

Software Requirements Specification

“Writing down the requirements”

Using Natural Language

- ⌘ More than 50% of the content of the specifications are usually written using plain English, Bahasa, Chinese, etc.
- ⌘ Easy to produce
- ⌘ Easy for customer to understand
- ⌘ Can be improved by using structured form, using fields and description placed in fields

Lecture Objectives

- ⌘ To describe different methods of specifying the software requirements
- ⌘ To illustrate the use of technical methods for specifying certain types of requirements
- ⌘ To describe the properties of good software requirement specifications

Examples of Specifications

- ⌘ *The printout of outstanding creditors are done monthly at the end of each month.*
- ⌘ *Purchase orders are automatically generated when the item quantity reaches below the reorder level.*
- ⌘ *Usually the report is produced based on the code given.*

Software Requirements Specification

- ⌘ From our understanding of the problem, we now write down what the customer wants from the software
- ⌘ Documentation of what the software is supposed to be
- ⌘ Many companies regard it as ‘contract’ between customer and developer
- ⌘ The basis for validation and verification

Problems of Natural Language

- ⌘ Lack of clarity
 - ☒ Words such as “some”, “usually”, “probably”
- ⌘ Ambiguity
 - ☒ Some words have more than one meaning
- ⌘ Context dependency
 - ☒ Same word in different sentences have different meaning
 - ☒ Depends on whole paragraph/page/document

Using Diagrams

Graphical representation of the analysis can present the information better using:

- ⌘ Entity-Relationship Diagrams
- ⌘ Data Flow Diagrams
- ⌘ State Transition Diagrams
 - ☐ event table, action table
- ⌘ Decision Tables
- ⌘ Decision Trees

Example of Decision Table

Conditions/ Actions	R u l e s					
	1	2	3	4	5	6
Employee Type	S	H	S	H	S	H
Hours worked	<40	<40	40	40	>40	>40
Pay base salary	X		X		X	
Calculate Hourly wage		X		X		X
Calculate Overtime						X
Produce Absence Report		X				

Decision Tables

- ⌘ Representation of logic that is part of the processing
- ⌘ Based on a set of conditions, different actions will be performed
- ⌘ Can be simplified by removing impossible actions

Constructing Decision Table

- ⌘ Name the conditions and the values each condition can assume
- ⌘ Name all possible actions that can occur
- ⌘ List all possible rules
- ⌘ Define the actions for each rule
- ⌘ Simplify the decision table

Structure of Decision Table

	Decision rules			
	Rule 1	Rule 2	Rule 3	Rule 4
(condition stub)	(Condition entries)			
(action stub)	(Action entries)			

Simplified Decision Table

Conditions/ Actions	Rules			
	1	2	3	4
Employee Type	S	H	H	H
Hours worked	-	<40	40	>40
Pay base salary	X			
Calculate Hourly wage		X	X	X
Calculate Overtime				X
Produce Absence Report		X		



Decision Trees

- ⌘ Graphical technique representing decisions using a series of nodes and branches
- ⌘ Each node is a decision point - a choice has to be made
- ⌘ Each branch has a corresponding value to the decision choice
- ⌘ Subsequent action is the result

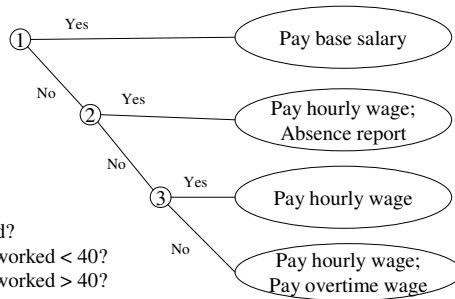


Mathematical Methods (Continued)

- ⌘ Recurrence Relations
 - ⊠ Consist of initial part and one or more recursive parts
 - ⊠ Example: Fibonacci sequence
 - ⊠ $F(0) = 1$
 - ⊠ $F(1) = 1$
 - ⊠ $F(N) = F(N-1) + F(N-2) \quad N \geq 2$
 - ⊠ Useful for generation of repeated items, e.g. account number



Example of Decision Tree



- Legend:
- 1) Salaried?
 - 2) Hours worked < 40?
 - 3) Hours worked > 40?



Advanced Mathematical Methods

- ⌘ Axiomatic Definition
 - ⊠ specify basic system properties (axioms) and how the system generate new properties
 - ⊠ Example:
 - ⊠ $REPLACE(stk,itm) = \text{if } EMPTY(stk) \text{ then error else } (PUSH(POP(stk),itm))$
- ⌘ Formal specifications
 - ⊠ Use of mathematically based techniques
 - ⊠ Sets, Operators, Sequences, Logic Operators



Mathematical Methods

- ⌘ Implicit Equations
 - ⊠ Specific equations relevant to software
 - ⊠ Includes other equation details e.g. data type, matrix size, algorithm
- ⌘ Regular Expressions
 - ⊠ To specify syntactic structure of string codes
 - ⊠ Example: $\langle id \rangle ::= \langle yr \rangle \langle course \rangle \langle number \rangle$
 - ⊠ $\langle number \rangle ::= \langle digit \rangle \langle digit \rangle \langle digit \rangle \langle digit \rangle$
 - ⊠ $\langle digit \rangle ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$



Good Specifications

- ⌘ Correct
- ⌘ Complete
- ⌘ Consistent
- ⌘ Unambiguous
- ⌘ Functional
- ⌘ Verifiable
- ⌘ Traceable
- ⌘ Easily changed



Traceability Methods

- ⌘ All requirements should be assigned a unique number
- ⌘ Requirements should explicitly identify related requirements by referring to their number
- ⌘ Each requirement should contain a cross-reference matrix showing related requirements



References

- ⌘ "Software Engineering: A Practitioner's Approach" 5th Ed. by Roger S. Pressman, Mc-Graw-Hill, 2001
- ⌘ "Software Engineering" by Ian Sommerville, Addison-Wesley, 2001
- ⌘ "Modern Systems Analysis and Design" by Jeffrey A. Hoffer, Joey F. George & Joseph S. Valacich, Benjamin/Cummings, 1996