

COURSE NAME

SOFTWARE  
REQUIREMENT  
ENGINEERING

CSC 4126

(UNDERGRADUATE)

---

## CHAPTER 4

### DOCUMENTING REQUIREMENTS

---

Dr. Mohammad Mahmudul Hasan

Associate Professor

Department of Computer Science

American international university – Bangladesh

Email: [m.hasan@aiub.edu](mailto:m.hasan@aiub.edu)

Web: <http://cs.aiub.edu/profile/m.hasan>



Google Scholar



LinkedIn

ORCID

WEB OF SCIENCE™

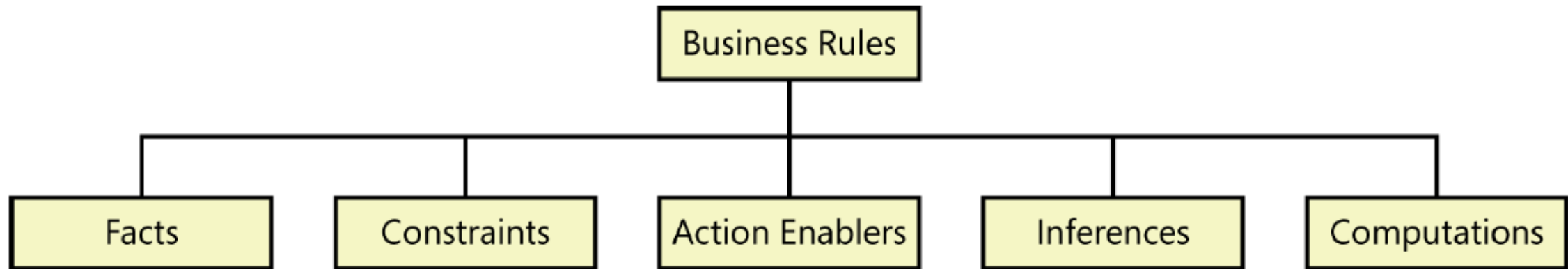


## INFLUENCE OF BUSINESS RULES

**TABLE 9-1** How business rules can influence various types of software requirements

Requirement type	Illustration of business rules' influence	Example
Business requirement	Government regulations can lead to necessary business objectives for a project.	<i>The Chemical Tracking System must enable compliance with all federal and state chemical usage and disposal reporting regulations within five months.</i>
User requirement	Privacy policies dictate which users can and cannot perform certain tasks with the system.	<i>Only laboratory managers are allowed to generate chemical exposure reports for anyone other than themselves.</i>
Functional requirement	Company policy is that all vendors must be registered and approved before an invoice will be paid.	<i>If an invoice is received from an unregistered vendor, the Supplier System shall email the vendor editable PDF versions of the supplier intake form and the W-9 form.</i>
Quality attribute	Regulations from government agencies, such as OSHA and EPA, can dictate safety requirements, which must be enforced through system functionality.	<i>The system must maintain safety training records, which it must check to ensure that users are properly trained before they can request a hazardous chemical.</i>

# A BUSINESS RULES TAXONOMY



**FIGURE 9-1** A simple business rule taxonomy.

## FACTS

- ❑ **Facts are simply statements that are true about the business at a specified point in time.**
- ❑ A fact describes associations or relationships between important business terms.
- ❑ Facts about data entities that are important to the system might appear in data models.
  
- ❑ Examples of facts include the following:
  - Every chemical container has a unique bar code identifier.
  - Every order has a shipping charge.
  - Non-refundable airline tickets incur a fee when the purchaser changes the route.
  - Books taller than 16 inches are shelved in the library's Oversize section.

## CONSTRAINTS

- ❑ **A constraint is a statement that restricts the actions that the system or its users are allowed to perform.**
- ❑ Someone describing a constraining business rule might say that certain actions must or must not or may not be performed, or that only certain people or roles can perform particular actions.
- ❑ Following are some examples of constraints with various origins.
  - **Organizational policies:** A loan applicant who is less than 18 years old must have a parent or a legal guardian as co-signer on the loan.
  - **Government regulations:** All software applications must comply with government regulations for usage by visually impaired persons.
  - **Industry standards:** Mortgage loan applicants must satisfy the Federal Housing Authority qualification standards.

