Prompt 1:

"""

You are a helpful assistant. Use the following context to answer the question.

Context:

{context}

Question:

{question}

""")

question 1:

WHat is software enginerring?

1)

Model: meta-llama/Llama-3.3-70B-Instruct-Turbo-Free

Prompt : prompt 1

Question : question 1

Embedding model: sentence-transformers/all-MiniLM-L6

Chunking method: recursive text splitting chunk size 800 and overlap 100

Answer:

According to the provided context, software engineering is defined as:

\* "A process, methods for managing and engineering software, and tools" (Page 13)

\* "A discipline or a body of knowledge, [but] engineering is a verb, an action word, a way of approaching a problem" (Page 13, quoting Scott Whitmir)

\* "Encompass[ing] a process, methods, and tools that enable complex computer-based systems to be built in a timely manner with quality" (Page 53)

In general, software engineering is a field that encompasses a set of principles, methods, and tools used to design, develop, and maintain software systems. It involves a systematic approach to software development, focusing on quality, timeliness, and customer satisfaction.

📚 Context Chunks Used:

--- Chunk 1 ---

[Source: C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf, Page 53, Chunk software\_engineering.pdf:53:2]

Software engineering encompasses process, methods, and tools that enable complex computer-based systems to be built in a timely manner with quality. The software process incorporates five framework activities—communication, planning, modeling, construction, and deployment—that are applicable to all software proj- ects. Software engineering practice is a problem solving activity that follows a set of core principles. A wide array of software myths continue to lead managers and practitioners astray, even as our collective knowledge of software and the technologies required to build it grows. As you learn more about software engineering, you’ll begin to un- derstand why these myths should be debunked whenever they are encountered. PROBLEMS AND POINTS TO PONDER

--- Chunk 2 ---

[Source: C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf, Page 41, Chunk software\_engineering.pdf:41:4]

CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 13 uote: “More than a discipline or a body of knowledge, engineering is a verb, an action word, a way of approaching a problem.” Scott Whitmir How do we define software engineering? ? 9 For numerous additional definitions of software engineering, see www.answers.com/topic/ software-engineering#wp-\_note-13. 10 Quality management and related approaches are discussed in Chapter 14 and throughout Part 3 of this book. Software engineering encompasses a process, methods for managing and engineering software, and tools. pre75977\_ch01.qxd 11/27/08 3:11 PM Page 13

--- Chunk 3 ---

[Source: C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf, Page 25, Chunk software\_engineering.pdf:25:1]

It has been almost three decades since the first edition of this book was written. During that time, software engineering has evolved from an obscure idea practiced by a relatively small number of zealots to a legitimate engineering discipline. Today, it is recognized as a subject worthy of serious research, conscientious study, and tumultuous debate. Through- out the industry, software engineer has replaced programmer as the job title of preference. Software process models, software engineering methods, and software tools have been adopted successfully across a broad spectrum of industry segments. Although managers and practitioners alike recognize the need for a more disciplined approach to software, they continue to debate the manner in which discipline is to be

--- Chunk 4 ---

[Source: C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf, Page 25, Chunk software\_engineering.pdf:25:2]

approach to software, they continue to debate the manner in which discipline is to be applied. Many individuals and companies still develop software haphazardly, even as they build systems to service today’s most advanced technologies. Many professionals and students are unaware of modern methods. And as a result, the quality of the software that we produce suffers, and bad things happen. In addition, debate and controversy about the true nature of the software engineering approach continue. The status of software engi- neering is a study in contrasts. Attitudes have changed, progress has been made, but much remains to be done before the discipline reaches full maturity. The seventh edition of Software Engineering: A Practitioner’s Approach is intended to

--- Chunk 5 ---

[Source: C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf, Page 42, Chunk software\_engineering.pdf:42:1]

each area of the technology and include modeling activities and other descriptive techniques. Software engineering tools provide automated or semiautomated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called computer-aided software engineering, is established. 1.4 T HE SOFTWARE PROCESS A process is a collection of activities, actions, and tasks that are performed when some work product is to be created. An activity strives to achieve a broad objective (e.g., communication with stakeholders) and is applied regardless of the application domain, size of the project, complexity of the effort, or degree of rigor with which

2)

Model: meta-llama/Llama-3.3-70B-Instruct-Turbo-Free

Prompt : prompt 1

Question : question 1

Embedding model: model\_name="Qwen/Qwen3-Embedding-0.6B",

Chunking method: recursive text splitting chunk size 800 and overlap 100

📘 Answer:

Software engineering encompasses a process, methods, and tools that enable complex computer-based systems to be built in a timely manner with quality. It is a broad array of principles, concepts, methods, and tools that are used to plan and develop software. Software engineering is a problem-solving activity that follows a set of core principles and provides a road map for getting to a successful destination in the creation of a computer-based system or product. It involves a set of framework activities, including communication, planning, modeling, construction, and deployment, and is conducted by software engineers and their managers. Overall, software engineering is a discipline that aims to improve the ease with which changes can be accommodated and reduce the amount of effort expended when changes are made to software systems.

📚 Context Chunks Used:

--- Chunk 1 ---

[Source: C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf, Page 53, Chunk software\_engineering.pdf:53:2]

Software engineering encompasses process, methods, and tools that enable complex computer-based systems to be built in a timely manner with quality. The software process incorporates five framework activities—communication, planning, modeling, construction, and deployment—that are applicable to all software proj- ects. Software engineering practice is a problem solving activity that follows a set of core principles. A wide array of software myths continue to lead managers and practitioners astray, even as our collective knowledge of software and the technologies required to build it grows. As you learn more about software engineering, you’ll begin to un- derstand why these myths should be debunked whenever they are encountered. PROBLEMS AND POINTS TO PONDER

--- Chunk 2 ---

[Source: C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf, Page 124, Chunk software\_engineering.pdf:124:2]

requirements . .107 testing . . . . . . .112 What is it? Software engineering practice is a broad array of princi- ples, concepts, methods, and tools that you must consider as software is planned and developed. Principles that guide practice establish a foundation from which soft- ware engineering is conducted. Who does it? Practitioners (software engineers) and their managers conduct a variety of soft- ware engineering tasks. Why is it important? The software process pro- vides everyone involved in the creation of a computer-based system or product with a road map for getting to a successful destination. Practice provides you with the detail you’ll need to drive along the road. It tells you where the bridges, the roadblocks, and the forks are located.

