

Software Project Management in the era of Software 3.0

Saeed Siddik

Assistant Professor

IIT University of Dhaka

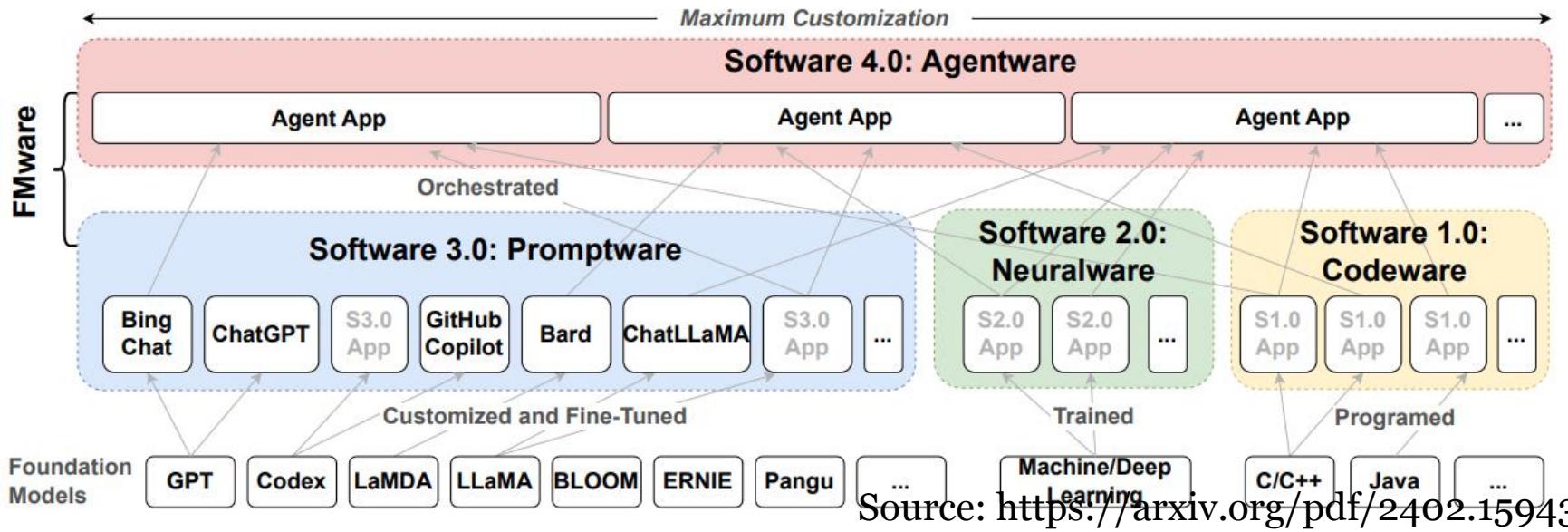
Evolution on Software Engineering

- SE 1.0 – The Code-Centric Era
 - Classical Software Engineering
- SE 2.0 – The Data, Service and Collaboration Era
 - DevOps and ML driven training-prediction
- SE 3.0 – The AI-Driven and Autonomous Era
 - LLMs and AIware in software automation

SE 3.0 - The AI-Driven and Autonomous Era (Promptware)

- Core Idea: Software is customized and fine-tuned through foundation models (FMs) such as GPT, using prompts instead of training from scratch.
- Process: Developers and users interact with pre-trained AI-models through prompt engineering and fine-tuning.
- Examples: ChatGPT, GitHub Copilot, Bard, and LLaMA.
- Characteristics:
 - Built on top of massive pre-trained models (foundation models).
 - Bridges AI with software engineering via prompt-based design.

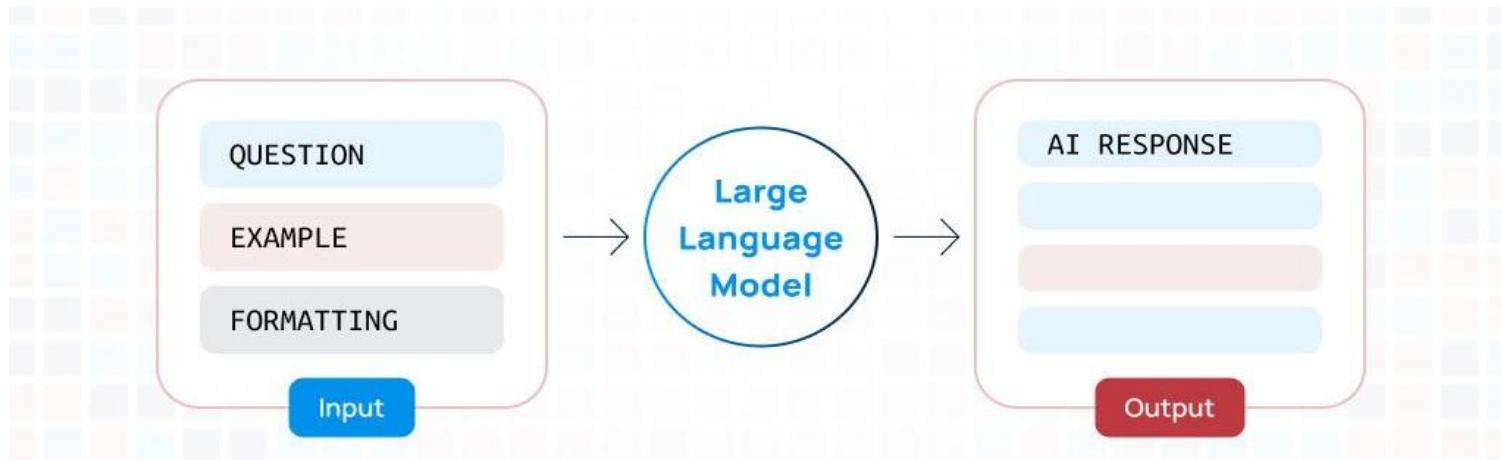
Agentware and its relation to prior software generations.



