

Chapter 7

Chapter 7 Requirements Definition and Specification

Learning Objective

... Techniques for defining and specifying software system requirements.

Frederick T Sheldon
Assistant Professor of Computer Science
Washington State University

Objectives

- ◆ To illustrate a forms-based method of writing requirements definition
- ◆ To describe ways of writing precise specifications
- ◆ To explain the importance of non-functional requirements
- ◆ To describe different types of non-functional requirement and how these can be specified

Topics covered

- ◆ Requirements definition
- ◆ Requirements specification
- ◆ Non-functional requirements

Definition and specification

- ◆ Requirements definition
 - Customer-oriented descriptions of the system's functions and constraints on its operation
- ◆ Requirements specification
 - Precise and detailed descriptions of the system's functionality and constraints. Intended to communicate what is required to system developers and serve as the basis of a contract for the system development

Requirements definition

- ◆ Should specify external behavior of the system so the requirements should not be defined using a computational model
- ◆ Includes functional and non-functional requirements
 - Functional requirements are statements of the services that the system should provide
 - Non-functional requirements are constraints on the services and functions offered by the system

Writing requirements definitions

- ◆ 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 confusion. Functional and non-functional requirements tend to be mixed-up
 - Requirements amalgamation. Several different requirements may be expressed together

APSE database requirement

4.A.5 The database shall support the generation and control of configuration objects; that is, objects which are themselves groupings of other objects in the database. The configuration control facilities shall allow access to the objects in a version group by the use of an incomplete name.

Editor grid requirement

2.6 Grid facilities To assist in the positioning of entities on a diagram, the user may turn on a grid in either centimeters 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 centimeters 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.

Defining requirements

- ◆ Editor requirement mixes up functional and non-functional requirements and is incomplete
- ◆ Easy to criticize but hard to write good requirements definitions
- ◆ Use of a standard format with pre-defined fields to be filled means that information is less likely to be missed out

Editor grid definition

2.6 Grid facilities

2.6.1 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.

2.6.2 When used in 'reduce-to-fit' mode (see 2.1), the number of units separating grid lines must be increased.

Rationale: If line spacing is not increased, the background will be very cluttered with grid lines.

Specification: ECLIPSE/WS/Tools/DE/FS Section 5.6

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

Node creation definition

3.5.1 Adding nodes to a design

3.5.1.1 The editor shall provide a facility where users can add nodes of a specified type to a design. Nodes are selected (see 3.4) when they are added to the design.

3.5.1.2 The sequence of actions to add a node should be as follows:

1. The user should select the type of node to be added.
2. The user moves the cursor to the approximate node position in the diagram and indicates that the node symbol should be added at that point.
3. The symbol may then be dragged to its final position.

Rationale: The user is the best person to decide where to position a node on the diagram. This approach gives the user direct control over node type selection and positioning.

Specification: ECLIPSE/WS/Tools/DE/FS. Section 3.5.1

Requirements specification

- ◆ The specifications adds detail to the requirements definition. It should be consistent with it.
- ◆ Usually presented with system models which are developed during the requirements analysis. These models may define part of the system to be developed
- ◆ Often written in natural language but this can be problematical

Problems with natural language

- ◆ Natural language relies on the specification readers and writers using the same words for the same concept
- ◆ A natural language specification is over-flexible and subject to different interpretations
- ◆ Requirements are not partitioned by language structures

Natural language alternatives

- ◆ Structured natural language
- ◆ Design description languages
- ◆ Requirements specification languages
- ◆ Graphical notations
- ◆ Mathematical specifications

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

Structured language specifications

- ◆ A limited form of natural language may be used to express requirements
- ◆ This removes some of the problems resulting from ambiguity and flexibility and imposes a degree of uniformity on a specification
- ◆ Often best supported using a forms-based approach

Form-based specifications

- ◆ Definition of the function or entity
- ◆ Description of inputs and where they come from
- ◆ Description of outputs and where they go to
- ◆ Indication of other entities required
- ◆ Pre and post conditions (if appropriate)
- ◆ The side effects (if any)

Form-based node specification

ECLIPSE/Workstation/Tools/DE/FS/3.5.1

Function Add node

Description Adds a node to an existing design. The user selects the type of node, and its position. When added to the design, the node becomes the current selection. The user chooses the node position by moving the cursor to the area where the node is added.

Inputs Node type, Node position, Design identifier.

Source Node type and Node position are input by the user, Design identifier from the database.

Outputs Design identifier.

Destination The design database. The design is committed to the database on completion of the operation.

Requires Design graph rooted at input design identifier.

Pre-condition The design is open and displayed on the user's screen.

Post-condition The design is unchanged apart from the addition of a node of the specified type at the given position.