--- Chunk 3 ---

[Source: C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf, Page 41, Chunk software\_engineering.pdf:41:4]

CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 13 uote: “More than a discipline or a body of knowledge, engineering is a verb, an action word, a way of approaching a problem.” Scott Whitmir How do we define software engineering? ? 9 For numerous additional definitions of software engineering, see www.answers.com/topic/ software-engineering#wp-\_note-13. 10 Quality management and related approaches are discussed in Chapter 14 and throughout Part 3 of this book. Software engineering encompasses a process, methods for managing and engineering software, and tools. pre75977\_ch01.qxd 11/27/08 3:11 PM Page 13

--- Chunk 4 ---

[Source: C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf, Page 125, Chunk software\_engineering.pdf:125:4]

1 Some writers argue for one of these terms to the exclusion of the others. In reality, software engineering is all three. pre75977\_ch04.qxd 11/27/08 3:27 PM Page 97

--- Chunk 5 ---

[Source: C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf, Page 613, Chunk software\_engineering.pdf:613:0]

SCM activities are developed to (1) identify change, (2) control change, (3) ensure that change is being properly implemented, and (4) report changes to others who may have an interest. It is important to make a clear distinction between software support and software configuration management. Support is a set of software engineering activities that occur after software has been delivered to the customer and put into operation. Software configuration management is a set of tracking and control activities that are initiated when a software engineering project begins and terminates only when the software is taken out of operation. A primary goal of software engineering is to improve the ease with which changes can be accommodated and reduce the amount of effort expended when changes

3)

Model: meta-llama/Llama-3.3-70B-Instruct-Turbo-Free

Prompt : prompt 1

Question : question 1

Embedding model: model\_name="Qwen/Qwen3-Embedding-0.6B",

Chunking method: semantic chunking

4)

LLM: gemini-1.5-flash

Prompt: prompt1

Embedding model: Qwen/Qwen3-Embedding-0.6B

Chunking method: semantic chunking with Qwen/Qwen3-Embedding-0.6B

Reanking :none  
📘 Answer:

Based on the provided text, software engineering is defined in several ways:

1. \*\*The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.\*\* This definition emphasizes a structured and measurable process.

2. \*\*The study of approaches as in (1).\*\* This highlights the academic and research aspects of the field.

3. Software engineering encompasses \*\*process, methods, and tools\*\* that enable complex computer-based systems to be built in a timely manner with quality. This emphasizes the practical aspects, including the software development process, techniques used, and supporting tools.

The text also notes that software engineering rests on an organizational commitment to quality and is a layered technology, with the process layer forming its foundation. It's described as a problem-solving activity following core principles and requiring adaptability and agility.

📚 Context Chunks Used:

--- Chunk 1 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 32, Chunk software\_engineering.pdf:32:1]

Software is engineered, not manufactured. pre75977\_ch01.qxd 11/27/08 3:11 PM Page 4

--- Chunk 2 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 32, Chunk software\_engineering.pdf:32:0]

programmer are the same questions that are asked when modern computer-based systems are built:1 4 CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 1 In an excellent book of essays on the software business, Tom DeMarco [DeM95] argues the coun- terpoint. He states: “Instead of asking why software costs so much, we need to begin asking ‘What have we done to make it possible for today’s software to cost so little?’ The answer to that ques- tion will help us continue the extraordinary level of achievement that has always distinguished the software industry.” • Why does it take so long to get software finished? • Why are development costs so high? • Why can’t we find all errors before we give the software to our customers? • Why do we spend so much time and effort maintaining existing programs? • Why do we continue to have difficulty in measuring progress as software is being developed and maintained? These, and many other questions, are a manifestation of the concern about software and the manner in which it is developed—a concern that has lead to the adoption of software engineering practice. 1.1.1 Defining Software Today, most professionals and many members of the public at large feel that they understand software. But do they? A textbook description of software might take the following form: Software is: (1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structures that enable the programs to ad- equately manipulate information, and (3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs. There is no question that other more complete definitions could be offered. But a more formal definition probably won’t measurably improve your under- standing. To accomplish that, it’s important to examine the characteristics of soft- ware that make it different from other things that human beings build. Software is a logical rather than a physical system element. Therefore, software has characteris- tics that are considerably different than those of hardware: 1. Software is developed or engineered; it is not manufactured in the classical sense. Although some similarities exist between software development and hard- ware manufacturing, the two activities are fundamentally different. In both activities, high quality is achieved through good design, but the manufactur- ing phase for hardware can introduce quality problems that are nonexistent How should we define software? ?

--- Chunk 3 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 42, Chunk software\_engineering.pdf:42:0]

that must be established for effective delivery of software engineering technology. The software process forms the basis for management control of software projects and establishes the context in which technical methods are applied, work products (models, documents, data, reports, forms, etc.) are produced, milestones are estab- lished, quality is ensured, and change is properly managed. Software engineering methods provide the technical how-to’s for building soft- ware.

--- Chunk 4 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 41, Chunk software\_engineering.pdf:41:1]

What are the “sound engineering principles” that can be applied to computer software development? How do we “economically” build software so that it is “reliable”? What is required to create computer programs that work “efficiently” on not one but many different “real machines”? These are the questions that continue to challenge software engineers. The IEEE [IEE93a] has developed a more comprehensive definition when it states: Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1). And yet, a “systematic, disciplined, and quantifiable” approach applied by one software team may be burdensome to another. We need discipline, but we also need adaptability and agility. Software engineering is a layered technology. Referring to Figure 1.3, any engineer- ing approach (including software engineering) must rest on an organizational com- mitment to quality. Total quality management, Six Sigma, and similar philosophies 10 foster a continuous process improvement culture, and it is this culture that ultimately leads to the development of increasingly more effective approaches to software engi- neering. The bedrock that supports software engineering is a quality focus. The foundation for software engineering is the process layer. The software engi- neering process is the glue that holds the technology layers together and enables rational and timely development of computer software. Process defines a framework CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 13 uote: “More than a discipline or a body of knowledge, engineering is a verb, an action word, a way of approaching a problem.” Scott Whitmir How do we define software engineering? ? 9 For numerous additional definitions of software engineering, see www.answers.com/topic/ software-engineering#wp-\_note-13. 10 Quality management and related approaches are discussed in Chapter 14 and throughout Part 3 of this book. Software engineering encompasses a process, methods for managing and engineering software, and tools.

