Chapter 2 — Software Requirements

March 9, 2009

Outline

"The hardest single part of building a system is deciding what to build."

— Brooks, 1987

Introduction

- Description of:
  - system's functions (services provided)
  - operational constraints

Introduction

- Description of:
  - system's functions (services provided)
  - operational constraints
- **Requirements Engineering**: for the above:
  - acquire
  - analyze
  - document
  - check

Outline

User and System Requirements

- Customer's wishes:
  - abstract
  - general

User and System Requirements

- Customer's wishes:
  - abstract
  - general
- Developer's draft:
  - detailed
  - verified by customer

User and System Requirements

- Customer's wishes:
  - abstract
  - general
- Developer's draft:
  - detailed
  - verified by customer
- **User requirements**:
  - in *natural language* + *diagrams*
  - expectations & operating conditions

User and System Requirements

- Customer's wishes:
  - abstract
  - general
- Developer's draft:
  - detailed
  - verified by customer
- **User requirements**:
  - in *natural language* + *diagrams*
  - expectations & operating conditions
- **System requirements**:
  - detailed list of system functions, services, operational constraints
  - *system requirements document* or *functional specification*
  - precise
  - contract between customer and developer

Functional, non-functional requirements

- Another set of categories

Functional, non-functional requirements

- Another set of categories
- **functional**
  - services (functions) to be provided
  - response to inputs
  - what *not* to do
- **non-functional**
  - constraints on services; apply to whole system
  - *timing constraints*
  - *development constraints*
  - *standards*
- **domain**

Functional, non-functional requirements

- Another set of categories
- **functional**
  - services (functions) to be provided
  - response to inputs
  - what *not* to do
- **non-functional**
  - constraints on services; apply to whole system
  - *timing constraints*
  - *development constraints*
  - *standards*
- **domain**
- not always clear-cut

Functional Requirements

- Depends upon:
  - software being developed
  - expected users
  - developers' approach

Functional Requirements

- Depends upon:
  - software being developed
  - expected users
  - developers' approach
- *system* functional requirements are more detailed than *user* functional requirements

Functional Requirements

- Depends upon:
  - software being developed
  - expected users
  - developers' approach
- *system* functional requirements are more detailed than *user* functional requirements
- *completeness* and *consistency*

Non-functional Requirements

- usually for *emergent properties* of system
  - reliability
  - response time
  - storage, . . .

Non-functional Requirements

- usually for *emergent properties* of system
  - reliability
  - response time
  - storage, . . .
- or for *constraints* on system
  - data format
  - I/O capabilities
  - processing power, . . .

Non-functional Requirements

- usually for *emergent properties* of system
  - reliability
  - response time
  - storage, . . .
- or for *constraints* on system
  - data format
  - I/O capabilities
  - processing power, . . .
- difficult to verify