## ACTION ENABLERS

- ❑ A rule that triggers some activity if specific conditions are true is an action enabler.
- ❑ Following are some examples of action-enabling business rules:
  - If the customer ordered a book by an author who has written multiple books, then offer the author's other books to the customer before completing the order.
  - After a customer places a book into the shopping cart, display related books that other customers also bought when they bought this one.

## INFERENCES

- ❑ Sometimes called inferred knowledge or a derived fact, an inference creates a new fact from other facts.
- ❑ Inferences are often written in the “if/then” pattern also found in action-enabling business rules, but the “then” clause of an inference simply provides a piece of knowledge, not an action to be taken.
- ❑ Some examples of inferences are:
  - If a payment is not received within 30 calendar days after it is due, then the account is inactive.
  - If the vendor cannot ship an ordered item within five days of receiving the order, then the item is considered back-ordered.

## COMPUTATIONS

- ❑ Computations transform existing data into new data by using specific mathematical formulas or algorithms.
- ❑ Many computations follow rules that are external to the enterprise, such as income tax withholding formulas.
- ❑ Following are a few examples of computational business rules written in text form:
  - The total price for an order is the sum of the price of the items ordered, less any volume discounts, plus state and county sales taxes for the location to which the order is being shipped, plus the shipping charge, plus an optional insurance charge.



# DOCUMENTING BUSINESS RULES

**TABLE 9-3** Some atomic business rules for a library

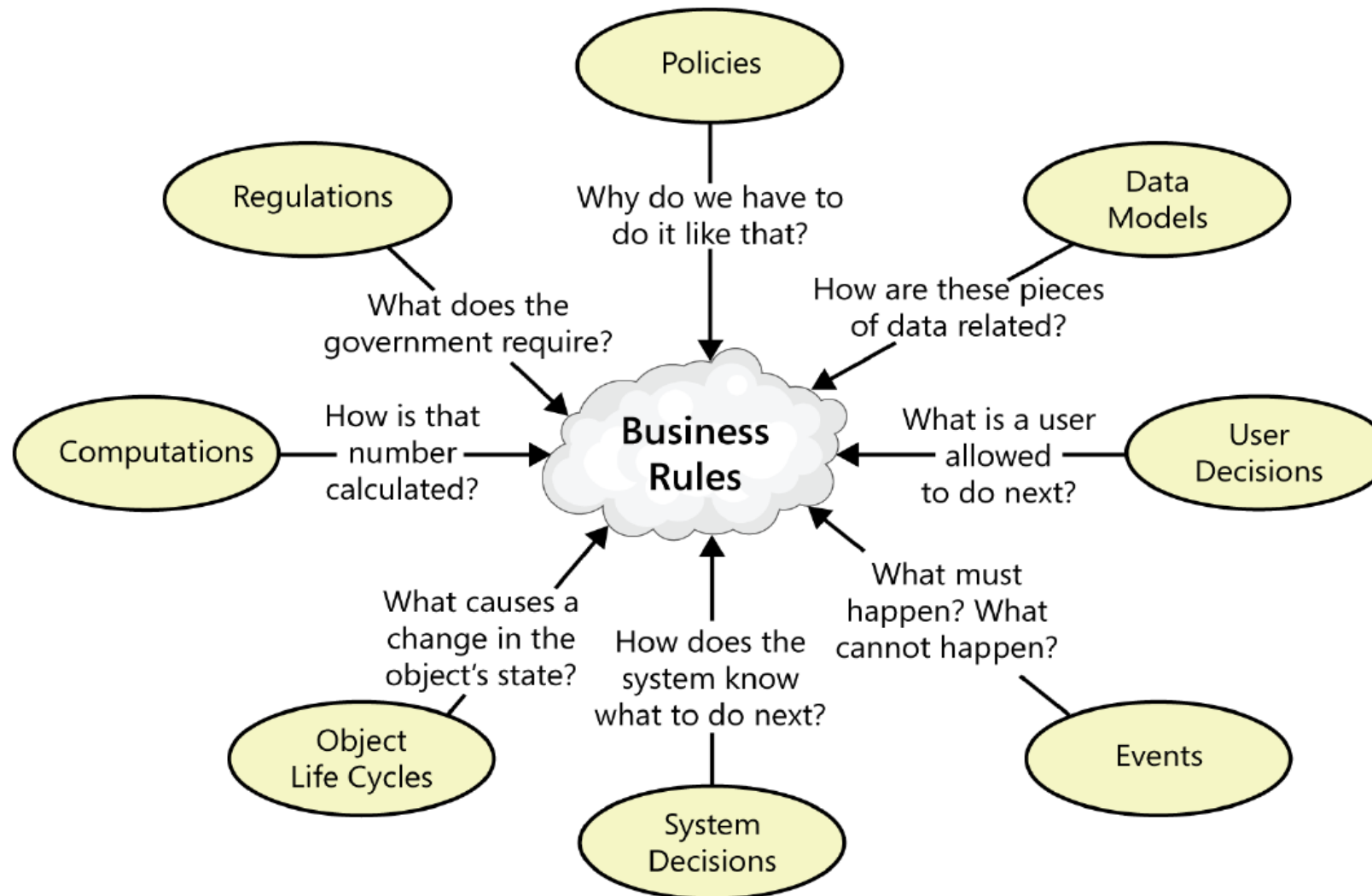
ID	Rule
Video.Media.Types	DVD discs and Blu-ray Discs are video items.
Video.Checkout.Duration	Video items may be checked out for one week at a time.
Renewal.Video.Times	Video items may be renewed up to two times.
Renewal.Video.Duration	Renewing a checked-out video item extends the due date by three days.
Renewal.HeldItem	A patron may not renew an item that another patron has on hold.