--- Chunk 5 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 53, Chunk software\_engineering.pdf:53:0]

CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 25 With the exception of a passing reference, software was hardly mentioned as part of the conversation. And yet, software will make or break the SafeHome product line. The engineering effort will succeed only if SafeHomesoftware succeeds. The market will accept the product only if the software embedded within it properly meets the customer’s (as yet unstated) needs. We’ll follow the progression of SafeHome software engineering in many of the chapters that follow. 1.8 S UMMARY Software is the key element in the evolution of computer-based systems and products and one of the most important technologies on the world stage. Over the past 50 years, software has evolved from a specialized problem solving and infor- mation analysis tool to an industry in itself. Yet we still have trouble developing high- quality software on time and within budget. Software—programs, data, and descriptive information—addresses a wide array of technology and application areas. Legacy software continues to present special challenges to those who must maintain it. Web-based systems and applications have evolved from simple collections of in- formation content to sophisticated systems that present complex functionality and multimedia content. Although these WebApps have unique features and require- ments, they are software nonetheless. Software engineering encompasses process, methods, and tools that enable complex computer-based systems to be built in a timely manner with quality. The software process incorporates five framework activities—communication, planning, modeling, construction, and deployment—that are applicable to all software proj- ects. Software engineering practice is a problem solving activity that follows a set of core principles.

5)

LLM: gemini-1.5-flash

Prompt: prompt1

Embedding model: Qwen/Qwen3-Embedding-0.6B

Chunking method: semantic chunking with Qwen/Qwen3-Embedding-0.6B

Reanking : model used cross-encoder/ms-marco-MiniLM-L-6-v2

response = chain.run(inputs)

📘 Answer:

Based on the provided text, software engineering is defined in several ways:

1. \*\*The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.\*\* This definition emphasizes a structured and measurable process.

2. \*\*The study of approaches as in (1).\*\* This highlights the academic and research aspects of the field.

3. A layered technology resting on an organizational commitment to quality, with the software engineering process forming the foundation. This emphasizes the importance of a quality-focused culture and a well-defined process.

4. Software engineering encompasses a process, methods for managing and engineering software, and tools. This broader definition includes the practical aspects of software development.

In essence, software engineering is a multi-faceted field that combines technical skills, management practices, and a focus on quality to build and maintain computer software effectively. It addresses challenges like high costs, long development times, and difficulty in error detection and progress measurement.

📚 Context Chunks Used:

--- Chunk 1 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 32, Chunk software\_engineering.pdf:32:0]

programmer are the same questions that are asked when modern computer-based systems are built:1 4 CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 1 In an excellent book of essays on the software business, Tom DeMarco [DeM95] argues the coun- terpoint. He states: “Instead of asking why software costs so much, we need to begin asking ‘What have we done to make it possible for today’s software to cost so little?’ The answer to that ques- tion will help us continue the extraordinary level of achievement that has always distinguished the software industry.” • Why does it take so long to get software finished? • Why are development costs so high? • Why can’t we find all errors before we give the software to our customers? • Why do we spend so much time and effort maintaining existing programs? • Why do we continue to have difficulty in measuring progress as software is being developed and maintained? These, and many other questions, are a manifestation of the concern about software and the manner in which it is developed—a concern that has lead to the adoption of software engineering practice. 1.1.1 Defining Software Today, most professionals and many members of the public at large feel that they understand software. But do they? A textbook description of software might take the following form: Software is: (1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structures that enable the programs to ad- equately manipulate information, and (3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs. There is no question that other more complete definitions could be offered. But a more formal definition probably won’t measurably improve your under- standing. To accomplish that, it’s important to examine the characteristics of soft- ware that make it different from other things that human beings build. Software is a logical rather than a physical system element. Therefore, software has characteris- tics that are considerably different than those of hardware: 1. Software is developed or engineered; it is not manufactured in the classical sense. Although some similarities exist between software development and hard- ware manufacturing, the two activities are fundamentally different. In both activities, high quality is achieved through good design, but the manufactur- ing phase for hardware can introduce quality problems that are nonexistent How should we define software? ?

--- Chunk 2 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 42, Chunk software\_engineering.pdf:42:0]

that must be established for effective delivery of software engineering technology. The software process forms the basis for management control of software projects and establishes the context in which technical methods are applied, work products (models, documents, data, reports, forms, etc.) are produced, milestones are estab- lished, quality is ensured, and change is properly managed. Software engineering methods provide the technical how-to’s for building soft- ware.

--- Chunk 3 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 41, Chunk software\_engineering.pdf:41:1]

What are the “sound engineering principles” that can be applied to computer software development? How do we “economically” build software so that it is “reliable”? What is required to create computer programs that work “efficiently” on not one but many different “real machines”? These are the questions that continue to challenge software engineers. The IEEE [IEE93a] has developed a more comprehensive definition when it states: Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1). And yet, a “systematic, disciplined, and quantifiable” approach applied by one software team may be burdensome to another. We need discipline, but we also need adaptability and agility. Software engineering is a layered technology. Referring to Figure 1.3, any engineer- ing approach (including software engineering) must rest on an organizational com- mitment to quality. Total quality management, Six Sigma, and similar philosophies 10 foster a continuous process improvement culture, and it is this culture that ultimately leads to the development of increasingly more effective approaches to software engi- neering. The bedrock that supports software engineering is a quality focus. The foundation for software engineering is the process layer. The software engi- neering process is the glue that holds the technology layers together and enables rational and timely development of computer software. Process defines a framework CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 13 uote: “More than a discipline or a body of knowledge, engineering is a verb, an action word, a way of approaching a problem.” Scott Whitmir How do we define software engineering? ? 9 For numerous additional definitions of software engineering, see www.answers.com/topic/ software-engineering#wp-\_note-13. 10 Quality management and related approaches are discussed in Chapter 14 and throughout Part 3 of this book. Software engineering encompasses a process, methods for managing and engineering software, and tools.