Non-functional Requirements

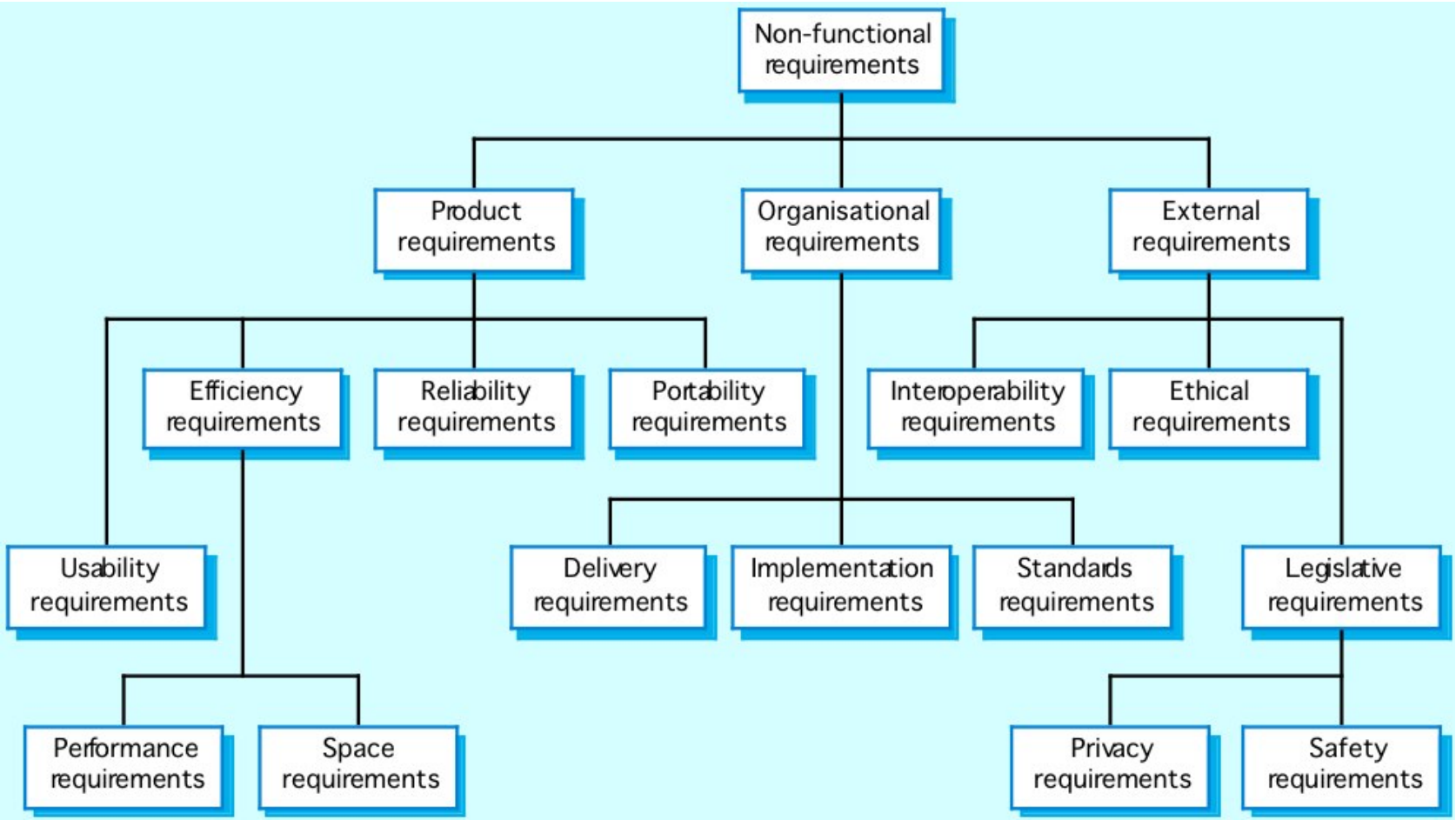


Figure: Types of non-functional requirements

Non-functional Requirements

- difficult to verify

Non-functional Requirements

- difficult to verify

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	M Bytes Number of ROM 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

Figure: Some Metrics for Non-functional Requirements

Non-functional Requirements

- difficult to verify

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	M Bytes Number of ROM 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

Figure: Some Metrics for Non-functional Requirements

- mix *goals* with *requirements*

Non-functional Requirements

- Non-functional $\leftrightarrow$ functional

Non-functional Requirements

- Non-functional ↔ functional
  - E.g.:
    - 1) respond in 0.5 seconds to path deviation
    - 2) use Java

Non-functional Requirements

- Non-functional ↔ functional
  - E.g.:
    - 1) respond in 0.5 seconds to path deviation
    - 2) use Java
- distinguish between functional & non-functional in documentation

Domain Requirements

- From application domain rather than user needs
- E.g.:
  - **HIPAA** requirements
  - **VPF** data format
  - **delete-on-print** for copyright compliance
  - "Deceleration of train hall be computed as "

$$D_{train} = D_{control} + D_{gradient}$$

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

Domain Requirements

- From application domain rather than user needs
- E.g.:
  - **HIPAA** requirements
  - **VPF** data format
  - **delete-on-print** for copyright compliance
  - "Deceleration of train hall be computed as "

$$D_{train} = D_{control} + D_{gradient}$$

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

- Developers must work with domain experts

User Requirements

- Natural language instead of jargon

User Requirements

- Natural language instead of jargon
- Possible issues:
  - lack of clarity
  - confusion on type
  - amalgamation

User Requirements

- Natural language instead of jargon
- Possible issues:
  - lack of clarity
  - confusion on type
  - amalgamation

  Example: "The website should allow editing of content on all pages. The admin should be able to grant various levels of access to the editors."
- separate **user** from **system** requirements

User Requirements Example

Bad

Grid facilities: To assist in the positioning of entities on a diagram, the user may turn on a grid in either centimetres or inches, via an option on the control panel. Initially, the grid is off. The grid may be turned on and off at any time during an editing session and can be toggled between inches and centimetres at any time. A grid option will be provided on the reduce-to-fit view but the number of grid lines shown will be reduced to avoid filling the smaller diagram with grid lines.

