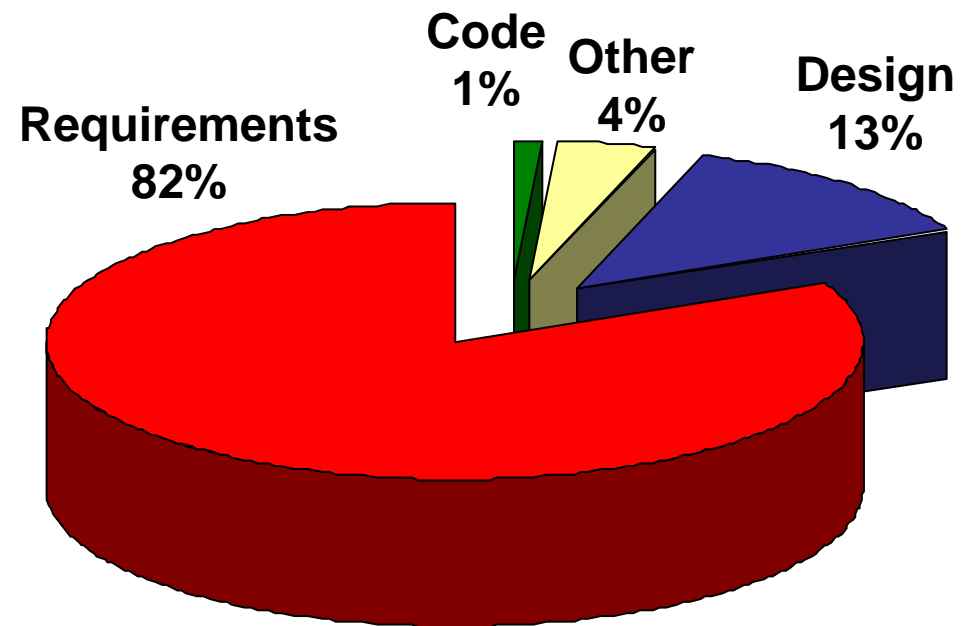
[1] http://www.nist.gov/public_affairs/releases/n02-10.htm (May 2002)

SEG3101 (Fall 2010).   Basics – nature and purpose of requirements

Why Focus on Requirements ?

- Distribution of Defects
- Distribution of Effort to Fix Defects

Code
7%

Requirements
56%

Other
10%

Design
27%

Code
1%

Other
4%

Design
13%

Requirements
82%

Source: Martin & Leffinwell

View of the Software Engineering Institute (SEI)

- Improve software development with the CMM/CMMI model for software development

  - Capability Maturity Model (CMM)

  - For software development, superseded by Capability Maturity Model Integration (CMMI)

- SEI's vision is:

  - The right software, delivered defect free, on time & on cost, every time

  - "Right software" implies software that satisfies requirements for functionality and qualities (e.g., performance, cost…) throughout its lifetime

  - "Defect free" software is achieved either through exhaustive testing after coding or by developing the code right the first time

u Ottawa

CHAOS Report (2004)[1]