--- Chunk 4 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 53, Chunk software\_engineering.pdf:53:0]

CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 25 With the exception of a passing reference, software was hardly mentioned as part of the conversation. And yet, software will make or break the SafeHome product line. The engineering effort will succeed only if SafeHomesoftware succeeds. The market will accept the product only if the software embedded within it properly meets the customer’s (as yet unstated) needs. We’ll follow the progression of SafeHome software engineering in many of the chapters that follow. 1.8 S UMMARY Software is the key element in the evolution of computer-based systems and products and one of the most important technologies on the world stage. Over the past 50 years, software has evolved from a specialized problem solving and infor- mation analysis tool to an industry in itself. Yet we still have trouble developing high- quality software on time and within budget. Software—programs, data, and descriptive information—addresses a wide array of technology and application areas. Legacy software continues to present special challenges to those who must maintain it. Web-based systems and applications have evolved from simple collections of in- formation content to sophisticated systems that present complex functionality and multimedia content. Although these WebApps have unique features and require- ments, they are software nonetheless. Software engineering encompasses process, methods, and tools that enable complex computer-based systems to be built in a timely manner with quality. The software process incorporates five framework activities—communication, planning, modeling, construction, and deployment—that are applicable to all software proj- ects. Software engineering practice is a problem solving activity that follows a set of core principles.

--- Chunk 5 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 32, Chunk software\_engineering.pdf:32:1]

Software is engineered, not manufactured. pre75977\_ch01.qxd 11/27/08 3:11 PM Page 4

**6)**

LLM: gemini-1.5-flash

Prompt: prompt1

Embedding model: Qwen/Qwen3-Embedding-0.6B

Chunking method: semantic chunking with Qwen/Qwen3-Embedding-0.6B

Reanking : model used cross-encoder/ms-marco-MiniLM-L-6-v2

response = chain.run(inputs)

**📘 Answer:**

**\*\*Question 1:\*\***

**Analyze the differences between embedded software, product-line software, web applications, and artificial intelligence software, providing specific examples from the provided text and illustrating how their functionalities and target markets vary. Consider the evolution of WebApps as part of your analysis.**

**\*\*Rubric:\*\***

**- \*\*Categorization & Comparison (3 marks):\*\* Accurately identifies and differentiates the four software categories, highlighting key distinctions in functionality and application.**

**- \*\*Example Application (3 marks):\*\* Provides relevant and specific examples from the text to support the distinctions made between the software categories.**

**- \*\*Analysis of WebApp Evolution (2 marks):\*\* Demonstrates understanding of the evolution of WebApps from simple hypertext to sophisticated integrated systems.**

**- \*\*Clarity and Organization (2 marks):\*\* Presents the analysis in a clear, concise, and well-organized manner.**

**Total: 10 marks**

**\*\*Question 2:\*\***

**Critically evaluate the statement: "Software is engineered, not manufactured." Discuss the implications of this statement in terms of software development processes, cost distribution, and the relationship between people and the work accomplished. Use evidence from the provided text to support your evaluation.**

**\*\*Rubric:\*\***

**- \*\*Understanding of the Core Statement (3 marks):\*\* Demonstrates a clear understanding of the central assertion that software is engineered, not manufactured.**

**- \*\*Analysis of Implications (4 marks):\*\* Analyzes the implications of this statement regarding software development processes, cost distribution, and the human element in software development.**

**- \*\*Use of Evidence (3 marks):\*\* Effectively uses specific evidence from the text to support the analysis and evaluation.**

**Total: 10 marks**

**\*\*Question 3:\*\***

**Synthesize the information presented about software myths and realities. Develop a concise argument explaining why understanding these myths is crucial for successful software project management. Focus on the myth regarding the completion of work after the program is running.**

**\*\*Rubric:\*\***

**- \*\*Identification of Key Myths (2 marks):\*\* Accurately identifies and summarizes at least two relevant software myths from the provided text.**

**- \*\*Analysis of Myth Impact (3 marks):\*\* Analyzes the impact of at least one chosen myth (e.g., the myth about work completion after the program runs) on software project management.**

**- \*\*Argument Development (3 marks):\*\* Develops a clear and logical argument explaining the importance of understanding software myths for successful project management.**

**- \*\*Clarity and Conciseness (2 marks):\*\* Presents the argument in a clear, concise, and well-structured manner.**

**Total: 10 marks**

**\*\*Question 4:\*\***

**Evaluate the challenges and opportunities presented by the diverse nature of software applications (embedded, product-line, web applications, AI). Discuss how these challenges and opportunities impact the software engineering process and the skills required of software engineers.**

**\*\*Rubric:\*\***

**- \*\*Identification of Challenges (3 marks):\*\* Identifies at least three significant challenges posed by the diversity of software applications.**

**- \*\*Identification of Opportunities (3 marks):\*\* Identifies at least three significant opportunities created by the diversity of software applications.**

**- \*\*Impact on Software Engineering (2 marks):\*\* Discusses the impact of these challenges and opportunities on the software engineering process.**

**- \*\*Skills Required (2 marks):\*\* Identifies the key skills needed by software engineers to address these challenges and opportunities.**

**Total: 10 marks**

**\*\*Question 5:\*\***

**Analyze the statement: "we have yet to develop a software technology that does it all, and the likelihood of one arising in the future is small." Discuss the factors contributing to this situation and explain why specialized software technologies continue to be developed for specific application domains or technology domains.**

**\*\*Rubric:\*\***

**- \*\*Understanding of the Statement (2 marks):\*\* Demonstrates a clear understanding of the statement's meaning and implications.**

