EDIDIONG IME- ESSIEN

CSC 308

17/SCI01/041

 Formal Methods

1. (ai)

 Formal Specifications

 Formal Proofs

 Model

 Checking Abstraction

(aii) . Systems are increasingly dependent on software components.

 . Complexity of systems with embedded software has increased rapidly.

 . Maintaining reliability in software-intensive systems is very difficult.

### b) Functional requirements: Any requirement which specifies what the system should do.

### Types of functional requirements

* Business Rules
* Transaction corrections, adjustments and cancellations
* Administrative functions
* Authentication
* Authorization levels
* Audit Tracking

### Non-functional requirements: Any requirement which specifies **how** the system performs a certain function.

### Types of functional requirements

* Serviceability
* Security
* Regulatory
* Manageability
* Environmental
* Data Integrity
* Usability
* Interoperability

##  Agile development methodology

Teams use the agile development methodology to minimize risk (such as bugs, cost overruns, and changing requirements) when adding new functionality. In all agile methods, teams develop the software in iterations that contain mini-increments of the new functionality. There are many different forms of the agile development method, including scrum, crystal, extreme programming (XP), and feature-driven development (FDD).

## DevOps deployment methodology

DevOps is not just a development methodology but also a set of practices that supports an organizational culture. DevOps deployment centers on organizational change that enhances collaboration between the departments responsible for different segments of the development life cycle, such as development, quality assurance, and operations.

## Waterfall development method

Many consider the waterfall method to be the most traditional software development method. The waterfall method is a rigid linear model that consists of sequential phases (requirements, design, implementation, verification, maintenance) focusing on distinct goals. Each phase must be 100% complete before the next phase can start. There’s usually no process for going back to modify the project or direction.

## Rapid application development

Rapid application development (RAD) is a condensed development process that produces a high-quality system with low investment costs. Scott Stiner, CEO and president of UM Technologies, [said in Forbes](http://www.forbes.com/sites/forbestechcouncil/2016/08/24/rapid-application-development-rad-a-smart-quick-and-valuable-process-for-software-developers/), “This RAD process allows our developers to quickly adjust to shifting requirements in a fast-paced and constantly changing market.” The ability to quickly adjust is what allows such a low investment cost.

2 (b) According to the spectrum of rigor. Formal methods ae techniques and tools based on mathematics and formal logic and to improve system reliability, design time and comprehensibility, they do so at the cost of an increased learning curve; the mathematical disciplines used to formally describe computational systems are outside the domain of a traditional engineering education.

 3 (ai) Propositional logic (also called sentential logic) is logic that includes sentence letters (A,B,C) and logical connectives, but not quantifiers. The semantics of propositional logic uses truth assignments to the letters to determine whether a compound propositional sentence is true.

Predicate logic is usually used as a synonym for first-order logic, but sometimes it is used to refer to other logics that have similar syntax. Syntactically, first-order logic has the same connectives as propositional logic, but it also has variables for individual objects, quantifiers, symbols for functions, and symbols for relations. The semantics include a domain of discourse for the variables and quantifiers to range over, along with interpretations of the relation and function symbols.

Constructing Propositions • Operations to construct compound propositions: • Conjunction ∧ AND

 • Disjunction ∨ OR

• Negation  ൓ NOT

 • Implication → IF‐THEN

Disjunction

The disjunction of propositions p and q is denoted by p ∨ q

• Its truth table is:

|  |  |  |
| --- | --- | --- |
|  p   | q | p ∨ q |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

Example:

• p: I am at home

 • q: It is raining

• p ∨ q: I am at home or it is raining

Ambiguity

• In natural languages “or” has two distinct meanings

• Inclusive Or:  • p ∨ q is true if either p or q or both are true

• Example:  • “Math 10a or Math 12 may be taken as a prerequisite for CS 6”

 • Meaning: take either one but may also take both

3b  “**Every man is mortal. Smith is a man**. **Therefore, Smith is mortal**”

M1=mortal

M2 = man

P1= every man is mortal

P2= smith is a man

∀ (m2): m1(p1,p2)

