Name:babasanmi oluwafikayomi ruth

Matric no: 17/sci01/018

Assignment

1. Ai) formal methods

|  |  |
| --- | --- |
| Abstraction formal specification Formal specification | Model checking formal proofs Analytical verification |

Aii) – formal methods generally lead to more robust software and increased confidence in its corrections.

We use formal methods because the complexity of the systems with embedded software has increased.

We use formal methods because systems are becoming increasingly dependent on software components

b.

|  |  |
| --- | --- |
| Functional requirement | Nonfunctional requirement |
| The end result is of product feature. | The end result is of product properties |
| It helps to verify the functionality of the software. | It helps to verity the performance of the software. |
| It focuses on user requirement | It concentrates on user’s expectation |
| Test execution is done before non-functional testing. | Test execution is done after the functional testing |
| The documentation describes what the product does. | The documentation describes how the product works |

1. A. – agile development methodology
* Devops deployment methodology
* Waterfall development methodology
* Rapid application development methodology
1. Agile development methodology: teams use the agile development methodology to minimize risk such as bugs, cost overruns 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 agile like scrum, crystal, extreame programming.
2. Devops development methodology: it is not just a development methodology but also a set of practices that supports an organizational culture. It centers on organizational changes that enhances collaboration between the departments responsible for different segments of the development life cycle such as development, quality assurance
3. Waterfall development methodology: this is a rigid linear model that consists of sequential phases (requirements, designs, implementation, verification, maintenance) focusing on distinct goals. Each phase must be complete before the next phase starts. There’s visually no process for going back to modify the project or direction.
4. Rapid application development methodology: RAD is a condensed development process that produces a high quality system with low investment costs. RAD contains four phases; requirements planning, user design, construction and cutover. The user design and construction phases repeat until the user confirms that the products meet all requirements.

B. techniques and tools based on mathematics and formal logic can assume various forms and levels of rigor.

Least rigorous spectrum rigor most rigorous

 Occasional mathematical notation -------------- fully formal specification

 Embedded in English specification language with a precise semantics

1. Ai) - Propositional logic converts a complete sentence into symbols and makes it logical whereas in first order predicate logic relation of a particular sentence will be made that involves relations, constants, functions, and constants.
* The limitation of predicate logic is that it does not represent any individual entities whereas first order logic can easily represent the individual establishment that means if you are writing a single sentence then it can be easily represented in first order logic.
* Predicate logic does not signify or express the generalization, specialization or pattern for example quantifiers cannot be used in predicate logic but in first order logic users can easily use quantifiers as it does express the generalization, specialization, and pattern.

Aii)

b. m1= mortal

m2=man

p1= everyman is moral

p2= smith is a man

(smith is mortal)

V (m2): m1(p1,p2)

1. o= object

m= monkey

~m= not monkey

ᴲ(o): m ~m

4b) - history-based specification: a.) behavior based on system histories. B.) assertions are inserted over time

* State-based specification: a.) behavior based on system states. B.) series of sequential steps e.g a financial transaction. C.) languages such as Z, VDM or B rely on this paradigm.
* Transition-based specification: a.) behavior based on transitions from state to state of the system. B.) best used with a reactive system. C.) languages such as PROMELA, state charts rely on this paradigm.

5a) – quantifier: it is an expression that indicates the scope of a term to which it is attached.

* Predicate: this is something which is affirmed or denied concerning an argument of a proposition.
* A term: it denotes a mathematical object.

### 6a) - Waterfall Model: The classic waterfall model was introduced in the 1970s by Win Royce at Lockheed. It is so named because it can be represented or graphically modeled as a cascade from establishing requirements, to design creation, to program implementation, to system test. It was a great step forward in software development as an engineering discipline.

### Rapid Prototyping Model

### Rapid prototyping has long been used in the development of one-off programs, based on the familiar model of the chemical engineer’s pilot plant. More recently it has been used to prototype larger systems in two variants—the "throwaway" model and the "operational" model,  A throwaway prototype approach is often used if the goal is to test the implementation method, language, or end-user acceptability.

### Extreme Programming

### Extreme Programming (XP) is a rather recent development of the incremental model that puts the client in the driver’s seat. Each feature or feature set of the final product envisioned by the client and the development team is individually scoped for cost and development time.  This development model is distinguished by its flexibility because it can work in the face of a high degree of specification ambiguity on the user’s part.