# DOCUMENTING BUSINESS RULES

TABLE 9-4 Some sample business rules catalog entries

ID	Rule definition	Type of rule	Static or dynamic	Source
ORDER-5	If the customer ordered a book by an author who has written multiple books, then offer the author’s other books to the customer before completing the order.	Action enabler	Static	Marketing policy XX
ACCESS-8	All website images must include alternative text to be used by electronic reading devices to meet accessibility requirements for visually impaired users.	Constraint	Static	ADA Standards for Accessible Design American with Disability Act.
DISCOUNT-13	A discount is calculated based on the size of the current order, as defined in Table BR-060.	Computation	Dynamic	Corporate pricing policy XX

## DISCOVERING BUSINESS RULES/PRACTICE



**FIGURE 9-3** Discovering business rules by asking questions from different perspectives.

# THE SOFTWARE REQUIREMENTS SPECIFICATION [SRS]

- ❑ Sometimes also called
  - Business requirements document (BRD)
  - Functional specification
  - Product specification
  - System specification
  - Requirements document

# SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

- ❑ The SRS states
  - the functions and capabilities that a software system must provide
  - its characteristics
  - the constraints that it must respect
- ❑ It should describe as completely as necessary the system's behaviours under various conditions, as well as desired system qualities such as performance, security, usability, etc.
- ❑ The SRS is the basis for subsequent project planning, design, and coding, as well as the foundation for system testing and user documentation.
- ❑ It should not contain design, construction, testing, or project management details other than known design and implementation constraints.