4a

O=object

M=monkey

-m=not monkey

There exist (o):m v -m

4b Types of formal Specification

* State-based specification[[3]](https://en.wikipedia.org/wiki/Formal_specification#cite_note-lamsweerde-roadmap-3)
	+ behavior based on system states
	+ series of sequential steps, (e.g. a financial transaction)
	+ languages such as [Z](https://en.wikipedia.org/wiki/Z_notation), [VDM](https://en.wikipedia.org/wiki/Vienna_Development_Method) or [B](https://en.wikipedia.org/wiki/B-Method) rely on this paradigm[[3]](https://en.wikipedia.org/wiki/Formal_specification#cite_note-lamsweerde-roadmap-3)
* Transition-based specification[[3]](https://en.wikipedia.org/wiki/Formal_specification#cite_note-lamsweerde-roadmap-3)
	+ behavior based on transitions from state-to-state of the system
	+ best used with a reactive system
	+ languages such as Statecharts, PROMELA, STeP-SPL, RSML or SCR rely on this paradigm[[3]](https://en.wikipedia.org/wiki/Formal_specification#cite_note-lamsweerde-roadmap-3)
* Functional specification[[3]](https://en.wikipedia.org/wiki/Formal_specification#cite_note-lamsweerde-roadmap-3)
	+ specify a system as a structure of mathematical functions
	+ OBJ, ASL, PLUSS, LARCH, HOL or PVS rely on this paradigm[[3]](https://en.wikipedia.org/wiki/Formal_specification#cite_note-lamsweerde-roadmap-3)

5 (ai) Well Formed Formula: is a finite sequence of symbols from a given alphabet that is part of a formal language

(ii) Quantifier:  a quantifier specifies the quantity of specimens in the domain of discourse that satisfy an open formula. The two most common formal quantifiers are "for each" ("∀"), and "there exists some" ("∃").

(iii) Predicate: is a statement that contains variables that may be true or false, depending on the values of the variables

(iv) Term: the subject or predicate of a categorical proposition (q.v.), or statement.

5b

6(ai) Software development strategies

* Testing of new systems (or regression testing of modified systems) to ensure that the goals of the functional design and technical design are met.
* Documentation of the system, both intrinsically for its future maintainers, and extrinsically for its future users. For large systems this step may involve end-user training as well.
* Maintenance of the application system over its typical five-year life cycle, employing the design document now recrafted as the Technical Specification or System Maintenance Document.

(b) Requirements documents are used to communicate the aims of a project in a clear, concise way to ensure all stakeholders are on the same page.

7a

|  |  |
| --- | --- |
| Preface | This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version. |
|  |  |
| Introduction | This should describe the need for the system.  It should briefly describe its functions and explain how it will work with other systems.  It should describe how the system fits into the overall business or strategic objectives of the organisation commissioning the software. |
|  |  |
| Glossary | This should define the technical terms used in the document.  You should not make assumptions about the experience or expertise of the reader. |
|  |  |
| User requirements definition | The services provided for the user and the non-functional system requirements should be descried in this section.  This description may use natural language, diagrams or other notations that are understandable by customers.  Product and process standards which must be followed should be specified. |
|  |  |
| System architecture | This chapter should present a high-level overview of the anticipated system architecture showing the distribution of functions across system modules.  Architectural components that are reused should be highlighted. |
|  |  |
| System requirements specification | This should describe the functional and non-functional requirements in more detail.  If necessary, further detail may also be added to the non-functional requirements, eg interfaces to other systems may be defined. |
|  |  |
| System models | This should set out one or more system models showing the relationships between the system components and the system and its environment.  These might be object models, data-flow models and semantic data models. |
|  |  |
| System evolution | This should describe the fundamental assumptions on which the system is based and anticipated changes due to hardware evolution, changing user needs, etc. |
|  |  |
| Appendices | These should provide detailed, specific information which is related to the application being developed.  Examples of appendices that may be included are hardware and database descriptions.  hardware requirements define the minimal and optimal configurations for the system.  Database requirements define the logical organisation of the data used by the system and the relationships between data. |
|  |  |
| Index | Several indexes to the document may be included.  As well as a normal alphabetic index, there may be an index of diagrams, an index of functions |

b) steps involved in writing a system requirement specification

1. Make an outline.
2. Define the purpose of your product.
3. Describe what you're building.
4. Detail the requirements.
5. Get it approved.