CS445/CS645/ECE451 Final Exam - Fall 2019

6 December 2019

Instructions

Based on notes:

- **Requirements Types:** D (Scope Determined) and G (Scope Determining).
- **Domain Modeling:** Domain Ignoramus, SYS, ENV, INTF, Shared Phenomena.
- **RE Concepts:** ZJVF, Use Case, Scenario, NFR, SRS.
- **Temporal Logic:** LTL, Temporal Connectives ($\Box$, $\Diamond$, $\bigcirc$, $\mathcal{U}$, $\mathcal{W}$).
- **UI and Documentation:** UI, Platt's Law, Why Software Sucks, UM.
- **Other:** Prescriptive vs. Descriptive Specs, Phenomenon A, Phenomenon B.

Question 1: Domain and Use Case Model of IEKS and Unknown Domains

(a) Adding Voice Command Backup

Diagram Modification (Refer to the provided image):

1. Add a new component: **SE:VoiceRecognizer(1)** inside **S:CarRadio(1)**.
2. Add a link between **SE:Mic(1)** and **SE:VoiceRecognizer(1)** inside **S:CarRadio(1)**.
3. Add a link between **SE:VoiceRecognizer(1)** and **S:StarterAndDriving(1)** inside **PriusPREK**.
4. Add a link between **E:Driver()** and **SE:Mic(1)**.
5. Highlight **S:CarRadio(1)** and **SE:Mic(1)**.

Text Modification:

- Under **S:CarRadio(1)**, add:
  - **SE:VoiceRecognizer(1)**

Explanation: We introduce a **VoiceRecognizer** as a new component of the **CarRadio**. It receives input from the existing **Mic** and communicates with the **StarterAndDriving** component. This reflects the domain knowledge that voice recognition requires a microphone and processing to trigger the start. We also highlight existing components used in the new feature.

Use Case Diagram Modification (Refer to the provided image):

1. Add a new use case bubble: **Speak Start Command**
2. Draw a line from **Driver** to **Speak Start Command**.

Use Case Text Modification:

- Add new use case: **StarterAndDriving.VoiceRecognizer.SpeakToStart [D,p]**

Explanation: We add a new use case representing the driver speaking the start command. This aligns with the added domain entities and reflects the new interaction path. We specify that the Driver (D) interacts with the Prius (p) to perform this action. The use case is placed under the StarterAndDriving section because it directly impacts the car's starting mechanism.

(b) Dilbert Domain Model and Use Case Model

Domain Model (DM):

LDC	**RFIT**	**ORCAT**	**MRT** ¡¡actor¿¿
		1	1
1	1 height	NERKED: Boolean	
URPED: Boolean	NERK(P: ORCAT)		URP() height

LDC — RFIT

ORCAT — MRT

Explanation:

- **LDC**: Stereotype ¡¡actor¿¿, multiplicity 1 (as it's the one initiating the action).
- **RFIT**: Attribute NERKED, method NERK(P) with parameter P of type ORCAT. Multiplicity 1.
- **ORCAT**: Multiplicity 1.
- **MRT**: Attribute URPED, method URP(). Multiplicity 1.
- **Links**: LDC is linked to RFIT (as LDC does the NERKing). RFIT is linked to ORCAT (as RFIT gets NERKED to ORCAT). ORCAT is linked to MRT (as the action on ORCAT causes a reaction on MRT). The link between RFIT and ORCAT represents the parameter passing for the NERK method, satisfying the requirement that the DM shows how an ORCAT can be an argument.

Use Case Model (UCM): LDC height | IEKS height NERK $\uparrow\uparrow$ LDC URP $\uparrow\uparrow$ LDC height |

Explanation:

- **Actor**: LDC.
- **Use Cases**: NERK, URP (derived from the DM's methods).
- **Links**: LDC is linked to both use cases, indicating that the LDC initiates these actions.

Missing Class for Parameter (P): ORCAT is the missing class.