RESOLUTION OF PROJECTS

This year's results show that 29% of all projects succeeded (delivered on time, on budget, with required features and functions); 53% are challenged (late, over budget and/or with less than the required features and functions); and 18% have failed (cancelled prior to completion or delivered and never used), as shown in Figure 2.0.
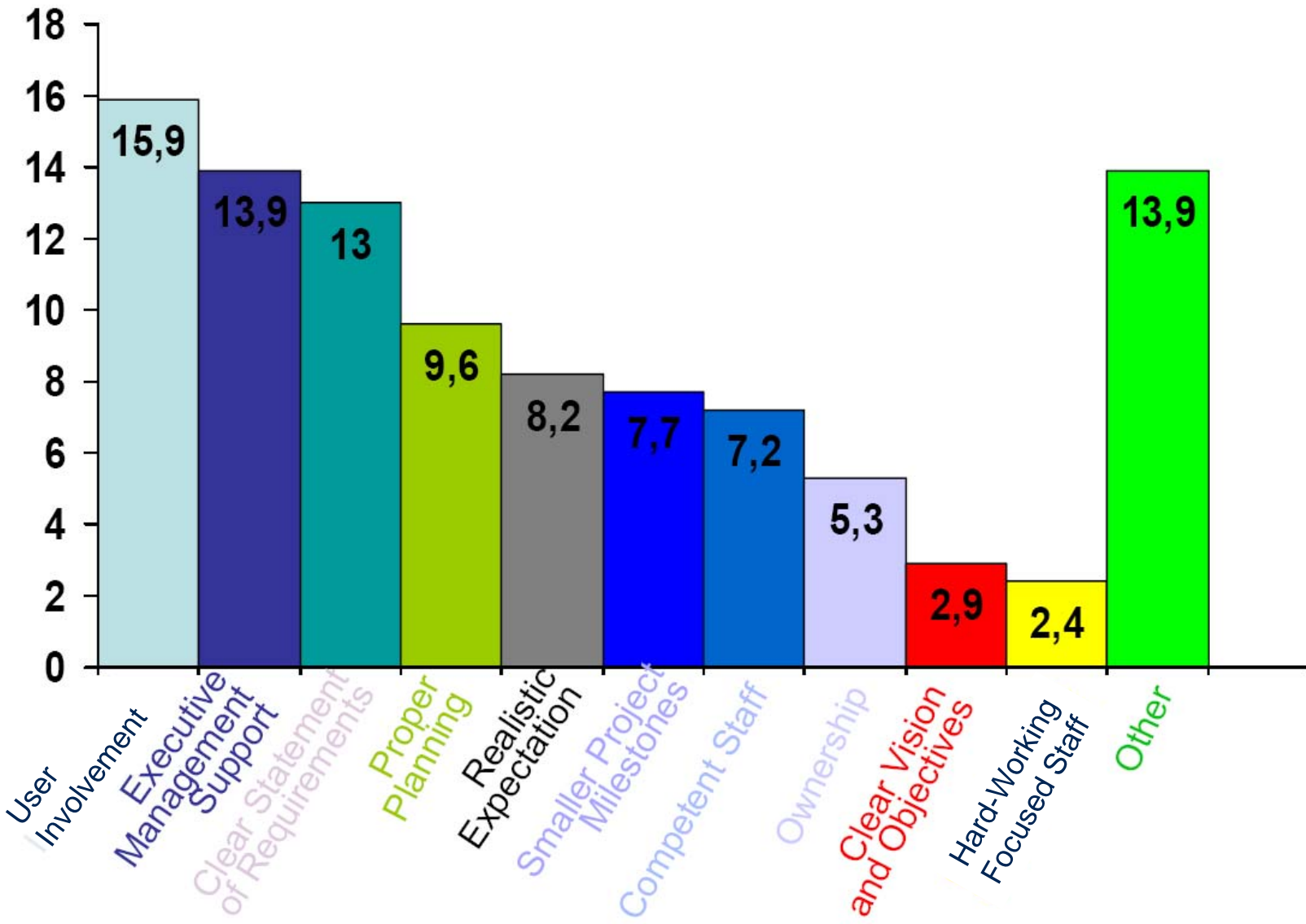
Challenged
53%