## AUDIENCE

- **Customers**, the marketing department, and sales staff need to know what product they can expect to be delivered.
- **Project managers** base their estimates of schedule, effort, and resources on the requirements.
- **Software development teams** need to know what to build.
- **Testers** use it to develop requirements-based tests, test plans, and test procedures.
- **Maintenance and support staff** use it to understand what each part of the product is supposed to do.
- **Documentation writers** base user manuals and help screens on the SRS and the user interface design.
- **Training personnel** use the SRS and user documentation to develop educational materials.
- **Legal staff** ensures that the requirements comply with applicable laws and regulations.
- **Subcontractors** base their work on—and can be legally held to—the specified requirements.

## READABILITY SUGGESTIONS

- Use an **appropriate template** to organize all the necessary information.
- **Label** and **style** sections, **subsections**, and individual requirements consistently.
- Use visual emphasis (**bold, underline, italics, colour, and fonts**) consistently and judiciously. Remember that colour highlighting might not be visible to people with colour blindness or when printed in grayscale.
- Create a **table of contents** to help readers find the information they need.
- **Number all figures and tables**, give them captions, and refer to them by number.
- If you are storing requirements in a document, define your word processor's **cross-reference** facility rather than hard-coded page or section numbers to refer **to other locations within a document**.

## READABILITY SUGGESTIONS

- If you are using documents, define **hyperlinks** to let the reader jump to related sections in the SRS or in other files.
- If you are storing requirements in a tool, use **links** to let the reader navigate to related information.
- Include **visual representations** of information when possible to **facilitate understanding**.
- Enlist a **skilled editor** to make sure the **document is coherent** and uses a **consistent vocabulary** and **layout**.



## LABELLING REQUIREMENTS

- ❑ Every requirement needs a **unique and persistent identifier**.
  - allows to refer to specific requirements in a **change request, modification history, cross-reference, or requirements traceability matrix**.
  - enables reusing the requirements in multiple projects.
- ❑ Uniquely identified requirements **facilitate collaboration between team members** when they're **discussing requirements**, as in a peer review meeting.
- ❑ Simple numbered or bulleted lists aren't adequate for these purposes.

## LABEL : SEQUENCE NUMBER

- The simplest approach gives every requirement a unique sequence number such as UC-9 or FR-26.
- Make it easy to retain a unique identifier if you move requirements around in a document. <Pros]
- A number is not reused if a requirement is deleted. <Cons]
- Doesn't provide any logical or hierarchical grouping of related requirements. <Cons]
- Doesn't imply any kind of ordering. <Cons]
- Gives no clue as to what each requirement is about. <Cons]

## LABEL : **HIERARCHICAL NUMBERING**

- ❑ Most commonly used convention.
- ❑ If the functional requirements appear in section 3.2 of your SRS, they will all have labels that begin with 3.2.
  - More digits indicate a more detailed, lower-level requirement.
  - 3.2.4.3 is a child requirement of 3.2.4.
- ❑ This method is simple, compact, and familiar. <Pros]
- ❑ The labels can grow to many digits in even a medium-sized SRS. <Cons]
- ❑ Numeric labels tell you nothing about the intent of a requirement. <Cons]
- ❑ Delete, insert, merge, or move whole sections, and a lot of labels change. <Cons]
  - These changes disrupt any references to those requirements elsewhere in the system.