Question 2: RE Reference Model, Verification & Validation, and Inspection

(a) Universal Statement CRSIN

Complete the sentence: to be part of S is considered dangerous because *if CRSIN, which is part of D, is not true in the real world, but S assumes it is, then S will not satisfy R in that situation.*

Complete the sentence: Therefore, S must not deal with only what happens when CRSIN is true. S must deal with also *the case where CRSIN is false, meaning that a resident of Canada does not have a unique SIN, or even has no SIN.*

Complete the sentence: Thus, the programmer must consider as two separate cases *the case where CRSIN is true (which probably occurs more often at run time) and the case where CRSIN is false (which probably requires more code to handle the exception).*

Explanation:

- The danger lies in the assumption that D is always true. If S relies on D being true, but D is false in reality, then S won't fulfill the requirements (R).
- S needs to account for both scenarios: when D (CRSIN) holds and when it doesn't.
- The programmer must consider the "normal" case (CRSIN true, likely more frequent) and the exceptional case (CRSIN false, likely requiring more complex error handling).

(b) Formality

Is R formal? No.

Why or why not? *R is not formal because it describes desired changes in the real world, which is inherently informal. Whether R is satisfied depends on the real-world environment and user needs, which are not subject to formal mathematical proof but rather to empirical validation.*

Is M formal? No.

Why or why not? *M is not formal because its behavior is learned from data (d) representing the real world. The real world is informal, so M's learned behavior is also informal and cannot be proven through mathematical logic.*

Explanation:

- Formality implies mathematical provability.
- R deals with real-world desires, making it informal.
- M learns from real-world data, inheriting its informality.

Is B formal? No.

Why or why not? *B is not formal because its function depends on the physical and chemical properties of the real world (protein shapes and interactions). These are governed by physical laws, not mathematical logic, making B's behavior subject to empirical verification rather than formal proof.*

Explanation: B's operation relies on physical interactions, making it informal like the real world.

(c) Inspection

Inspection of any document can be described as a search for defects in the inspected document. In this two-part process, the first part, the preparation step is done by the document inspectors while the second part, the logging meeting step is done by the document's author and the inspectors, as they try to log all the defects found in the inspected document.

Explanation: Inspection involves a systematic search for defects, with a preparation phase (individual review) followed by a logging meeting (collaborative defect identification).

(d) Validation vs. Verification

Data from various sources, including the Iceberg slides, show that fixing defects found during operation (post-release) costs 10-200 times more than fixing defects found during development. Since validation (finding defects related to incorrect or missing requirements) often happens later in the development cycle or even post-release, this cost difference implies that validation is significantly harder (and more expensive) than verification (which typically occurs earlier during development).

Explanation: The high cost of fixing post-release defects, many of which stem from validation issues, indicates that validation is more challenging.

(e) Testing and Ambiguity

Are we certain that the program is completely correct? No.

Why or why not? *Even with 1,000,000 successful test cases, we cannot be certain the program is completely correct. Testing can only show the presence of defects, not their absence. There might be untested scenarios or edge cases where the program fails.*

Explanation: Testing proves defects exist, not that they don't.

Are we certain that the sentence is completely not ambiguous? No.

Why or why not? *Even if 10 people agree on one interpretation, it's possible that they all share a common misunderstanding or that the sentence is ambiguous in a way not captured by their interpretations. There might be other contexts or perspectives where the sentence could be interpreted differently.*

Explanation: Agreement doesn't guarantee a lack of ambiguity.

(f) Active Review

Main drawback: *Like normal testing, an active review can only show the presence of defects, not their absence. It might not cover all possible scenarios or reveal all potential issues.*

Advantage: *An active review can be performed before any code is written, allowing for early detection and correction of requirements errors, which is much cheaper than fixing them later in the development cycle.*

Explanation: Active reviews share testing's limitations but offer early error detection.

Question 3: Elicitation, User Interfaces, and User's Manuals

(a) Microsoft 365 for Corporate Buyers

i. **Owner/Client:** Microsoft.

ii. **Customer:** The corporation buying the software (e.g., a company purchasing licenses for its employees).

iii. **User:** The employees of the corporation who use the software.

(b) Microsoft 365 for Individual Buyers

i. **Owner/Client:** Microsoft.

ii. **Customer:** The individual buyer (you or me).

iii. **User:** The individual buyer (you or me).

(c) Program Developed for Own Use

i. **Owner/Client:** You.

ii. **Customer:** You.

iii. **User:** You.

(d) Program Developed by Boeing's Software Developers

i. **Owner/Client:** Boeing.

ii. **Customer:** Boeing's engineering department or a specific project within Boeing.

iii. **User:** Boeing's engineers.

(e) Sources Not to Use in Elicitation

Is there any source of information about a CBS that you intend to build that you should not use in requirements elicitation? Yes.