User Requirements Example

“Less” bad

2.6.1 Grid facilities

The editor shall provide a grid facility where a matrix of horizontal and vertical lines provide a background to the editor window. This grid shall be a passive grid where the alignment of entities is the user's responsibility.

Rationale: A grid helps the user to create a tidy diagram with well-spaced entities. Although an active grid, where entities 'snap-to' grid lines can be useful, the positioning is imprecise. The user is the best person to decide where entities should be positioned.

Specification: ECLIPSE/WS/Tools/DE/FS Section 5.6
Source: Ray Wilson, Glasgow Office

User Requirements Guidelines

- Invent and adhere to a format
- Consistent use of language
  - “shall”, “should”
- text highlighting: bold, italics, color
- avoid jargon

System Requirements

- Expansion of user requirements
  - add detail

System Requirements

- Expansion of user requirements
  - add detail
- Starting point for system design
- Serve as contract

System Details

- Avoid internal description of system
  - e.g. system design or implementation

System Details

- Avoid internal description of system
  - e.g. system design or implementation
- difficult in practice

System Details

- Avoid internal description of system
  - e.g. system design or implementation
- difficult in practice
  - may need an initial architecture
  - a specific architecture may be enforced *in the requirements*

Structured Language

- Natural language may not be best choice
  - overflexible
  - ambiguous

Structured Language

- Natural language may not be best choice
  - overflexible
  - ambiguous
- Use **structured language**
- Helped by forms

Structured Language

Insulin Pump/Control Software/SRS/3.3.2

Function Compute insulin dose: Safe sugar level

Description Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.

Inputs Current sugar reading (r2), the previous two readings (r0 and r1)

Source Current sugar reading from sensor. Other readings from memory.

Outputs CompDose – the dose in insulin to be delivered

Destination Main control loop

Action: CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.

Requires Two previous readings so that the rate of change of sugar level can be computed.

Pre-condition The insulin reservoir contains at least the maximum allowed single dose of insulin..