## LABEL : HIERARCHICAL TEXTUAL TAGS

**Product:** Ordering products from the website

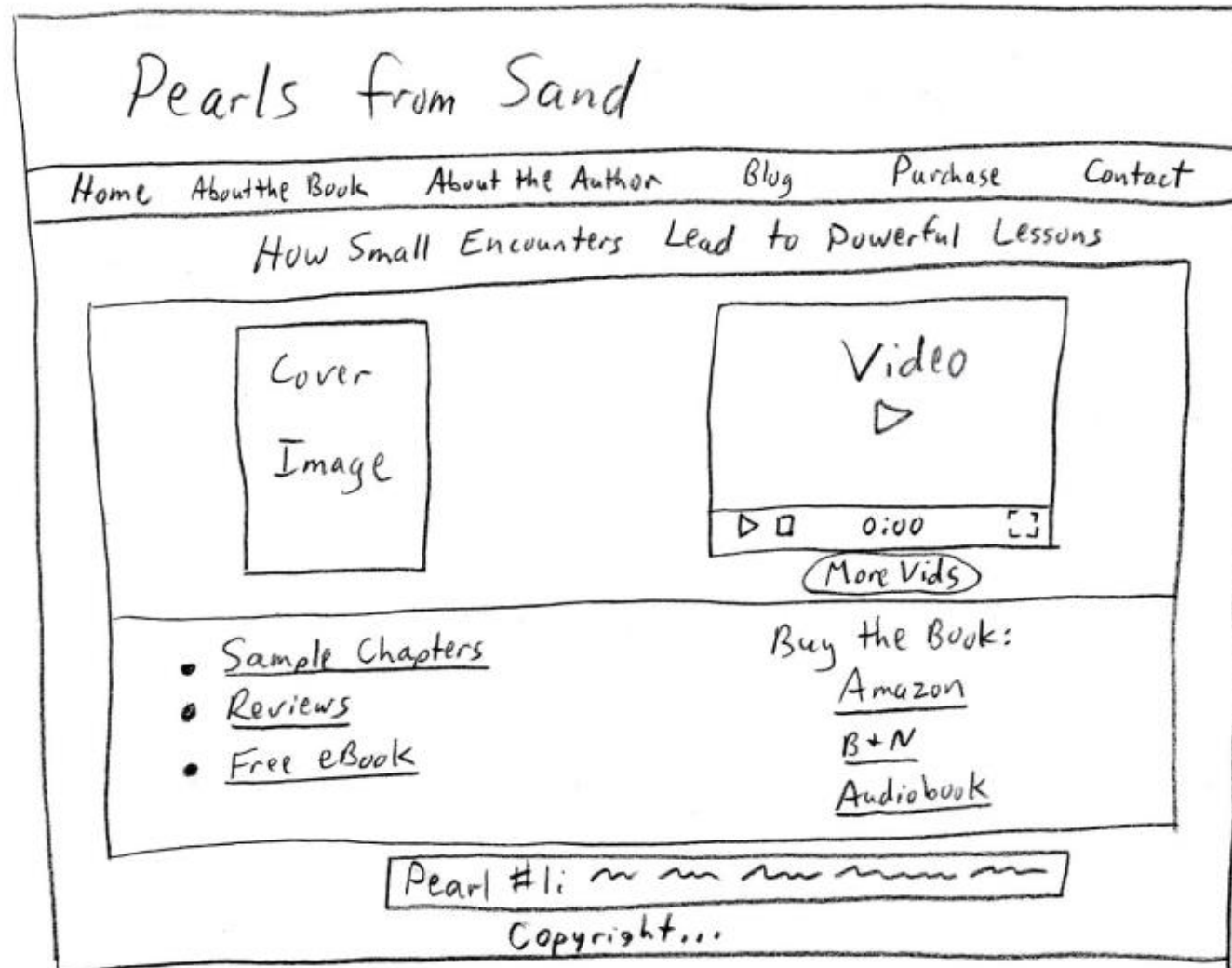
- |                  |   |
|------------------|---|
| <b>.Cart</b>     | The website shall use a shopping cart to contain products a Customer selects to purchase.   |
| <b>.Discount</b> | The shopping cart shall provide one discount code field. Each discount code provides either a specific discount percentage or a fixed dollar discount amount on specific items in the cart. |
| <b>.Error</b>    | If the Customer enters an invalid discount code, the website shall display an error message.  |
| <b>.Shipping</b> | The shopping cart shall add a shipping charge if a Customer orders a physical product that must be mailed.  |

- ❑ Can simplify the scheme by combining the hierarchical naming technique with a sequence number suffix for small sets of requirements: Product.Cart.01, Product.Cart.02, and so on. <Pros]
- ❑ Tags are longer and you do have to think of meaningful names for them. <Cons]

## DEALING WITH INCOMPLETENESS

- ❑ Sometimes you may **lack** a piece **of information** about a specific requirement.
  - Use the notation TBD (to be determined) to flag these knowledge gaps.
- ❑ Plan to **resolve all TBDs before implementing** a set of requirements.
- ❑ Any uncertainties that remain **increase the risk of a developer or a tester making errors** and having to **perform rework**.