Why or why not? *You should not use sources that contain implementation details or design decisions that are not relevant to the user's needs or the system's external behavior. Focusing on such sources can lead to implementation bias, restricting design freedom and potentially creating a system that is not aligned with the actual user requirements. It's important to focus on the "what" (user needs, external behavior) rather than the "how" (implementation details) during elicitation.*

Explanation: Avoid sources with implementation bias that don't reflect user needs.

(f) CBS with Heavy User Interaction

(i) **Main reason:** *The main reason is that the user interface is part of the system's external behavior and defines how users interact with the system's functions. The UI's design directly impacts how functions are invoked and how output is presented. Therefore, the UI and functions are tightly coupled and must be specified together to ensure consistency and usability.*

(ii) *A failure to have this architecture from the beginning in an agile development of C leads to the necessity of major refactoring.*

Explanation: The UI defines interaction; it's coupled with functions and needs specification to avoid inconsistencies. Missing this in agile leads to refactoring.

(g) Distinguishing Word Uses

Here are four uses of the word *enter*:
Please press the **enter** key in order to see the text "`enter`" on the screen.
Explanation:

- *Italics*: Used for the word "enter" when referring to its meaning.
- Underline with added word "press": Used to indicate the action on a keyboard key.
- **Bold**: Used for the name of a keyboard key.
- `Monospace` with quotation marks: Used for the word "enter" when it appears as part of input or output.

(h) Code, Ideas, SRS, and UM

While code in a programming language is a very precise way to specify the behavior of software, it is very difficult to modify when it is found to be wrong for any reason.

On the other hand, ideas in someone's mind about the behavior of software are very easy to change when they are found to be wrong for any reason. However, it is difficult for anyone to know for sure what the ideas are.

The nice thing about an SRS or UM is that it both (1) is easier to change than is program code when it is found to be wrong for any reason and (2) is clearer than are ideas for anyone to know for sure what the SRS or UM says.

Explanation: Code is hard to change; ideas are easy to change but hard to communicate; SRS/UM is easier to change than code and clearer than ideas.

Question 4: NFRs and Cost Estimation

(a) Verification or Validation?

*This decision is **verification** if S contains only functional requirements.*

Example of a functional requirement: If the user enters their PIN, and the PIN is correct, then the system shall grant access.

Explanation: Verification checks if the system meets the specified requirements. For functional requirements, this is straightforward.

(b) Problems with "The CBS shall be user friendly."

- *The term "user-friendly" is subjective and not clearly defined. Different users may have different interpretations of what constitutes user-friendliness.*
- *It is difficult to measure or test whether a system is "user-friendly" in an objective way. There are no specific, quantifiable criteria to determine if this requirement is met.*

Explanation: Subjectivity and lack of measurability make it hard to define and test.

(c) Problems with "The CBS shall have a fast response time."

- *The term "fast" is vague and relative. It does not specify a clear, measurable performance target.*
- *The required response time may vary depending on the specific operation or context. A single "fast" criterion may not be appropriate for all situations.*

Explanation: Vagueness and context-dependency make it difficult to define and apply.

(d) Linguists

Linguists consider a word like fast to be vague because, in any situation, it has no clear, natural, obvious meaning.
Explanation: "Fast" lacks a precise, universal meaning.

(e) Mathematicians

Mathematicians consider whether a CBS has a fast response time to be (1) not Boolean, with answers `true` or `false` and nothing in between, but to be (2) continuous, with answers having all values ranging from $\underline{0}$ to $\underline{\infty}$ (infinity).
Explanation: Response time is a continuous variable, not a binary true/false.

(f) Sources of Error in Cost Prediction

- *Incomplete or incorrect understanding of the requirements, leading to underestimation of the work involved.*
- *Unforeseen technical challenges or changes in technology that can increase development time and effort.*
- *Inaccurate estimation of the size or complexity of the software, leading to miscalculation of resource needs.*

Explanation: Inaccurate requirements, unforeseen issues, and incorrect size/complexity estimates cause errors.

(g) Biggest Influences on Cost

- *The size and complexity of the software being developed.*
- *The experience and skill level of the development team.*
- *The number and type of Non-Functional Requirements (NFRs) and their associated constraints.*

Explanation: Size/complexity, team skill, and NFRs are major cost drivers.

(h) Function Point Analysis

Function Point Analysis shows how to estimate the number of function points for a CBS from essentially the CBS's use cases, and then to estimate the CBS's code size from these same function points.
Explanation: Function points, derived from use cases, help estimate code size.