Post-condition r0 is replaced by r1 then r1 is replaced by r2

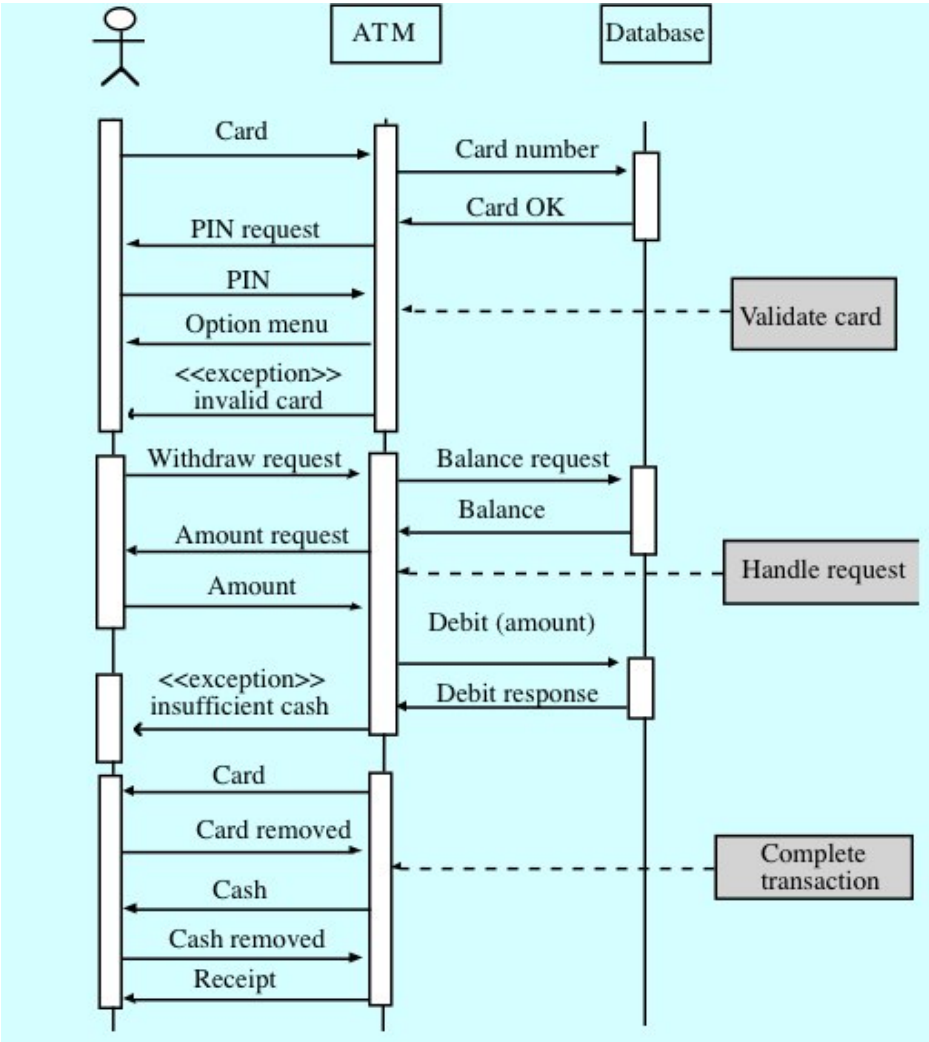
Side-effects None

Graphical Models

- Very useful for showing:
  - *state changes*
  - *sequence of actions*

Graphical Models

- Very useful for showing:
  - *state changes*
  - *sequence of actions*



Interface Specifications

- System must work with existing systems
  - specify existing systems' interfaces

Interface Specifications

- System must work with existing systems
  - specify existing systems' interfaces
- Types of interfaces:

Interface Specifications

- System must work with existing systems
  - specify existing systems' interfaces
- Types of interfaces:
  - *procedural interfaces*, also called APIs
  - *data structures* exchanged
  - *data representation* (bit-byte level)

Procedural Interface in Java

- Can specify in Java using `interface`

Procedural Interface in Java

- Can specify in Java using `interface`

```
interface PrintServer {

// defines an abstract printer server
// requires:          interface Printer, interface PrintDoc
// provides: initialize, print, displayPrintQueue, cancelPrintJob, switchPrinter

        void initialize ( Printer p ) ;
        void print ( Printer p, PrintDoc d ) ;
        void displayPrintQueue ( Printer p ) ;
        void cancelPrintJob (Printer p, PrintDoc d) ;
        void switchPrinter (Printer p1, Printer p2, PrintDoc d) ;
} //PrintServer
```

Procedural Interface in Java

- Can specify in Java using `interface`

```
interface PrintServer {

// defines an abstract printer server
// requires:          interface Printer, interface PrintDoc
// provides: initialize, print, displayPrintQueue, cancelPrintJob, switchPrinter

        void initialize ( Printer p ) ;
        void print ( Printer p, PrintDoc d ) ;
        void displayPrintQueue ( Printer p ) ;
        void cancelPrintJob (Printer p, PrintDoc d) ;
        void switchPrinter (Printer p1, Printer p2, PrintDoc d) ;
} //PrintServer
```