**- \*\*Analysis of Contributing Factors (4 marks):\*\* Identifies and analyzes at least three factors contributing to the continued development of specialized software technologies.**

**- \*\*Explanation of Specialization (2 marks):\*\* Explains why specialized software technologies are necessary and continue to be developed.**

**- \*\*Clarity and Logic (2 marks):\*\* Presents the analysis in a clear, logical, and well-organized manner.**

**Total: 10 marks**

**📚 Context Chunks Used:**

**--- Chunk 1 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 36, Chunk software\_engineering.pdf:36:1]**

**Embedded software can perform limited and esoteric functions (e.g., key pad control for a microwave oven) or provide significant function and control capability (e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems). Product-line software—designed to provide a specific capability for use by many different customers. Product-line software can focus on a limited and esoteric marketplace (e.g., inventory control products) or address mass consumer markets (e.g., word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, and personal and business financial applications). Web applications—called “WebApps,” this network-centric software cate- gory spans a wide array of applications. In their simplest form, WebApps can be little more than a set of linked hypertext files that present information using text and limited graphics. However, as Web 2.0 emerges, WebApps are evolving into sophisticated computing environments that not only provide stand-alone features, computing functions, and content to the end user, but also are integrated with corporate databases and business applications. Artificial intelligence software—makes use of nonnumerical algorithms to solve complex problems that are not amenable to computation or straightfor- ward analysis. Applications within this area include robotics, expert systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing.**

**--- Chunk 2 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 31, Chunk software\_engineering.pdf:31:0]**

**to build and maintain high-quality computer programs. Some of these technologies are targeted at a specific application domain (e.g., website design and implementa- tion); others focus on a technology domain (e.g., object-oriented systems or aspect- oriented programming); and still others are broad-based (e.g., operating systems such as Linux). However, we have yet to develop a software technology that does it all, and the likelihood of one arising in the future is small.**

**--- Chunk 3 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 51, Chunk software\_engineering.pdf:51:1]**

**Myths that are still believed by software practitioners have been fostered by over 50 years of programming culture. During the early days, pro- gramming was viewed as an art form. Old ways and attitudes die hard. Myth: Once we write the program and get it to work, our job is done. Reality: Someone once said that “the sooner you begin ‘writing code,’ the longer it’ll take you to get done.” Industry data indicate that between 60 and 80 percent of all effort expended on software will be ex- pended after it is delivered to the customer for the first time. Myth: Until I get the program “running” I have no way of assessing its quality. Reality: One of the most effective software quality assurance mechanisms can be applied from the inception of a project— the technical review. Software reviews (described in Chapter 15) are a “quality filter” that have been found to be more effective than testing for finding certain classes of software defects. Myth: The only deliverable work product for a successful project is the working program.**

**--- Chunk 4 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 32, Chunk software\_engineering.pdf:32:1]**

**Software is engineered, not manufactured. pre75977\_ch01.qxd 11/27/08 3:11 PM Page 4**

**--- Chunk 5 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 33, Chunk software\_engineering.pdf:33:0]**

**(or easily corrected) for software. Both activities are dependent on people, but the relationship between people applied and work accomplished is entirely different (see Chapter 24). Both activities require the construction of a “product,” but the approaches are different. Software costs are concen- trated in engineering.**

7)

LLM: gemini-1.5-flash

Prompt: prompt1

Embedding model: Qwen/Qwen3-Embedding-0.6B

Chunking method: semantic chunking with Qwen/Qwen3-Embedding-0.6B

Reanking : model used BAAI/bge-reranker-v2-m3

Retrieval: dense retrieval using : Qwen/Qwen3-Embedding-0.6B

📘 Answer:

\*\*Question 1:\*\*

Compare and contrast the characteristics of embedded software, product-line software, web applications, and artificial intelligence software, providing specific examples from the provided text to illustrate your points. Discuss the implications of these differences for software development methodologies.

\*\*Rubric:\*\*

- \*\*Categorization and Comparison:\*\* Accurately identifies and compares the key characteristics of each software type (2 marks)

- \*\*Illustrative Examples:\*\* Provides relevant and specific examples from the text to support the comparison (3 marks)

- \*\*Methodological Implications:\*\* Discusses the impact of these differences on software development approaches (3 marks)

- \*\*Clarity and Organization:\*\* Presents the comparison in a clear, logical, and well-organized manner (2 marks)

Total: 10 marks

\*\*Question 2:\*\*

The text states that "Software is engineered, not manufactured." Explain this statement, contrasting the processes and challenges involved in software engineering with those of traditional manufacturing. Use examples to support your explanation.

\*\*Rubric:\*\*

- \*\*Understanding of the Core Concept:\*\* Clearly explains the fundamental difference between software engineering and manufacturing (2 marks)

- \*\*Process Comparison:\*\* Compares and contrasts the processes involved in each (3 marks)

- \*\*Challenge Identification:\*\* Identifies key challenges unique to software engineering (2 marks)

- \*\*Use of Examples:\*\* Supports the explanation with clear and relevant examples (3 marks)

Total: 10 marks

\*\*Question 3:\*\*

Analyze the three software myths presented in the text. For each myth, explain why it is a misconception and describe the reality that contradicts it. Discuss the potential consequences of believing these myths.

\*\*Rubric:\*\*

- \*\*Myth Identification and Explanation:\*\* Accurately identifies and explains each myth (3 marks)

- \*\*Reality Presentation:\*\* Clearly presents the reality that contradicts each myth (3 marks)

- \*\*Consequences Analysis:\*\* Analyzes the potential negative consequences of believing these myths (4 marks)

Total: 10 marks

\*\*Question 4:\*\*

Based on the provided text, discuss the evolution of web applications from simple hypertext files to sophisticated computing environments integrated with corporate databases and business applications. What factors have driven this evolution?