(i) COCOMO

COCOMO shows how to estimate for a CBS, both its

1. project effort in person months, and
2. project duration in months

from the CBS's estimated code size obtained from a Function Point Analysis of the CBS.
Explanation: COCOMO uses code size (from function points) to estimate effort and duration.

(j) Team Size and Productivity

Answer: 7

Explanation: Let n be the team size. The total productive work is $8n$. The communication overhead is $n(n-1)$. The net productive work is $8n - n(n-1)$. We want to find the smallest n such that adding one more person (increasing the team size to $n+1$) reduces the net productive work. So we need to find n such that: $8n - n(n-1) > 8(n+1) - (n+1)(n+1-1)$ $8n - n^2 + n > 8n + 8 - n^2 - n$ $9n - n^2 > 7n + 8 - n^2$ $2n > 8$ $n > 4$ When $n = 7$, adding one more person (to make it 8) reduces productive work. $8(7) - 7(6) = 56 - 42 = 14$ $8(8) - 8(7) = 64 - 56 = 8$

Hence, Fred Brooks said, "Adding more people to a late project makes it even later.".

Question 5: Ambiguity

(a) Distributing "only"

1. ✓Only my e-key operates my Prius.
2. My only e-key operates my Prius.
3. My e-key only operates my Prius.
4. My e-key operates only my Prius.
5. ✓My e-key operates my only Prius.

Sentence with as many onlys as possible: My only e-key operates my only Prius.
Explanation:

- Sentences 1 and 5 are true given A.
- Sentence 1: Only one thing operates my Prius, and that is my e-key.
- Sentence 5: I have only one Prius, and my e-key operates it.
- The combined sentence emphasizes that I have one e-key and one Prius, and the e-key operates the Prius.

(b) Three Additional Methods

(1) My only finger prints also operate my Prius.

(2) My only PIN also operates my Prius.

(3) A cellphone loaded with an app that imitates my e-key also operates only my Prius.

Explanation:

- "Only" is needed before "finger prints" and "PIN" to indicate that these are specific instances of those methods belonging to me.
- "Also" is needed to indicate that these are additional methods besides the e-key.
- In sentence 8, "only" is needed before "my Prius" to emphasize that the cellphone app only operates my specific Prius.

(c) Sentences Still True Given A and B

Given A and B, none of sentences 1 through 5 are still true.

Explanation: A and B state that other methods besides the e-key can operate the Prius, invalidating all the sentences.

(d) Joe's Finger Prints

(9) Joe's <u>only</u> finger prints <u>also</u> operate <u>only</u> my Prius.

Explanation:

- "Only" before "finger prints" specifies Joe's prints, not just any finger prints.
- "Also" indicates this is an additional method.
- "Only" before "my Prius" clarifies that Joe's prints operate only my car.

(e) True Sentences Given A, B, and C

10. Only my e-key operates my Prius.
11. Only an e-key operates my Prius.

Explanation: Neither sentence is true because A, B, and C state that other methods besides an e-key (and specifically my e-key) can operate the Prius.

(f) Correcting Incorrect Sentences

12. Pay for ✓what only you need.
13. A paper's abstract summarizes ✓only the paper.
14. I smoke ✓only Winstons.
15. Of the two versions, please read the second ✓only in full detail.
16. I eat ✓only vegetables.
17. A hamburger should be eaten ✓only after cooking it.

Explanation: In each case, "only" is moved to immediately precede the word or phrase it logically modifies.

(g) Empirical Test

The empirical test involved giving people sentences with "only" misplaced before the main verb and asking them to correct the sentences. The observation was that most people incorrectly placed "only" before the main verb instead of before the word being limited.

Explanation: People were tested on their placement of "only" in sentences.

(h) Developer's Action on Ambiguous Statement

The developer should seek clarification from the stakeholders who provided the specification. They should ask for a more precise and unambiguous statement of the requirement to ensure that the implemented system meets the intended needs.

Explanation: Clarify with stakeholders to resolve ambiguity.