- may need more description

Outline

Software Requirements Document

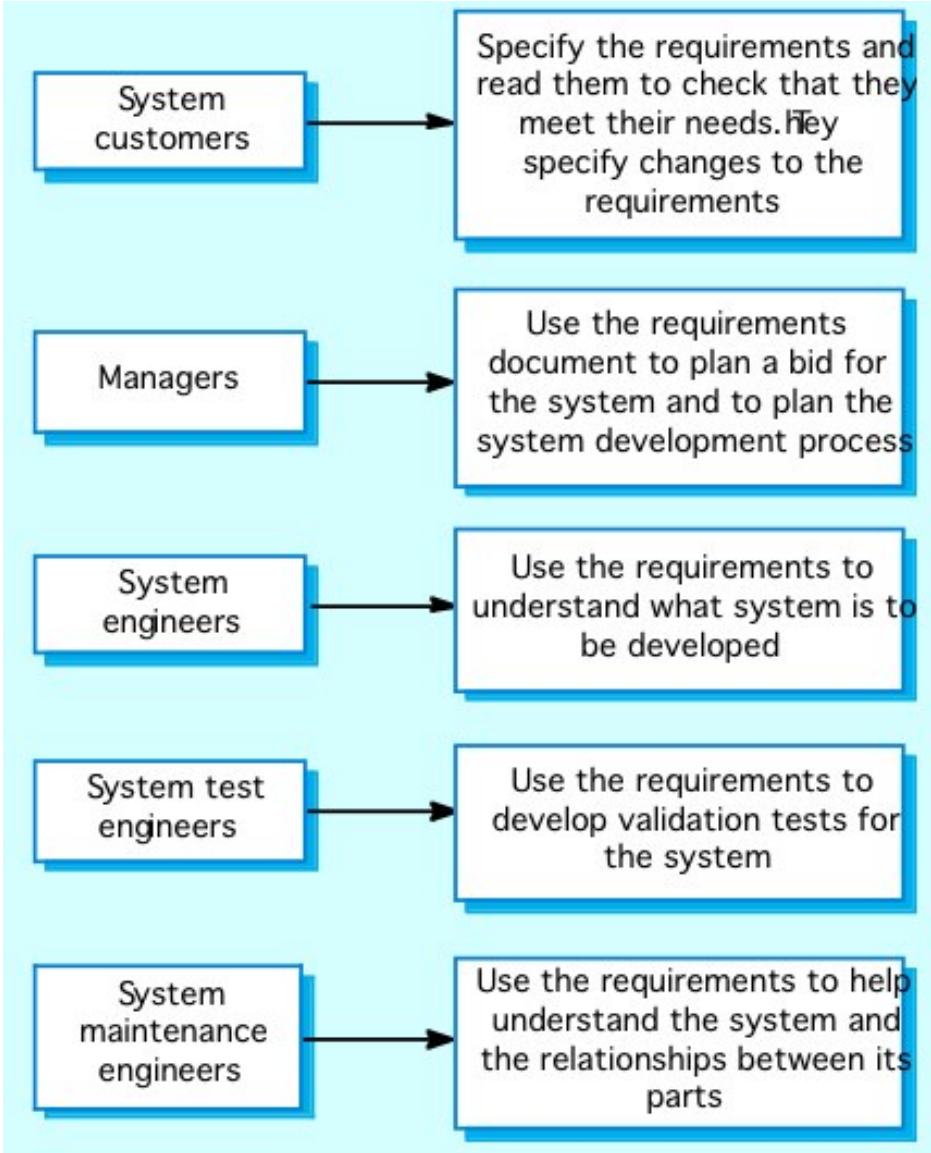
- Also *Software Requirements Specification* (SRS)
- Official statement
- Both **system** & **user**
  - sometimes combined

Users of SRD

- Wide variety

Users of SRD

- Wide variety



Parts of SRD

- Introduction
  - Rationale, system function, expected use

Parts of SRD

- Introduction
  - Rationale, system function, expected use
- Glossary
  - technical terms used

Parts of SRD

- Introduction
  - Rationale, system function, expected use
- Glossary
  - technical terms used
- User requirements definition
  - function / non-functional
  - use natural language, diagrams

Parts of SRD

- Introduction
  - Rationale, system function, expected use
- Glossary
  - technical terms used
- User requirements definition
  - function / non-functional
  - use natural language, diagrams
- System architecture
  - high-level overview

Parts of SRD

- Introduction
  - Rationale, system function, expected use
- Glossary
  - technical terms used
- User requirements definition
  - function / non-functional
  - use natural language, diagrams
- System architecture
  - high-level overview
- System requirements specification
  - detail the function / non-functional requirements

Parts of SRD

- Introduction
  - Rationale, system function, expected use
- Glossary
  - technical terms used
- User requirements definition
  - function / non-functional
  - use natural language, diagrams
- System architecture
  - high-level overview
- System requirements specification
  - detail the function / non-functional requirements
- System evolution
  - assumptions made and changes for: hardware, data, user needs, etc.

Parts of SRD

- Introduction
  - Rationale, system function, expected use
- Glossary
  - technical terms used
- User requirements definition
  - function / non-functional
  - use natural language, diagrams
- System architecture
  - high-level overview
- System requirements specification
  - detail the function / non-functional requirements
- System evolution
  - assumptions made and changes for: hardware, data, user needs, etc.
- Appendices
  - if any
  - hardware / DB / data descriptions

Other SRD Guidelines

- In PDF or HTML
- properly sectioned

Other SRD Guidelines

- In PDF or HTML
- properly sectioned
- **comprehensive** or
- **agile**
  - *keep up-to-date*