Succeeded
29%

Failed
18%

Figure 2.0

[1] Standish Group Inc., 2004

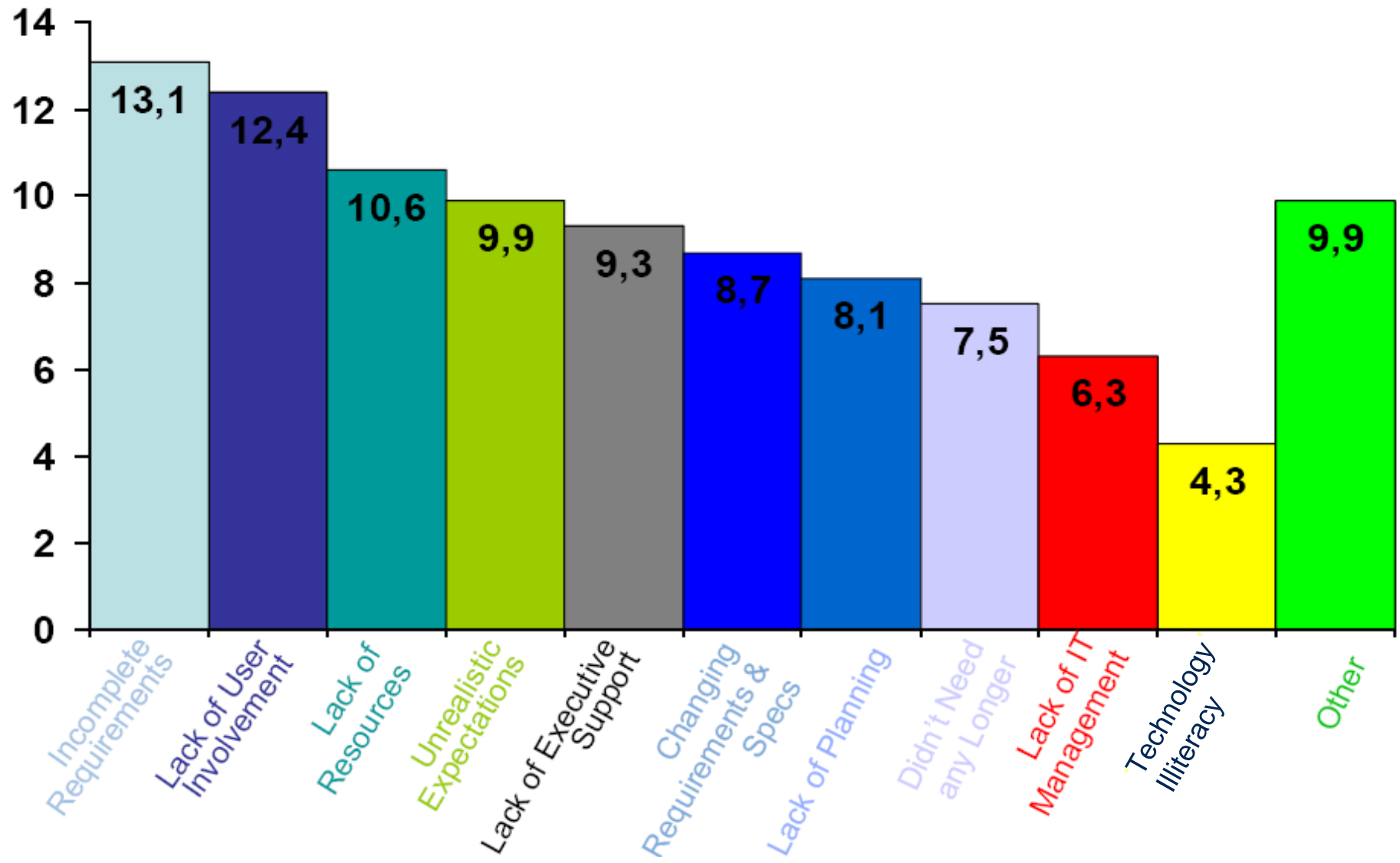uOttawa

Progression since 1994
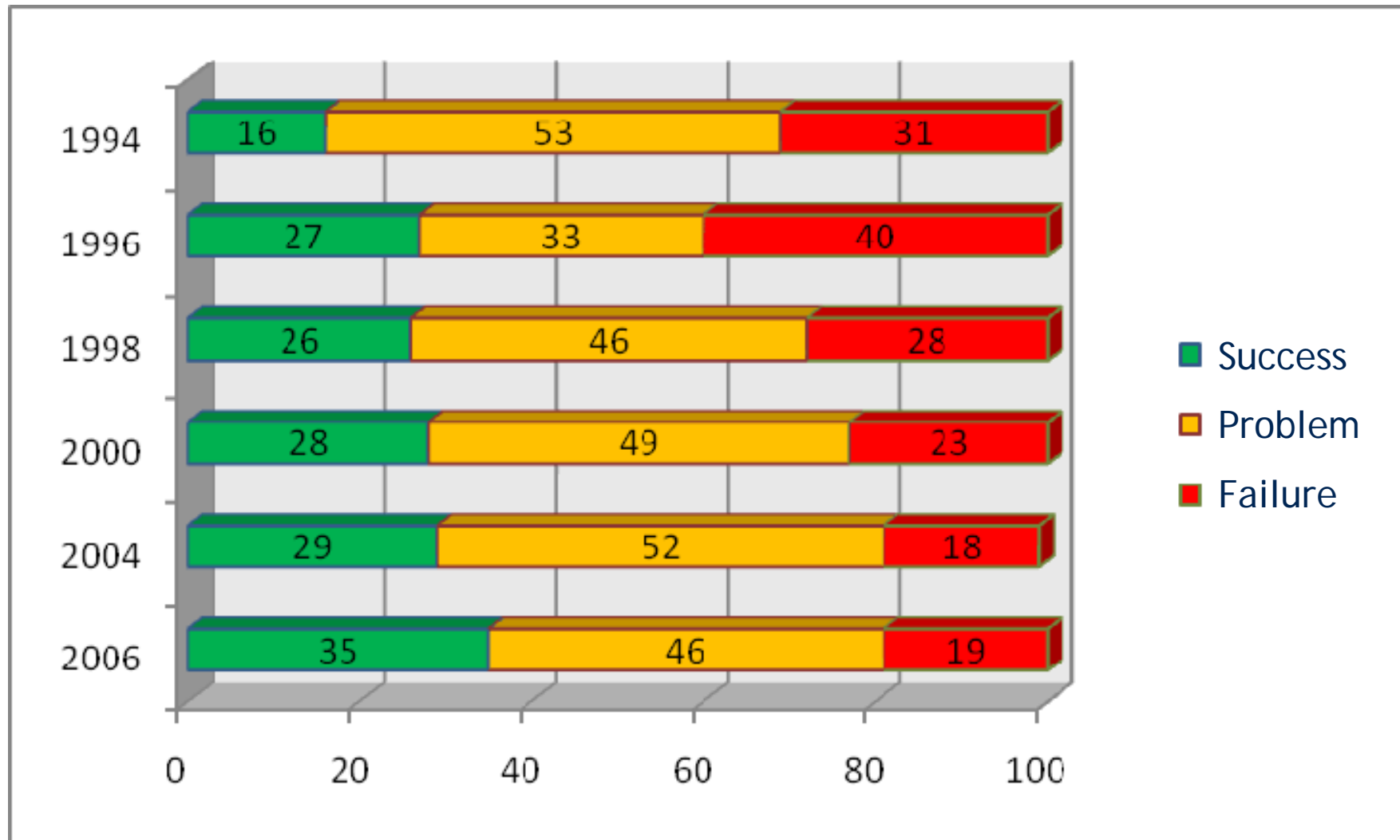
Source: Standish Group Inc., 1994-2006

uOttawa

Success Factors

Source: Standish Group Inc., 1995

u Ottawa

Problem Causes

Source: Standish Group Inc., 1995

Evolution of Success Factors

CHAOS

1994	1996	1999	2000
User involvement	User involvement	User involvement	Executive management support
Executive management support	Executive management support	Executive management support	User involvement
Clear statement of requirements	Clear statement of requirements	Clear statement of requirements	Experienced project manager
Proper planning	Firm basic requirements	Experienced project manager	Clear business objectives
Realistic expectations	Competent staff	Small project milestones	Minimized scope
Small project milestones	Small project milestones	Firm basic requirements	Standard software infrastructure
Competent staff	Experienced project manager	Competent staff	Firm basic requirements
Ownership	Proper planning	Proper planning	Formal methodology
Clear vision and objectives	Ownership	Ownership	Reliable estimates
Hard-working, focused staff	Other	Other	Other

Involvement/Ownership	Requirements	Management Support	Planning	Staff

Source: Standish Group Inc., 2000

uOttawa

Managing Evolving Requirements

"Changing requirements is as certain as death and taxes"

Requirement tools: These seem to have the biggest impact on the success of a project. This may seem strange since "Firm Basic Requirements" is number six on the top ten list. However these tools, if used as a platform for communications between all the stakeholders, such as executive sponsors and users, can provide enormous benefits. This tool needs to be at the top of the shopping list for any firm involved in developing software applications.