Side-effects None

Definition: ECLIPSE/Workstation/Tools/DE/RD/3.5.1

PDL-based requirements definition

- ◆ Requirements may be defined operationally using a language like a programming language but with more flexibility of expression
- ◆ Most appropriate in two situations
 - Where an operation is specified as a sequence of actions and the order is important
 - When hardware and software interfaces have to be specified
- ◆ Disadvantages are
 - The PDL may not be sufficiently expressive to define domain concepts
 - The specification will be taken as a design rather than a specification

PDL description of an ATM

Form-based node specification

```
procedure ATM is -- ATM/RS/CONT/1 Control specification for an ATM
  PIN: Pin_no ;
  Acc_no: Account_number ; Balance: Amount ;
  Service: Available_services ; Valid_card, Valid_PIN: Boolean ;
begin
loop
  Get_card ( Acc_no, PIN, Valid_card ) ;
  if Valid_card then
    Validate_PIN (PIN, Valid_PIN) ;
    if Valid_PIN then
      Get_account (Acc_no, Balance) ;
      Get_service (Service) ;
      while a service is selected loop
        Deliver_selected_service ;
        Get_service (Service) ;
      end loop ;
      Return_card ;
    end if ;
  end if ;
end loop ;
end ATM ;
```

Interface specification

- ◆ Almost all software systems operate in an environment where there are other systems. They may be interfaces to these systems in different ways
- ◆ Three types of interface may have to be defined in a requirements specification
 - Procedural interfaces. Sub-systems offer a range of services
 - Data interfaces. Data structures are exchanged
 - Representation interfaces. Specific data representation patterns may have to be used

Procedural interface example

```
package Print_server is
  procedure Initialize (P: PRINTER) ;
  procedure Print (P: PRINTER ; F: PRINT_FILE) ;
  procedure Display_print_queue (P: PRINTER) ;
  procedure Cancel_print_job (P: PRINTER; N: PRINT_ID) ;
  procedure Switch_printer (P1, P2: PRINTER; N: PRINT_ID) ;
end Print_server ;
```

Data interface example

```
type MESSAGE is record
  Sender : SYSTEM_ID;
  Receiver : SYSTEM_ID;
  Dispatch_time : DATE;
  Length : MESSAGE_LENGTH;
  Terminator : CHARACTER;
  Message : TEXT;
end record;
type SYSTEM_ID is range 20_000..30_000;
type YEAR_TYPE is range 1980..2080;
type DATE is record
  Seconds : NATURAL;
  Year : YEAR_TYPE;
end record;
type MESSAGE_LENGTH is range 0..10_000;
type TEXT is array (MESSAGE_LENGTH) of CHARACTER;
```

Size representation

```
for SYSTEM_ID'SIZE use 2*BYTE;
for YEAR_TYPE'SIZE use 2*BYTE;
for MESSAGE_LENGTH'SIZE use 2*BYTE;
```

Representation interface example

```
type STATE is (Halted, Waiting, Ready, Running);
for STATE use (Halted => 1, Waiting => 4, Ready => 16,
              Running => 256);
```

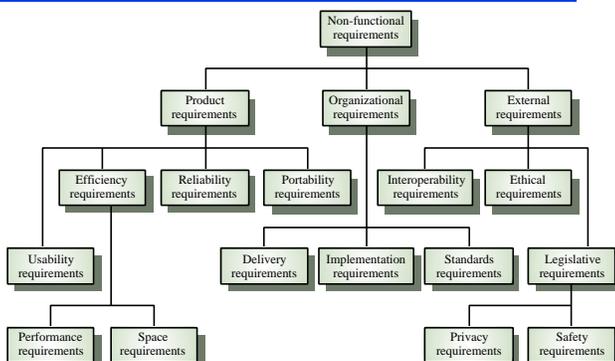
Non-functional requirements

- ◆ Define system properties and constraints e.g. 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 classifications

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

Non-functional requirement types



Non-functional requirements examples

- ◆ **Product requirement**
 - 4.C.8 It shall be possible for all necessary communication between the APSE and the user to be expressed in the standard Ada character set.
- ◆ **Organizational requirement**
 - 9.3.2 The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95.
- ◆ **External requirement**
 - 7.6.5 The system shall provide facilities that allow any user to check if personal data is maintained on the system. A procedure must be defined and supported in the software that will allow users to inspect personal data and to correct any errors in that data.

Requirements verifiability

- ◆ Requirements should be written so that they can be objectively verified
- ◆ The problem with this requirement is its use of vague terms such as ‘errors shall be minimized’
 - The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized.
- ◆ The error rate should be been quantified
 - Experienced controllers should be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users should not exceed two per day.

Requirements measures

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

Requirements separation

- ◆ Functional and non-functional requirements should, in principle, be distinguished in a requirements specification
- ◆ However, this is difficult as requirements may be expressed as whole system requirements rather than constraints on individual functions
- ◆ It is sometimes difficult to decide if a requirement is functional or a non-functional
 - For example, requirements for safety are concerned with non-functional properties but may require functions to be added to the system

System-level requirements

- ◆ Some requirements place constraints on the system as a whole rather than specific system functions
- ◆ Example
 - The time required for training a system operator to be proficient in the use of the system must not exceed 2 working days.
- ◆ These may be emergent requirements (see Chapter 2) which cannot be derived from any single sub-set of the system requirements

Key points

- ◆ A requirements definition is used by customers and end-users. It must be written in a language which they can understand
- ◆ Rationale for a requirement should always be included in a requirements definition
- ◆ Requirements should be written so that they may be verified

Key points

- ◆ Requirements specifications are intended to precisely communicate the system functions and constraints. They may be written in some form of structured language
- ◆ Three classes of non-functional requirement are product requirements, process requirements and external requirements
- ◆ Natural language is normally used to write non-functional requirements because of their variability and complexity