Question 6: State Machines and Linear Temporal Logic

(a) Modified State Machine

KULBP[CHK] DDUL+4ODL

TMK[¬KIC] ↙ ↓ PIUDB[¬KIC]

ADL ADUL

KLBP[KOC & CHK] ↗ BUP1[¬CHK]

↓ [KIC]

↑ [KIC]

PSP[KIC]

HSSable → HSOn

←

PSP[CHK]

Legend:

- ADL: all doors locked
- DDUL+4ODL: driver door unlocked and 4 other doors locked
- ADUL: all doors unlocked
- HSSable: hybrid system startable
- HSOn: hybrid system on
- KULBP: key's unlock button pushed
- KLBP: key's lock button pushed
- PSP: power switch pushed
- CHK: car hears key
- Pw2SoA: pushed within 2 seconds of another
- KIC: key in cabin
- KOC: key outside cabin
- TMK: Turn Manual Key
- PIUDB: Push Interior Unlock-Doors Button
- BUP1: Backup Method 1
- BUP2: Backup Method 2

Explanation:

- **TMK**: Added a transition from ADL to DDUL+4ODL labeled "TMK[¬KIC]" (Turn Manual Key when the key is not in the cabin).
- **PIUDB**: Added a transition from DDUL+4ODL to ADUL labeled "PIUDB[¬KIC]" (Push Interior Unlock-Doors Button when the key is not in the cabin).
- **BUP1**: Added a transition from ADUL to HSSable labeled "BUP1[¬CHK]" (Backup Method 1 when the car does not hear the key).
- **BUP2**: The transition from ADL to HSSable will be replaced by a new state machine for Backup Method 2 as explained in part (b).

(b) Replacing Backup Method Transitions

Complete the sentence: To fully reflect the behavior of a backup method, each such simple transition should be replaced by *a new state machine that describes the specific steps and conditions involved in that backup method, including any intermediate states and transitions.*

Explanation: A single transition is insufficient; a detailed state machine is needed for each backup method.

(c) Linear Temporal Logic Specification of the Original State Machine

1. $\Box(\text{ADL} \wedge \text{KULBP} \wedge \text{CHK} \rightarrow \bigcirc \text{DDUL+4ODL})$
2. $\Box(\text{DDUL+4ODL} \wedge \text{KULBP} \wedge \text{CHK} \wedge \text{Pw2SoA} \rightarrow \bigcirc \text{ADUL})$
3. $\Box(\text{ADUL} \wedge \text{KIC} \rightarrow \bigcirc \text{HSSable})$
4. $\Box(\text{HSSable} \wedge \text{PSP} \rightarrow \bigcirc \text{HSOn})$
5. $\Box(\text{HSOn} \wedge \text{PSP} \rightarrow \bigcirc \text{HSSable})$
6. $\Box(\text{HSSable} \wedge \text{KLBP} \wedge \text{KOC} \wedge \text{CHK} \rightarrow \bigcirc \text{ADL})$
7. $\Box(\text{ADL} \wedge \text{KLBP} \wedge \text{KOC} \wedge \text{CHK} \rightarrow \bigcirc \text{ADL})$

Explanation: Each formula describes a transition in the original state machine using LTL connectives.

(d) Temporal Logic Formulae

1. T $(A) \Rightarrow \Diamond(A)$
2. F $\Diamond(A) \Rightarrow \Box(A)$
3. T $\Box(A) \Rightarrow \Box(\Box(A))$
4. T $\Box(A) \Rightarrow \Diamond(A)$
5. T $\Box(A) \Rightarrow \bigcirc(A)$
6. F $\bigcirc(A) \Rightarrow \Box(A)$
7. T $\Diamond(\Box(A)) \Rightarrow \bigcirc(\Box(A))$
8. F $\bigcirc(\Box(A)) \Rightarrow \Diamond(\Box(A))$
9. T $\Diamond(\bigcirc(A)) \Rightarrow \bigcirc(\Diamond(A))$
10. T $\bigcirc(\Diamond(A)) \Rightarrow \Diamond(\bigcirc(A))$
11. F $\bigcirc(\Box(A)) \Rightarrow \Box(\bigcirc(A))$
12. F $\bigcirc(\Diamond(A)) \Rightarrow \Diamond(\bigcirc(A))$
13. T $\Box(\Diamond(A)) \Rightarrow \Diamond(\Box(A))$
14. T $\Diamond(\Box(A)) \Rightarrow \Box(\Diamond(A))$
15. T $(A \,\mathcal{U}\, B) \Rightarrow \Diamond(B)$
16. F $\Diamond(B) \Rightarrow (A \,\mathcal{U}\, B)$
17. T $\Box(A) \Rightarrow (\bigcirc(A) \vee \Box(A))$
18. T $(\bigcirc(A) \vee \Box(A)) \Rightarrow \Diamond(A)$

Explanation: Each formula is evaluated based on the definitions of the LTL connectives.