### 6b) The software requirements document (also called [software requirements specification or SRS](https://www.google.com.eg/url?sa=t&rct=j&q=&esrc=s&source=web&cd=10&cad=rja&uact=8&ved=0CFAQFjAJ&url=http%3A%2F%2Fwww.csse.monash.edu.au%2F~sitar%2FCSE4002%2FLectures%2Fsrs_template-1.doc&ei=dx_DVICYNYfbPZz7gOgL&usg=AFQjCNF2DZX7_wbYlZyeAcrgjpYYTOhF6g&sig2=llfrACPbfZhd4ca5jxr-lg)) is an official document of what should be implemented. It’s also used as a contract between the system buyer and the software developers.

7.

  Make an outline.

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

### Create an Outline (Or Use an SRS Template)

Your first step is to create an outline for your software requirements specification. This may be something you create yourself. Or you may use an existing SRS template.

If you’re creating this yourself, here’s what your outline might look like:

1. Introduction

1.1 Purpose

1.2 Intended Audience

1.3 Intended Use

1.4 Scope

1.5 Definitions and Acronyms

2. Overall Description

2.1 User Needs

2.2 Assumptions and Dependencies

3. System Features and Requirements

            3.1 Functional Requirements

            3.2 External Interface Requirements

            3.3 System Features

            3.4 Nonfunctional Requirements

 The introduction to your SRS is very important. It sets the expectation for the product you’re building.

So, start by defining the purpose of your product.

#### Intended Audience and Intended Use

Define who in your organization will have access to the SRS — and how they should use it. This may include developers, testers, and project managers. It could also include stakeholders in other departments, including leadership teams, sales, and marketing.

#### Product Scope

Describe the software being specified. And include benefits, objectives, and goals. This should relate to overall business goals, especially if teams outside of development will have access to the SRS.

#### Definitions and Acronyms

It’s smart to include a risk definition. Avoiding risk is top-of-mind for many developers — especially those working on safety-critical development teams.

Here’s an example. If you’re creating a medical device, the risk might be the device fails and causes a fatality.

By defining that risk up front, it’s easier to determine the specific requirements you’ll need to mitigate it.

### 3. Give an Overview of What You’ll Build

Your next step is to give a description of what you’re going to build. Is it an update to an existing product? Is it a new product? Is it an add-on to a product you’ve already created?

These are important to describe upfront, so everyone knows what you’re building.

You should also describe why you’re building it and who it’s for.

#### User Needs

User needs — or user classes and characteristics — are critical. You’ll need to define who is going to use the product and how.

You’ll have primary and secondary users who will use the product on a regular basis. You may also need to define the needs of a separate buyer of the product (who may not be a primary/secondary user). And, for example, if you’re building a medical device, you’ll need to describe the patient’s needs.

#### Assumptions and Dependencies

There might be factors that impact your ability to fulfill the requirements outlined in your SRS. What are those factors?

Are there any assumptions you’re making with the SRS that could turn out to be false? You should include those here, as well.

Finally, you should note if your project is dependent on any external factors. This might include software components you’re reusing from another project.

### 4. Detail Your Specific Requirements

The next section is key for your development team. This is where you detail the specific requirements for building your product.

#### Functional Requirements

Functional requirements are essential to building your product.

If you’re developing a medical device, these requirements may include infusion and battery. And within these functional requirements, you may have a subset of risks and requirements.

#### External Interface Requirements

External interface requirements are types of functional requirements. They’re important for embedded systems. And they outline how your product will interface with other components.

There are several types of interfaces you may have requirements for, including:

* User
* Hardware
* Software
* Communications

#### System Features

System features are types of functional requirements. These are features that are required in order for a system to function.

#### Other Nonfunctional Requirements

Nonfunctional requirements can be just as important as functional ones.

These include:

* Performance
* Safety
* Security
* Quality

The importance of this type of requirement may vary depending on your industry. Safety requirements, for example, will be critical in the medical device industry.

[*IEEE*](https://ieeexplore.ieee.org/document/278253) also provides guidance for writing software requirements specifications, if you’re a member.

### 5. Get Approval for the SRS

Once you’ve completed the SRS, you’ll need to get it approved by key stakeholders. And everyone should be reviewing the latest version of the document.