\*\*Rubric:\*\*

- \*\*Evolutionary Description:\*\* Accurately describes the evolution of web applications (3 marks)

- \*\*Driving Factors Identification:\*\* Identifies and explains the key factors driving this evolution (4 marks)

- \*\*Supporting Evidence:\*\* Uses evidence from the text to support the description and analysis (3 marks)

Total: 10 marks

\*\*Question 5:\*\*

The text mentions that software costs are concentrated in engineering. Explain why this is the case, contrasting it with the cost distribution in traditional manufacturing. What implications does this have for software project management?

\*\*Rubric:\*\*

- \*\*Cost Concentration Explanation:\*\* Clearly explains why software costs are concentrated in engineering (3 marks)

- \*\*Comparison with Manufacturing:\*\* Effectively contrasts this with cost distribution in manufacturing (3 marks)

- \*\*Project Management Implications:\*\* Discusses the implications of this cost distribution for software project management (4 marks)

Total: 10 marks

📚 Context Chunks Used:

--- Chunk 1 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 36, Chunk software\_engineering.pdf:36:1]

Embedded software can perform limited and esoteric functions (e.g., key pad control for a microwave oven) or provide significant function and control capability (e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems). Product-line software—designed to provide a specific capability for use by many different customers. Product-line software can focus on a limited and esoteric marketplace (e.g., inventory control products) or address mass consumer markets (e.g., word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, and personal and business financial applications). Web applications—called “WebApps,” this network-centric software cate- gory spans a wide array of applications. In their simplest form, WebApps can be little more than a set of linked hypertext files that present information using text and limited graphics. However, as Web 2.0 emerges, WebApps are evolving into sophisticated computing environments that not only provide stand-alone features, computing functions, and content to the end user, but also are integrated with corporate databases and business applications. Artificial intelligence software—makes use of nonnumerical algorithms to solve complex problems that are not amenable to computation or straightfor- ward analysis. Applications within this area include robotics, expert systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing.

--- Chunk 2 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 32, Chunk software\_engineering.pdf:32:1]

Software is engineered, not manufactured. pre75977\_ch01.qxd 11/27/08 3:11 PM Page 4

--- Chunk 3 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 51, Chunk software\_engineering.pdf:51:1]

Myths that are still believed by software practitioners have been fostered by over 50 years of programming culture. During the early days, pro- gramming was viewed as an art form. Old ways and attitudes die hard. Myth: Once we write the program and get it to work, our job is done. Reality: Someone once said that “the sooner you begin ‘writing code,’ the longer it’ll take you to get done.” Industry data indicate that between 60 and 80 percent of all effort expended on software will be ex- pended after it is delivered to the customer for the first time. Myth: Until I get the program “running” I have no way of assessing its quality. Reality: One of the most effective software quality assurance mechanisms can be applied from the inception of a project— the technical review. Software reviews (described in Chapter 15) are a “quality filter” that have been found to be more effective than testing for finding certain classes of software defects. Myth: The only deliverable work product for a successful project is the working program.

--- Chunk 4 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 33, Chunk software\_engineering.pdf:33:0]

(or easily corrected) for software. Both activities are dependent on people, but the relationship between people applied and work accomplished is entirely different (see Chapter 24). Both activities require the construction of a “product,” but the approaches are different. Software costs are concen- trated in engineering.

--- Chunk 5 ---

[Source: [C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf), Page 31, Chunk software\_engineering.pdf:31:0]

to build and maintain high-quality computer programs. Some of these technologies are targeted at a specific application domain (e.g., website design and implementa- tion); others focus on a technology domain (e.g., object-oriented systems or aspect- oriented programming); and still others are broad-based (e.g., operating systems such as Linux). However, we have yet to develop a software technology that does it all, and the likelihood of one arising in the future is small.

**8)**

LLM: gemini-1.5-flash

Prompt: prompt1

Embedding model: Qwen/Qwen3-Embedding-0.6B

Chunking method: semantic chunking with Qwen/Qwen3-Embedding-0.6B

Reanking : model used BAAI/bge-reranker-v2-m3

**Retreival : Sparse retrieval as well as dense retrieval**

**response = chain.run(inputs)**

**📘 Answer:**

**\*\*Question 1:\*\* Based on the provided text, compare and contrast the nature of software development with that of hardware manufacturing. Analyze the implications of these differences for software quality, maintenance, and the overall lifecycle of a software product. Consider DeMarco's counterpoint regarding software costs in your analysis.**

**Rubric:**

**\* \*\*Conceptual Understanding of Software vs. Hardware:\*\* Demonstrates a clear understanding of the fundamental differences between software and hardware, including their respective development processes and characteristics (3 marks). \*Excellent: Accurately identifies and explains key differences, such as software's logical nature vs. hardware's physical nature, and the implications of these differences.\* \*Good: Identifies most key differences but may lack depth in explanation.\* \*Fair: Identifies some differences but with inaccuracies or incompleteness.\* \*Poor: Fails to identify significant differences.\***

**\* \*\*Analysis of Software Lifecycle Implications:\*\* Analyzes the impact of these differences on software quality, maintenance, and lifecycle, providing specific examples from the text (3 marks). \*Excellent: Provides a thorough analysis, linking the differences to specific challenges in software development and maintenance.\* \*Good: Provides a good analysis but may lack depth or specific examples.\* \*Fair: Offers a superficial analysis with limited examples.\* \*Poor: Fails to adequately analyze the implications.\***

**\* \*\*Critical Evaluation of DeMarco's Argument:\*\* Critically evaluates DeMarco's counterpoint regarding software costs, considering its validity and implications for the software industry (4 marks). \*Excellent: Provides a nuanced evaluation, considering both sides of the argument and offering insightful conclusions.\* \*Good: Presents a valid evaluation but may lack depth or critical analysis.\* \*Fair: Offers a superficial evaluation with limited critical insight.\* \*Poor: Fails to adequately evaluate DeMarco's argument.\***

**Total: 10 marks**

**\*\*Question 2:\*\* The text states that "software doesn’t wear out, but it does deteriorate." Explain this statement, contrasting the failure curve of software with that of hardware. Then, propose at least three practical strategies for mitigating software deterioration, drawing upon your understanding of software engineering principles.**