## CONCEPTUAL USER INTERFACES (SKETCHES)



**FIGURE 10-1** Example of a user interface “sketch” suitable for inclusion in a requirements document.

## USER INTERFACES & THE SRS

### PROS:

- ❑ Exploring possible user interfaces (product's look and feel) with paper prototypes, working mock-ups, wireframes, or simulation tools makes the requirements tangible to both users and developers.

### CONS:

- ❑ Delaying baselining of the SRS until the UI design is complete can slow down development
- ❑ Including UI design in the requirements can result in the visual design driving the requirements, which often leads to functional gaps.
- ❑ Screen layouts don't replace written user and functional requirements. Don't expect developers to deduce the underlying functionality and data relationships from screen shots.

# SRS TEMPLATE

## I. Introduction

- 1.1 **Purpose** – identify the product whose requirements are specified in this document, and its readers
- 1.2 **Document Convention** – meaning of specific formatted text, style, highlighting, or notations
- 1.3 **Project Scope** – a short description of the software being specified
- 1.4 **References** – list any documents or other resources to which this SRS refers

## 2. Overall Description

- 2.1 **Productive perspective** – describe the product's context and origin
- 2.2 **User classes and characteristics** – various users classes who use this product
- 2.3 **Operating Environment** – describe the environment (h/w, OS, versions, organization structure, geographical location) in which the software will operate
- 2.4 **Design and implementation constraints** – certain PL must be used to maintain the software in future



# SRS TEMPLATE

## 3. System Features

3.1 Description of feature

3.2 Functional requirements

3.3 Cross-reference

## 4. External Interface Requirements

4.1 User interface

4.2 Software interface

4.3 Hardware interface

4.4 Communication interface

4.5 Cross-reference

# SRS TEMPLATE

## 5. Quality Attributes

5.1 Usability

5.2 Performance

5.4 [others]

5.5 Cross-references

## 6. Data Requirements

6.1 Logical data model – UML diagrams

6.2 Data dictionary - data type, length, format, and allowed values for data elements

## Appendix A: Glossary

## CHARACTERISTICS OF REQUIREMENT STATEMENTS

- Complete
- Correct
- Feasible
- Necessary
- Prioritized
- Unambiguous
- Verifiable

## GUIDELINES FOR WRITING REQUIREMENTS

- ❑ Two important goals of writing requirements are that:
  - Anyone who reads the requirement comes to the same interpretation as any other reader.
  - Each reader's interpretation matches what the author intended to communicate.
- ❑ System or user perspective
  - You can write functional requirements from the perspective of either something the system does or something the user can do.
  - State requirements in a consistent fashion, such as “The system shall” or “The user shall,” followed by an action verb, followed by the observable result.
  - Specify the trigger action or condition that causes the system to perform the specified behavior.
  - A generic template for a requirement written from the system's perspective is:  
[optional precondition] [optional trigger event] the system shall [expected system response].

# GUIDELINES FOR WRITING REQUIREMENTS

## □ Writing style

- Clarity and conciseness
- The keyword “shall”
- Active voice
- Individual requirements

# GUIDELINES FOR WRITING REQUIREMENTS

## Level of detail : Appropriate detail

### ☐ You should include more detail when:

- The work is being done for an external client
- Development or testing will be outsourced
- Project team members are geographically dispersed
- System testing will be based on requirements
- Accurate estimates are needed
- Requirements traceability is needed

### ☐ It's safe to include less detail when:

- The work is being done internally for your company
- Customers are extensively involved
- Developers have considerable domain experience
- Patterns are available, as when a previous application is being replaced
- A package solution will be used

## REPRESENTATION TECHNIQUES

- Some alternatives to the natural language requirements that we're used to are lists, tables, visual analysis models, charts, mathematical formulas, photographs, sound clips, and video clips.
- These won't suffice as substitutes for written requirements in many cases, but they serve as excellent supplemental information to enhance the reader's understanding.

