

FUNDAMENTALS OF SOFTWARE ENGINEERING - CHAPTER 5

Timothy Hnat

Department of Computer Science
University of Virginia

June 7, 2007



INTRODUCTION

- What is a specification?



INTRODUCTION

- What is a specification?
- It is a precise statement of the requirements that the system must satisfy
- Software Engineering - An agreement between a producer of a service and a consumer of the service



INTRODUCTION

- What is a specification?
- It is a precise statement of the requirements that the system must satisfy
- Software Engineering - An agreement between a producer of a service and a consumer of the service

SPECIFICATIONS

- A specification at some level states the requirements for the implementation at a lower level.
- Specification is a critical part of the design process



USES OF SPECIFICATIONS

- A statement of user requirements
 - Not clearly understood by the developer
 - Initial description produced by an inexperienced user will lack a precise formulation
 - Major failures occur because of misunderstandings between the producer and the user
 - An ability to verify the specifications is necessary



USES OF SPECIFICATIONS

- A statement of user requirements
 - Not clearly understood by the developer
 - Initial description produced by an inexperienced user will lack a precise formulation
 - Major failures occur because of misunderstandings between the producer and the user
 - An ability to verify the specifications is necessary
- A statement of the interface between the machine and the controlled environment
 - Computer take inputs and provide outputs. If this specification is not correct, very bad things may happen



USES OF SPECIFICATIONS

- A statement of user requirements
 - Not clearly understood by the developer
 - Initial description produced by an inexperienced user will lack a precise formulation
 - Major failures occur because of misunderstandings between the producer and the user
 - An ability to verify the specifications is necessary
- A statement of the interface between the machine and the controlled environment
 - Computer take inputs and provide outputs. If this specification is not correct, very bad things may happen
- A statement of the requirements for the implementation
 - Specification provide the reference point for implementation
 - Ultimate goal of implementation is to meet the specifications



USES OF SPECIFICATIONS

- A statement of user requirements
 - Not clearly understood by the developer
 - Initial description produced by an inexperienced user will lack a precise formulation
 - Major failures occur because of misunderstandings between the producer and the user
 - An ability to verify the specifications is necessary
- A statement of the interface between the machine and the controlled environment
 - Computer take inputs and provide outputs. If this specification is not correct, very bad things may happen
- A statement of the requirements for the implementation
 - Specification provide the reference point for implementation
 - Ultimate goal of implementation is to meet the specifications
- A reference point during production maintenance
 - Corrective maintenance usually updates the implementation
 - Adaptive maintenance changes the requirements, specifications, and implementation



SPECIFICATION QUALITIES

- Must be clear, unambiguous, and understandable



SPECIFICATION QUALITIES

- Must be clear, unambiguous, and understandable
- Consistent



SPECIFICATION QUALITIES

- Must be clear, unambiguous, and understandable
- Consistent
- Complete
 - Internally complete - defines any new concept or terminology that is used
 - External completeness - w/respect to requirements
 - Unrealistic to ask for complete specifications in the strict sense



SPECIFICATION QUALITIES

- Must be clear, unambiguous, and understandable
- Consistent
- Complete
 - Internally complete - defines any new concept or terminology that is used
 - External completeness - w/respect to requirements
 - Unrealistic to ask for complete specifications in the strict sense

INCREMENTALITY PRINCIPLE

One starts with a basic specification document and expands it through several steps



CLASSIFICATION OF SPECIFICATION STYLES

- Formal/Informal styles
 - Informal - Written in a natural language w/figures, tables, etc...
 - Formal - Written with a formal notation (OCL)
 - UML is a semi-formal specification



CLASSIFICATION OF SPECIFICATION STYLES

- Formal/Informal styles
 - Informal - Written in a natural language w/figures, tables, etc...
 - Formal - Written with a formal notation (OCL)
 - UML is a semi-formal specification
- Operation/Descriptive styles
 - Operational - Specifies the intended behavior, usually through models
 - Descriptive - Specifies the desired properties in a declarative fashion



CLASSIFICATION OF SPECIFICATION STYLES

- Formal/Informal styles
 - Informal - Written in a natural language w/figures, tables, etc...
 - Formal - Written with a formal notation (OCL)
 - UML is a semi-formal specification
- Operation/Descriptive styles
 - Operational - Specifies the intended behavior, usually through models
 - Descriptive - Specifies the desired properties in a declarative fashion

TRADEOFFS

- Descriptive specifications are more abstract (No bias towards implementation)
- No style is correct for all situations
- No style can guarantee that a designer will come up with a good design



VERIFICATION OF SPECIFICATIONS

- Important to verify prior to starting an implementation to assess the correctness
- Method 1 - Observe the dynamic behavior of the specified system
- Method 2 - Analyze the properties of the system and deduce the specification
- Both techniques are more effective as a function of the formality of the specification



DATA FLOW DIAGRAMS: SPECIFYING FUNCTIONS OF INFORMATION SYSTEMS

- DFDs specify functions and their data flows.
- Success because of the graphical notation and ease of use



DATA FLOW DIAGRAMS: SPECIFYING FUNCTIONS OF INFORMATION SYSTEMS

- DFDs specify functions and their data flows.
- Success because of the graphical notation and ease of use

DFD elements

- Functions as bubbles
- Data flows as arrows
- Data stores as open boxes
- Input-output as special I/O boxes



DATA FLOW DIAGRAMS: SPECIFYING FUNCTIONS OF INFORMATION SYSTEMS II

Problems with DFDs

- Semantics of the symbols may not be precise
- Control aspects are not defined by the model (Synchronization problems?)