Source: http://standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf, 1999

u Ottawa

Types of Requirements

So Many "Requirements"… (1)

- A goal is an objective or concern that guides the RE process. It can be used to discover and evaluate functional and non-functional requirements
  - A goal is not yet a requirement…
- Note: All requirements must be verifiable (by some test, inspection, audit etc.)
- A functional requirement is a requirement defining functions of the system under development
  - Describes what the system should do
- A non-functional requirement is a requirement that is not functional. This includes many different kinds of requirements. – Therefore one often considers the following sub-categories:

uOttawa

Different types of non-functional requirements

- **Performance requirements**, characterizing system properties such as expected performance, capacity, reliability, robustness, usability, etc.

- **Design constraints** (also called **process requirements**), providing constraints on how the system should be designed and built – related to development process, documentation, programming language, maintainability, etc.

- **Commercial constraints**, such as development time frame and costs.

So Many "Requirements"… (2)

- A user requirement is a desired goal or function that a user and other stakeholders expect the system to achieve

  - Does not necessarily become a system requirement

- A business rule is a requirement derived from business practices within a given industrial sector, or in a given company, or defined by government regulations or standards.

  - May lead to system requirements. Can be functional or non-functional

- Problem domain requirements should be satisfied within the problem domain in order to satisfy some of the goals

- System requirements are the requirements for the system to be built, as a whole

  - A system is a collection of interrelated components working together towards some common objective (may be software, mechanical, electrical and electronic hardware and be operated by people)

  - Systems Engineering is a multidisciplinary approach to systems development – software is only a part (but often the problematic part)

uOttawa

26

So Many "Requirements"… (3)

- Important note: Software Requirements Engineering is a special case of Requirements Engineering. Many topics discussed in this course are quite general and apply to requirements engineering, in general.

- In a system containing software, software requirements are derived from the system requirements. The system then consists of hardware and software, and the hardware (and often the operating system and other existing software modules) are part of the environment in which the software is used.

u Ottawa

Functional Requirements

- What inputs the system should accept
- What outputs the system should produce
- What data the system should store other systems might use
- What computations the system should perform
- The timing and synchronization of the above

- Depend on the type of software, expected users, and the type of system where the software is used
- Functional user requirements may be high-level statements of what the system should do, but functional system requirements should describe the system services in detail

u Ottawa

Examples of Functional Requirements

- The user shall be able to search either all of the initial set of databases or select a subset from it.

- The system shall provide appropriate viewers for the user to read documents in the document store.

- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.

Note: not all requirements on this and following slides are high quality requirements but are typical requirements found too often in documents

Non-Functional Requirements (NFR) (1)

- Non-functional requirements are important

  - If they are not met, the system is useless

  - Non-functional requirements may be very difficult to state precisely (especially at the beginning) and imprecise requirements may be difficult to verify

- They are sometimes called quality requirements, quality of service, or extra-functional requirements.