**Rubric:**

**\* \*\*Explanation of Software Deterioration:\*\* Clearly explains the difference between "wear out" and "deterioration" in the context of software and hardware, using the provided information and illustrating with examples (3 marks). \*Excellent: Provides a clear and concise explanation with relevant examples illustrating the concept.\* \*Good: Explains the difference but may lack clarity or specific examples.\* \*Fair: Offers a partial explanation with limited understanding.\* \*Poor: Fails to adequately explain the concept.\***

**\* \*\*Comparison of Failure Curves:\*\* Accurately compares and contrasts the failure curves of software and hardware, highlighting the key differences and their implications (3 marks). \*Excellent: Provides a detailed comparison, accurately representing the characteristics of each curve and their significance.\* \*Good: Compares the curves but may lack precision or depth of analysis.\* \*Fair: Offers a superficial comparison with inaccuracies.\* \*Poor: Fails to adequately compare the curves.\***

**\* \*\*Proposed Mitigation Strategies:\*\* Proposes at least three practical strategies for mitigating software deterioration, justifying each strategy with sound software engineering principles (4 marks). \*Excellent: Proposes three or more well-justified strategies, demonstrating a strong understanding of software engineering principles.\* \*Good: Proposes three strategies but may lack sufficient justification or depth.\* \*Fair: Proposes fewer than three strategies or provides weak justifications.\* \*Poor: Fails to propose effective strategies.\***

**Total: 10 marks**

**📚 Context Chunks Used:**

**--- Chunk 1 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 32, Chunk software\_engineering.pdf:32:0]**

**programmer are the same questions that are asked when modern computer-based systems are built:1 4 CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 1 In an excellent book of essays on the software business, Tom DeMarco [DeM95] argues the coun- terpoint. He states: “Instead of asking why software costs so much, we need to begin asking ‘What have we done to make it possible for today’s software to cost so little?’ The answer to that ques- tion will help us continue the extraordinary level of achievement that has always distinguished the software industry.” • Why does it take so long to get software finished? • Why are development costs so high? • Why can’t we find all errors before we give the software to our customers? • Why do we spend so much time and effort maintaining existing programs? • Why do we continue to have difficulty in measuring progress as software is being developed and maintained? These, and many other questions, are a manifestation of the concern about software and the manner in which it is developed—a concern that has lead to the adoption of software engineering practice. 1.1.1 Defining Software Today, most professionals and many members of the public at large feel that they understand software. But do they? A textbook description of software might take the following form: Software is: (1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structures that enable the programs to ad- equately manipulate information, and (3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs. There is no question that other more complete definitions could be offered. But a more formal definition probably won’t measurably improve your under- standing. To accomplish that, it’s important to examine the characteristics of soft- ware that make it different from other things that human beings build. Software is a logical rather than a physical system element. Therefore, software has characteris- tics that are considerably different than those of hardware: 1. Software is developed or engineered; it is not manufactured in the classical sense. Although some similarities exist between software development and hard- ware manufacturing, the two activities are fundamentally different. In both activities, high quality is achieved through good design, but the manufactur- ing phase for hardware can introduce quality problems that are nonexistent How should we define software? ?**

**--- Chunk 2 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 33, Chunk software\_engineering.pdf:33:2]**

**The idealized curve is a gross oversimplification of actual failure models for software. However, the implication is clear—software doesn’t wear out. But it does deteriorate! CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 5 “Wear out”“Infant mortality” Time Failure rate FIGURE 1.1 Failure curve for hardware Software doesn’t wear out, but it does deteriorate. If you want to reduce software deterioration, you’ll have to do better software design (Chapters 8 to 13). pre75977\_ch01.qxd 11/27/08 3:11 PM Page 5**

**--- Chunk 3 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 53, Chunk software\_engineering.pdf:53:1]**

**A wide array of software myths continue to lead managers and practitioners astray, even as our collective knowledge of software and the technologies required to build it grows. As you learn more about software engineering, you’ll begin to un- derstand why these myths should be debunked whenever they are encountered. PROBLEMS AND POINTS TO PONDER 1.1. Provide at least five additional examples of how the law of unintended consequences applies to computer software. 1.2. Provide a number of examples (both positive and negative) that indicate the impact of software on our society. 1.3.**

**--- Chunk 4 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 31, Chunk software\_engineering.pdf:31:0]**

**to build and maintain high-quality computer programs. Some of these technologies are targeted at a specific application domain (e.g., website design and implementa- tion); others focus on a technology domain (e.g., object-oriented systems or aspect- oriented programming); and still others are broad-based (e.g., operating systems such as Linux). However, we have yet to develop a software technology that does it all, and the likelihood of one arising in the future is small.**

**--- Chunk 5 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 34, Chunk software\_engineering.pdf:34:1]**

**When a hardware component wears out, it is replaced by a spare part. There are no software spare parts. Every software failure indicates an error in design or in the process through which design was translated into machine executable code. Therefore, the software maintenance tasks that accommodate requests for change involve considerably more complexity than hardware maintenance. 3.**

9)  
LLM: gemini-1.5-flash

Prompt: prompt1

Embedding model: Qwen/Qwen3-Embedding-0.6B

Chunking method: semantic chunking with Qwen/Qwen3-Embedding-0.6B

Reanking : model used BAAI/bge-reranker-v2-m3

**Retreival : langchain’s ensemble retriever**

**📘 Answer:**

**\*\*Question 1:\*\* Explain the concept of software deterioration as described in the provided text. Analyze the reasons why software, unlike hardware, tends to degrade over time despite ongoing maintenance and updates. Discuss the implications of this phenomenon for software development lifecycle management.**

**\*\*Rubric:\*\***

**\* \*\*Conceptual Understanding of Software Deterioration:\*\* (3 marks) Accurately defines software deterioration and explains its core mechanisms as presented in the text. Demonstrates understanding of the relationship between changes, errors, and the rising failure rate.**

**\* \*\*Analysis of Underlying Causes:\*\* (3 marks) Identifies and analyzes at least two key reasons for software deterioration, providing clear explanations and supporting evidence from the text.**