Question 7: Requirements Determination is Unstoppable and the Requirements Iceberg

(a) Reasons for Continued Requirements Determination

The most fundamental reasons are that (1) the initial specification S is often incomplete, meaning that not all necessary details or functionalities are captured, and (2) new requirements or changes to existing requirements may emerge during development as the stakeholders' understanding of the system evolves or as external factors change. As developers write code and encounter unforeseen issues or edge cases not covered by S, they need to determine how to handle them, effectively continuing the requirements determination process. Similarly, new insights or changing needs from the customer may necessitate revisiting and updating the requirements.

Explanation: Incomplete initial specs and evolving needs/understanding necessitate continued requirements determination.

(b) Informal and Formal in Jackson's Quotation

Informal: *The informal part refers to the real-world needs, desires, and goals of the stakeholders, as well as the properties and constraints of the environment in which the system will operate. These are often expressed in natural language and are subject to interpretation and change.*

Formal: *The formal part refers to the precise, unambiguous specification of the system's behavior, often expressed using formal methods like logic, state machines, or mathematical models. This formal specification serves as the basis for implementation and verification.*

Explanation: Informal = real-world needs/environment; Formal = precise system specification.

(c) Scope Determining and Scope Determined Requirements

Scope Determining (G):

- *The IEKS shall support unlocking the car using a smartphone app.*
- *The IEKS shall allow authorized users to start the car remotely.*

Scope Determined (D):

- *The IEKS shall recognize an unlock command within 2 seconds of receiving a valid signal from the key fob.*
- *The IEKS shall lock all doors when the key fob's lock button is pressed and the key fob is outside the car.*

Explanation:

- G requirements define new features (smartphone app, remote start).
- D requirements specify details within the defined scope (unlock timing, lock conditions).

(d) Head Start a Poor Bet

This head start is a poor bet because it assumes that the developers' current understanding of the domain is accurate and complete, which is often not the case. A significant portion of the code written based on this incomplete understanding (potentially much more than (100-P)%) may need to be changed or discarded when the actual requirements are determined. This can lead to wasted effort and rework, ultimately delaying the project rather than providing a head start. The larger P is, the smaller the wasted effort is, but the effort to talk to the customer to elicit the requirements is unaffected by P. Therefore, the larger P is, the smaller the fraction of total effort is wasted, but the smaller P is, the larger the fraction of effort is wasted.

A better use of the rest of the team would be to involve them in the requirements elicitation process, including activities like brainstorming, scenario development, and domain modeling. This helps ensure that everyone on the team develops a shared understanding of the requirements and reduces the risk of wasted effort later.

The Martin Tsai study showed that involving the development team in the requirements elicitation process (in this case, by having them build a prototype based on an initial SRS) helped uncover significant errors and ambiguities in the SRS, which would have been much harder and costlier to discover later. This demonstrates the benefit of having the whole team engaged in understanding the requirements early on.

(e) Discover and Specify All Requirements

Why is doing so impossible? *It's impossible to discover and specify all requirements upfront because (1) stakeholders' needs and understanding evolve over time, leading to new requirements or changes to existing ones, and (2) the complexity of real-world systems often makes it impossible to anticipate all possible scenarios and edge cases upfront.*

Why is doing so necessary? *While impossible to achieve fully, striving to discover and specify as many requirements as possible upfront is necessary to minimize the cost and effort of rework later. Clear and complete requirements provide a solid foundation for design and implementation, reducing the likelihood of costly changes and errors later in the development cycle.*

So, what is a solution? *A practical solution is to focus on identifying and specifying the scope-determining (G) requirements first. These establish the overall scope and provide a framework for defining the scope-determined (D) requirements. As the project progresses and more details become known, the D requirements can be added and refined iteratively, reducing the impact of evolving needs and unforeseen issues.*

(f) Software vs. Program

In what ways is each of the following software?

(1) A complete NL RS, S, is software because it represents an executable model of the desired system behavior, albeit at a high level of abstraction. It can be "executed" by humans to understand how the system is supposed to behave in different scenarios.

(2) A C program P that correctly implements S is software because it represents an executable model of the desired system behavior, albeit at a low level of abstraction. It can be "executed" by computer hardware to produce the actual system behavior.

In what ways are the so-called computers that execute S and P different?