- *Three main categories [1]:*

  - **Performance requirements** reflecting: usability, efficiency, reliability, maintainability and reusability (note: also security requirements)
    - Response time, throughput
    - Resource usage
    - Reliability, availability
    - Recovery from failure
    - Allowances for maintainability and enhancement
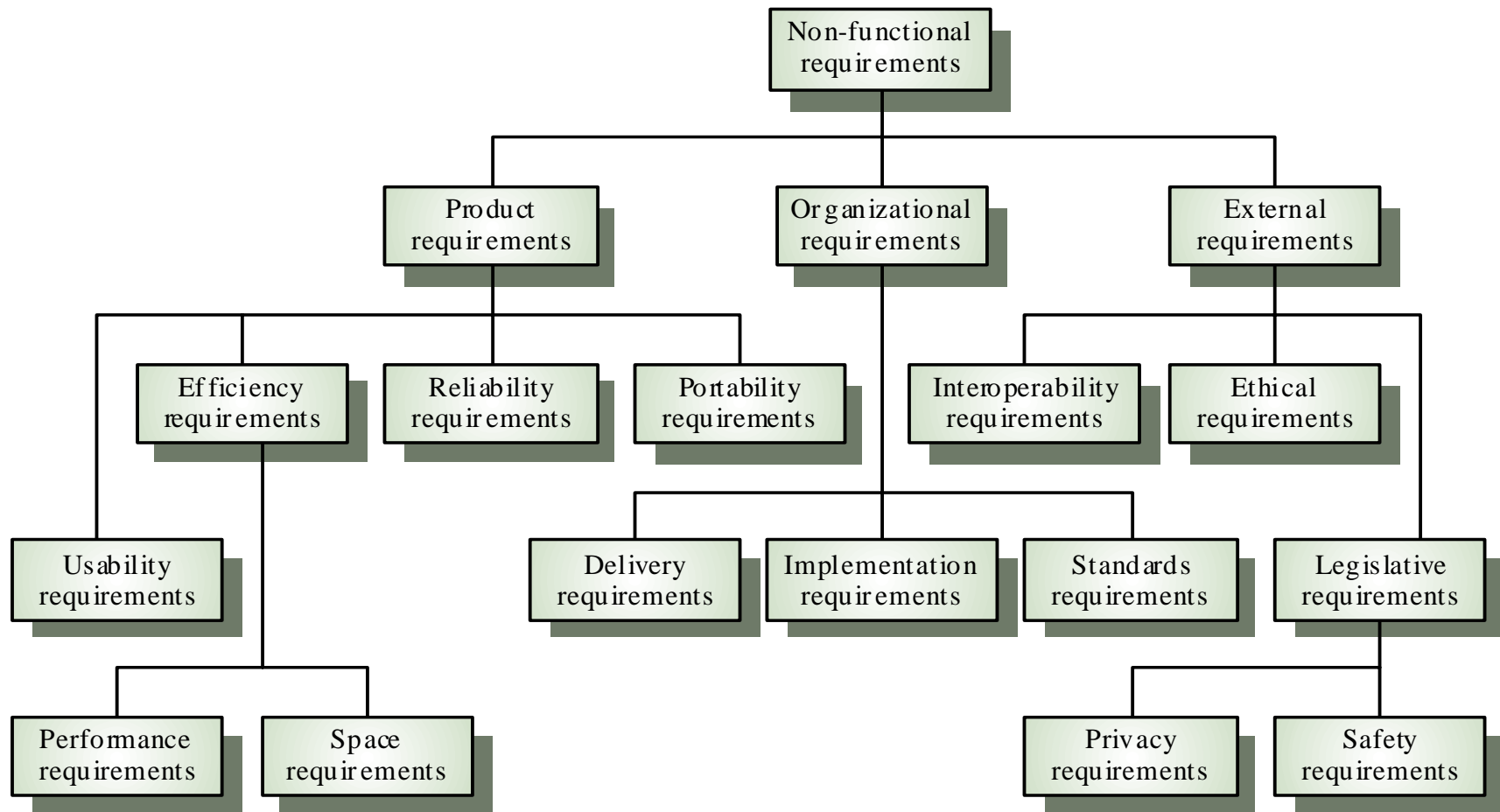   - Allowances for reusability

[1] Lethbridge and Laganière, Object Oriented Software Engineering: Practical Software Development using UML and Java, 2005

uOttawa

Non-Functional Requirements (NFR) (2)

- **Design constraints:** Categories constraining the environment and technology of the system.

  - Platform (minimal requirements, OS, devices…)

  - Technology to be used (language, DB, …)

- **Commercial constaints:** Categories constraining the project plan and development methods

  - Development process (methodology) to be used

  - Cost and delivery date

    - Often put in contract or project plan instead

uOttawa

Various NFR Types

- *Other ontologies also exist*


```
                         Non-functional
                          requirements

        Product              Organizational         External
      requirements            requirements        requirements

  Efficiency    Reliability    Portability    Interoperability    Ethical
  requirements  requirements   requirements    requirements    requirements

Usability                    Delivery    Implementation  Standards    Legislative
requirements                 requirements requirements   requirements requirements

Performance    Space                                    Privacy      Safety
requirements   requirements                            requirements  requirements
```

Source: Gerald Kotonya and Ian Sommerville, Requirements Engineering – Processes and Techniques, Wiley, 1998

uOttawa

Examples of Non-Functional Requirements

- Product requirement

  - It shall be possible for all necessary communication between the APSE and the user to be expressed in the standard Ada character set.