DATA FLOW DIAGRAMS: SPECIFYING FUNCTIONS OF INFORMATION SYSTEMS II

Problems with DFDs

- Semantics of the symbols may not be precise
- Control aspects are not defined by the model (Synchronization problems?)

Overcoming difficulties

- Use a complementary notation to describe those aspects of the system that are not captured adequately by DFDs
- Augmenting the DFD model in order to cope with aspects that are not captured by its traditional version (control flow arrows)
- Revising the traditional definition of a DFD to make it fully formal



DESCRIBING THE SYSTEM

- UML Diagrams for Specifying Behaviors
 - Blueprint for software systems
 - Useful in the requirements stage



DESCRIBING THE SYSTEM

- UML Diagrams for Specifying Behaviors
 - Blueprint for software systems
 - Useful in the requirements stage
- Finite State Machines: Describing Control Flow
 - Can make specifications more precise
 - Simple and widely used, may be a problem for intricate cases
 - Limited because it is a finite state machine
 - Difficult to combine machines into one in a clear way
 - FSMs can not represent asynchronous behavior



DESCRIBING THE SYSTEM

- Petri Nets: Specifying Asynchronous Systems
 - Apply to parallel activities
 - Solves the problem that FSMs left
 - A nondeterministic model
 - Can show/lead to deadlock situations



DESCRIBING THE SYSTEM

- Petri Nets: Specifying Asynchronous Systems
 - Apply to parallel activities
 - Solves the problem that FSMs left
 - A nondeterministic model
 - Can show/lead to deadlock situations
- Limitations and extensions of Petri nets
 - Tokens only represent a message, It would be nice to have multiple message types
 - Not possible to specify a selection policy for different transitions (proved mathematically)
 - Timing issues: not explicit in PN or computers
 - Limitations can be removed by careful additions to the PN



ENTITY-RELATIONSHIP DIAGRAMS

- Describes the relationships among the data of an information system.
- Motivation - Conceptual model of data suitable for specifying user views and logical requirements in information systems
- ER models are not standard (Problems?)
- What are graphical models good for?



LOGIC SPECIFICATIONS

- First-order Theory - basic logical statements
- Can take ER diagrams and form FOT statements



VERIFYING SPECIFICATIONS: A COMPARISON OF DESCRIPTIVE AND OPERATIONS STYLE

- Logical formalism describes the system by means of deductions
- Logical deduction can be applied to analyze system properties
- A PN interpreter can verify a system, but cannot provide the proof
- A PN interpreter cannot be used to guarantee is a network is deadlock free



VERIFYING SPECIFICATIONS: A COMPARISON OF DESCRIPTIVE AND OPERATIONS STYLE

- Logical formalism describes the system by means of deductions
- Logical deduction can be applied to analyze system properties
- A PN interpreter can verify a system, but cannot provide the proof
- A PN interpreter cannot be used to guarantee is a network is deadlock free

Why?



VERIFYING SPECIFICATIONS: A COMPARISON OF DESCRIPTIVE AND OPERATIONS STYLE

- Logical formalism describes the system by means of deductions
- Logical deduction can be applied to analyze system properties
- A PN interpreter can verify a system, but cannot provide the proof
- A PN interpreter cannot be used to guarantee is a network is deadlock free

Why?

- Proving theorems within FORs is undecidable.



ALGEBRAIC SPECIFICATIONS

- Heterogeneous algebra - collection of different sets on which several operations are defined
- Useful is specifying the semantics of modules such as abstract data types.



REQUIREMENTS FOR SPECIFICATION NOTATIONS

- For real world systems, very complex
- Separation of concerns - Functional vs Performance vs User vs ...
- Incrementality is more important for specifications than for implementation
- A specialized tool will help with specifications



BUILDING MODULAR SPECIFICATIONS

- Modularizing finite state machines: the case of Statecharts
- Modularizing logic specifications: the case of Z
- Specifications for the End User - Take formal specs and create a prose description



CONCLUSIONS

- A specification is a precise statement.
- Desirable qualities of specification
 - Formal and Informal
 - Operational and Descriptive
 - Data flow diagrams, finite state machines, statecharts, Petri Nets
 - Entityrelationship diagrams, class diagrams, logic, algebraic specifications
 - Data flow and entity-relationship diagrams are semiformal
 - Logic is a formal approach



CONCLUSIONS

- Modularity is the key to managing the writing of complex specifications
- No ideal specification style or language (consider a variety)
- Makes it more difficult to formulate a rigorous, unified semantics
- Specification in enhancing the interaction between the end user and the developer (Towards more accurate specifications)