## AVOIDING AMBIGUITY

- **Fuzzy words** – vague words
- **The A/B construct** (related or synonymous, opposite terms)

- **Boundary values**

*Vacation requests of up to 5 days do not require approval. Vacation requests of 5 to 10 days require supervisor approval. Vacation requests of 10 days or longer require management approval.*

This phrasing makes it unclear as to which category vacation requests of exactly 5 days and exactly 10 days belong.

- **Negative requirements**

***Prevent** the user from activating the contract if the contract is **not** in balance.*

Consider rephrasing this double negative (“prevent” and “not in balance”) as a positive statement:

*The system shall allow the user to activate the contract only if the contract is in balance.*



## AVOIDING INCOMPLETENESS

- **Complex logic** – possible alternative options are missing  
*If the Premium plan is not selected AND proof of insurance is not provided, the customer should automatically default in the Basic plan. [Premium plan is selected, and proof of insurance is not provided]*
- **Symmetry** – “save the information in the document” but retrieval part is missing if didn’t save it
- **Missing exceptions**

*If the user is working in an existing file and chooses to save the file, the system shall save it with the same name*

This requirement alone does not indicate what the system should do if it’s unable to save the file with the same name. An appropriate second requirement to go with the first might be:

*If the system is unable to save a file using a specific name, the system shall give the user the option to save it with a different name or to cancel the save operation.*

## MODELING THE REQUIREMENTS

- Data flow diagram (DFD)
- Process flow diagram such as Swimlane diagram
- State-transition diagram (STD) and state tables
- Dialog map
- Decision table and decision tree
- Event-response table
- Feature tree
- Use case diagram
- Activity diagram
- Entity-relationship diagram (ERD), Class Diagram

## DATA DICTIONARY

- A set of information describing the contents, format, and structure of a database and the relationship between its elements, used to control access to and manipulation of the database


























Entity	Attribute	Type/Size	Validation	Key
Pupil	PupilID	Number (5)	10000-99999	Primary
Pupil	Forename	Text (10)	Required	
Pupil	Surname	Text (15)	Required	
Pupil	DOB	Date (8)	Valid Date	
Teacher	TeacherID	Number (3)	100-999	Primary
Teacher	Name	Text (15)	Required	
Teacher	Room	Text (3)		
Teacher	Subject	Text (10)		

## SPECIFYING REPORTS

- A **dashboard** is a screen display or printed report that uses multiple textual and/or graphical representations of data to provide a consolidated, multidimensional view of what is going on in an organization or a process.

### Scrum Task Board Template

Company name

Stories	To Do		In Progress	Testing	Done
 This is a sample text. Replace it with your own text.	 This is a sample text. Replace it with your own text.	 This is a sample text. Replace it with your own text.	 This is a sample text.	 This is a sample text.	 This is a sample text. Replace it with your own text.
	 This is a sample text. Replace it with your own text.	 This is a sample text. Replace it with your own text.	 This is a sample text.	 This is a sample text.	
	 This is a sample text. Replace it with your own text.	 This is a sample text. Replace it with your own text.	 This is a sample text.	 This is a sample text.	 This is a sample text. Replace it with your own text.
 This is a sample text. Replace it with your own text.	 This is a sample text.	 This is a sample text.	 This is a sample text.	 This is a sample text.	 This is a sample text. Replace it with your own text.
	 This is a sample text.	 This is a sample text.	 This is a sample text. Replace it with your own.	 This is a sample text.	

## REFERENCES

- Wiegers, K., & Beatty, J. (2013). *Software requirements*. Pearson Education.
- <http://www.cs.ccsu.edu/~stan/classes/CS530/notes14/04-Requirements.html>