- Process requirement

  - The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCoSPSTAN95.

- Security requirement

  - The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.

uOttawa

Measurable Non-Functional Requirements

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Source: Gerald Kotonya and Ian Sommerville, Requirements Engineering – Processes and Techniques, Wiley, 1998

Goals

- A Goal

  - Conveys the intention or the objective of one or many stakeholders

  - Can guide the discovery of verifiable non-functional requirements that can be tested objectively

uOttawa

Example of Goal and NFR

- A system goal

  - The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized.

- A verifiable usability requirement derived from this goal

  - Experienced controllers shall be able to use all the system functions after a total of three hours of training.

  - The average number of errors made by experienced controllers shall not exceed two per day.

  - Assumption: An experienced controller has at least 2 years experience with the old system (as stated by the stakeholder)

Application-Domain Requirements

- Derived from the application domain

- Describe system characteristics and features that reflect the domain

- May be new functional requirements, constraints on existing requirements, or define specific computations

- If domain requirements are not satisfied, the system may be unworkable

uOttawa

Examples of Application-Domain Requirements

- Library system

  - The system interface to the database must comply with standard Z39.50.

  - Because of copyright restrictions, some documents must be deleted immediately on arrival. Depending on the user's requirements, these documents will first be printed either locally or printed to a network printer and retrieved by the user.

- 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.

u Ottawa

Problems concerning Application-Domain Requirements

- ## Understandability

  - Requirements are expressed in the language of the application domain

  - This is often not understood by software engineers developing the system

- ## Implicitness / Tacit knowledge

  - Domain specialists understand the area so well that they do not think of making the domain requirements explicit

  - People are often unaware of the tacit knowledge they possess and therefore cannot express it to others

Emergent Properties (when the system consists of several sub-systems)

- Properties of the system as a whole

  - Requirements which cannot be addressed by a single component, but which depend for their satisfaction on how all the software components interoperate

  - Only emerge once all individual subsystems have been integrated

  - Dependent on the system architecture

- Examples of emergent properties

  - Reliability

  - Maintainability

  - Performance

  - Usability

  - Security

  - Safety

uOttawa

For More Information

B. A. Nuseibeh and S. M. Easterbrook, Requirements Engineering: A Roadmap. In A. C. W. Finkelstein (ed) The Future of Software Engineering, ACM Press, 2000
http://www.cs.toronto.edu/~sme/papers/2000/ICSE2000.pdf

Simson Garfinkel, History's Worst Software Bugs, Wired News, 2005
http://www.wired.com/news/technology/bugs/0,2924,69355,00.html

INCOSE Requirements Working Group
http://www.incose.org/practice/techactivities/wg/rqmts/

Tools Survey: Requirements Management (RM) Tools
http://www.incose.org/productspubs/products/rmsurvey.aspx
http://www.volere.co.uk/tools.htm

IEEE (1993) Recommended Practice for Software Requirements Specifications. IEEE Std 830-1993, NY, USA.

IEEE (1995) Guide for Developing System Requirements Specifications. IEEE Std P1233/D3, NY, USA.

Requirements Engineering Conference
http://www.requirements-engineering.org/

uOttawa

Main References

Jeremy Dick, Elizabeth Hull, Ken Jackson: Requirements Engineering, Springer-Verlag, 2004

Soren Lauesen: Software Requirements - Styles and Techniques, Addison Wesley, 2002

Ian K. Bray: An Introduction to Requirements Engineering, Addison Wesley, 2002

Karl E. Wiegers: Software Requirements, Microsoft Press, 2003

Gerald Kotonya, Ian Sommerville: Requirements Engineering – Processes and Techniques, Wiley, 1998

Roger S. Pressman: Software Engineering – A Practitioner's Approach, McGraw-Hill, 2005

Tim Lethbridge, Robert Laganière: Object Oriented Software Engineering: Practical Software Developement using UML and Java, 2nd edition, McGraw-Hill, 2005

Ivy F. Hooks, Kristin A. Farry: Customer-Centered Products – Creating Successful Products Through Smart Requirements Management, Amacom, 2001

CHAOS Report, Standish Group

uOttawa