**\* \*\*Application to Software Development:\*\* (4 marks) Explains the implications of software deterioration for software development lifecycle management, suggesting at least two practical strategies to mitigate this issue.**

**Total: 10 marks**

**\*\*Question 2:\*\* The text states that "Software is engineered, not manufactured." Compare and contrast the engineering of software with the manufacturing of hardware, focusing on at least three key differences. Discuss how these differences impact the development process and the nature of the resulting product.**

**\*\*Rubric:\*\***

**\* \*\*Identification of Key Differences:\*\* (3 marks) Clearly identifies and explains at least three significant differences between software engineering and hardware manufacturing.**

**\* \*\*Comparative Analysis:\*\* (3 marks) Provides a thorough comparison of the two processes, highlighting the implications of each difference for the development process and the final product.**

**\* \*\*Impact on Development and Product:\*\* (4 marks) Analyzes the impact of these differences on the development process (e.g., cost, time, methodology) and the characteristics of the resulting software and hardware products.**

**Total: 10 marks**

**\*\*Question 3:\*\* Critically evaluate the "myth" that "Once we write the program and get it to work, our job is done." Using evidence from the provided text and your own understanding of software development, discuss the realities of post-delivery effort in software projects. What are the implications of this myth for project planning and resource allocation?**

**\*\*Rubric:\*\***

**\* \*\*Critical Evaluation of the Myth:\*\* (3 marks) Clearly articulates the flaws in the statement and provides a reasoned critique of its underlying assumptions.**

**\* \*\*Discussion of Post-Delivery Effort:\*\* (4 marks) Provides a detailed explanation of the significant effort required after initial delivery, including maintenance, updates, bug fixes, and other post-release activities.**

**\* \*\*Implications for Project Planning:\*\* (3 marks) Discusses the implications of this reality for realistic project planning, resource allocation, and cost estimation.**

**Total: 10 marks**

**\*\*Question 4:\*\* Based on the text's discussion of software myths, identify and explain at least three common misconceptions about software development. For each misconception, propose a realistic alternative perspective based on sound software engineering principles.**

**\*\*Rubric:\*\***

**\* \*\*Identification of Misconceptions:\*\* (2 marks) Clearly identifies three distinct and relevant misconceptions about software development from the provided text.**

**\* \*\*Explanation of Misconceptions:\*\* (3 marks) Provides a clear and concise explanation of why each misconception is inaccurate or misleading.**

**\* \*\*Realistic Alternative Perspectives:\*\* (5 marks) Offers well-reasoned and practical alternative perspectives for each misconception, grounded in sound software engineering principles.**

**Total: 10 marks**

**\*\*Question 5:\*\* The text mentions the "law of unintended consequences" in relation to software. Explain this law in the context of software development. Provide three specific examples of unintended consequences that can arise from software development projects, and analyze the potential impact of each.**

**\*\*Rubric:\*\***

**\* \*\*Explanation of the Law:\*\* (2 marks) Clearly defines and explains the "law of unintended consequences" as it applies to software systems.**

**\* \*\*Specific Examples:\*\* (4 marks) Provides three distinct and well-explained examples of unintended consequences in software development.**

**\* \*\*Impact Analysis:\*\* (4 marks) Analyzes the potential negative impact of each unintended consequence on users, systems, or organizations.**

**Total: 10 marks**

**📚 Context Chunks Used:**

**--- Chunk 1 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 34, Chunk software\_engineering.pdf:34:0]**

**This seeming contradiction can best be explained by considering the actual curve in Figure 1.2. During its life,2 software will undergo change. As changes are made, it is likely that errors will be introduced, causing the failure rate curve to spike as shown in the “actual curve” (Figure 1.2). Before the curve can return to the original steady-state failure rate, another change is requested, causing the curve to spike again. Slowly, the minimum failure rate level begins to rise—the software is deteriorating due to change. Another aspect of wear illustrates the difference between hardware and software.**

**--- Chunk 2 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 32, Chunk software\_engineering.pdf:32:1]**

**Software is engineered, not manufactured. pre75977\_ch01.qxd 11/27/08 3:11 PM Page 4**

**--- Chunk 3 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 53, Chunk software\_engineering.pdf:53:1]**

**A wide array of software myths continue to lead managers and practitioners astray, even as our collective knowledge of software and the technologies required to build it grows. As you learn more about software engineering, you’ll begin to un- derstand why these myths should be debunked whenever they are encountered. PROBLEMS AND POINTS TO PONDER 1.1. Provide at least five additional examples of how the law of unintended consequences applies to computer software. 1.2. Provide a number of examples (both positive and negative) that indicate the impact of software on our society. 1.3.**

**--- Chunk 4 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 51, Chunk software\_engineering.pdf:51:1]**

**Myths that are still believed by software practitioners have been fostered by over 50 years of programming culture. During the early days, pro- gramming was viewed as an art form. Old ways and attitudes die hard. Myth: Once we write the program and get it to work, our job is done. Reality: Someone once said that “the sooner you begin ‘writing code,’ the longer it’ll take you to get done.” Industry data indicate that between 60 and 80 percent of all effort expended on software will be ex- pended after it is delivered to the customer for the first time. Myth: Until I get the program “running” I have no way of assessing its quality. Reality: One of the most effective software quality assurance mechanisms can be applied from the inception of a project— the technical review. Software reviews (described in Chapter 15) are a “quality filter” that have been found to be more effective than testing for finding certain classes of software defects. Myth: The only deliverable work product for a successful project is the working program.**

**--- Chunk 5 ---**

**[Source:** [**C:\Users\dhili\Desktop\SRIP\week3\data\software\_engineering.pdf**](file:///C:\Users\dhili\Desktop\SRIP\week3\data\software_engineering.pdf)**, Page 33, Chunk software\_engineering.pdf:33:0]**

**(or easily corrected) for software. Both activities are dependent on people, but the relationship between people applied and work accomplished is entirely different (see Chapter 24). Both activities require the construction of a “product,” but the approaches are different. Software costs are concen- trated in engineering.**