The "computer" that executes S is a human, who uses natural language processing and reasoning to understand and interpret the requirements. The "computer" that executes P is a computer hardware system, which uses electronic circuits and machine code to perform calculations and operations.

(g) ARPAnet/Internet

Original Requirements: The original requirements for the ARPAnet were that it be *able to continue functioning even if parts of the network were destroyed or unavailable due to attack or failure.*

Does the Internet satisfy these original requirements? *Yes, the Internet largely satisfies these original requirements due to its decentralized architecture and routing protocols, allowing it to route traffic around failed or unavailable nodes. However, the increasing centralization of certain services and infrastructure introduces new vulnerabilities.*

Explain how the Internet is a classic E-type system: *The Internet is a classic E-type system because it evolves and adapts to changing user needs and technological advancements. It exhibits self-modifying behavior as new protocols, services, and applications are developed and deployed, continuously changing the system's functionality and complexity.*

Explain what often happens with a requirements-exploring prototype: *Often, the prototype is quickly developed with minimal concern for non-functional requirements like performance, security, and maintainability. When this prototype is then pressed into production, these neglected aspects become major problems, leading to instability, security vulnerabilities, and difficulties in maintenance and scaling.*

Explain what should happen in this case: *A new system should be built from scratch, taking into account all the learned requirements from the prototype, including both functional and non-functional requirements. The prototype serves as a valuable source of information but should not be directly used as the basis for the production system.*

What does Berry say needs to be done to fix the Internet to be secure? *Berry suggests that a fundamental redesign of the Internet's architecture and protocols is necessary to achieve true security. This would involve addressing issues like the lack of built-in authentication and authorization mechanisms, the vulnerability to spoofing and denial-of-service attacks, and the difficulty in tracking down malicious actors.*

What does Kleinrock say needs to be done to save the Internet from the dark side? *Kleinrock emphasizes the need for a layered approach to security, involving technological solutions, legal and regulatory frameworks, and ethical guidelines. He advocates for developing "architectural immunity" to make the Internet more resilient to attacks and for fostering a culture of responsible online behavior.*

Question 8: Graduate Student Lectures

(a) Henry Pabst's Internship

Major mistake during elicitation: *Henry's major mistake was not talking to the key stakeholders who actually perform the monthly key rotation. He relied on his own assumptions and understanding of the process, which turned out to be incomplete and inaccurate.*

Bad result of this mistake: *As a result, Henry developed a solution that did not fully address the stakeholders' needs and workflow. His automated tool did reduce the number of keys to manage but did not fit seamlessly into the existing process, ultimately not achieving the desired improvement in rotation speed.*

Other mistakes in elicitation:

- *Not spending enough time understanding the existing system and its complexities.*

- *Making assumptions about the stakeholders' needs without validating them.*
- *Not considering the broader context of the key rotation process and its implications.*

Realistic deployment expectation? *No, it was not realistic for Henry to expect deployment within his 4-month internship. His plan underestimated the time needed for requirements elicitation, system understanding, and development, as well as potential integration challenges.*

More reasonable deliverable: *A more reasonable deliverable would have been a detailed requirements specification, a working prototype demonstrating the core functionality of his tool, and a plan for integration and testing.*

(b) RongHao Yang's Internships

Advantages of I-company:

- *More resources and support.*
- *Opportunity to work on a larger scale project.*
- *Potential for greater impact.*

Advantages of W-company:

- *More autonomy and ownership.*
- *Faster-paced development cycle.*
- *Greater opportunity to learn and grow.*

1 **Why build an optimizing tool?** *I-company wanted to build an optimizing tool to improve the performance of its flagship database (IDB), potentially attracting more customers and increasing revenue.*

Why AI unnecessary? *RongHao found that existing SQL optimization techniques were sufficient to achieve the desired performance improvements. The AI aspect added complexity without providing significant practical benefit.*

2 **Why an AI-based tool?** *I-company likely wanted to market its database as being "AI-powered" to gain a competitive advantage and attract attention, even if the AI aspect wasn't technically necessary.*

Reason for secrecy: *I-company might have been keeping the AI aspect secret to avoid revealing its marketing strategy to competitors or to manage customer expectations about the actual role of AI in the tool.*

Kind of requirement (i): *Functional requirement*

Kind of requirement (ii): *Non-functional requirement (NFR)*

Refinement of "kind" classification: As a refinement of this "kind" classification, Requirement (ii) is a technology ability requirement.