Developing Alware systems

- Prompt Engineering
- Fine-tuning
- Retrieval Augmented Generation (RAG)

Prompt Engineering



Prompt Design Strategies

- **Instructional Prompts:** direct commands (e.g., “Generate test cases for this function”).
- **Contextual Prompts:** provide background, data, or examples.
- **Role-based Prompts:** define persona (“You are a DevOps assistant...”).
- **Chain-of-Thought Prompts:** encourage stepwise reasoning.
- **System Prompts:** configure persistent behavior of AI agents.

Example of Instructional Prompt: direct command

- **Prompt:** "Generate 5 unit test cases in Python (using pytest) for the function 'compute_mean(values: List[float])' covering normal, empty, and error cases; include expected inputs and outputs."
- **Expected output:** A list of pytest test functions with inputs, expected outputs, and one test that checks raised exception for invalid input.

Example of Contextual Prompt: provide background/data/examples

- **Prompt:** "Given the following bug report and recent commit diff (paste below), summarize the root cause and propose a code patch sketch. Bug report: 'API returns 500 when input size > 10MB.' Commit diff: <paste>."
- **Expected output:** A concise root-cause analysis referencing the diff, and a short patch outline (file, function, pseudo-code) that addresses buffering/streaming or validation.

Example of Role-based Prompt: define a persona / responsibilities

- **Prompt:** "You are a Senior DevOps Engineer. Review this CI pipeline (YAML below) and list three improvements to reduce deployment failures and one rollback strategy."
- **Expected output:** Three actionable CI/CD improvements (e.g., add integration tests, implement canary releases, tighten resource limits) and a clear rollback procedure.

Example of Chain-of-Thought Prompt: encourage stepwise reasoning

- **Prompt:** "Explain step-by-step how you would find the root cause of intermittent test failures in a microservice: list hypotheses, tests to run for each hypothesis, expected observations, and next actions."
- **Expected output:** A numbered reasoning chain: hypothesis A → test A1 → expected outcome → next step, then hypothesis B → ... useful for teaching debugging workflows.

Example of System Prompt: configure persistent agent behavior

- **Prompt:** "Explain step-by-step how you would find the root cause of intermittent test failures in a microservice: list hypotheses, tests to run for each hypothesis, expected observations, and next actions."
- **Expected output:** A numbered reasoning chain: hypothesis A → test A1 → expected outcome → next step, then hypothesis B → ... useful for teaching debugging workflows.

Example of System Prompt: configure persistent agent behavior

- **Prompt:** "System: You are 'CodeQA', an assistant that always returns answers in three parts—(1) brief summary, (2) detailed explanation with references to code lines, (3) suggested test cases. Never provide legal or private data."
- **Expected output:** All subsequent responses follow the three-part structure, producing consistent, policy-compliant replies suitable for integration into an AIware pipeline.

Best Practices in Prompt Engineering

- Be specific: define output format and success criteria.
- Add context: mention task scope, tools, or dataset.
- Use examples: to guide tone, structure, or code style.
- Avoid ambiguity: test prompts iteratively.
- Evaluate responses: refine prompts like you debug software.

Prompt Engineering Best Practices



Specify an Audience



Be Clear and Specific



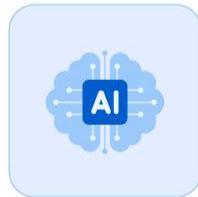
Set a Persona



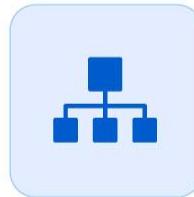
Comprehend the Task at Hand



Remove Ambiguity



Tell AI What Not to Do



Break Down Complex Tasks



Structure Prompts by Priority



Specify the Output Format

Prompt Engineering in SDLC

- In AI-native projects, prompts evolve like software artifacts.
- Managed through version control and collaborative prompt libraries.
- Used in:
 - Requirement elicitation (e.g., “Describe system constraints...”)
 - Testing & QA (e.g., “Find logical errors in this code...”)
 - Deployment (defining system prompt for agents).
- Use tools like GitHub or prompt management systems (e.g., PromptHub, LangSmith).
- Prompts become reusable project assets, like source code or test cases.

Testing and Refining Prompts for Quality Output

- Evaluate prompt performance using:
 - Accuracy: Does output meet task requirements?
 - Consistency: Are results reproducible?
 - Relevance: Does it align with context and constraints?
- Refinement process =
Prompt–Response–Review–Adjust cycle.
- Use feedback loops or automatic scoring tools (BLEU, ROUGE for text).

BLEU and ROUGE score to assess prompt and response quality

- BLEU (Bilingual Evaluation Understudy):
 - Measures precision : how much generated text overlaps with reference text.
 - Common in machine translation and code generation tasks.
 - Higher BLEU → closer match to reference output.
- ROUGE (Recall-Oriented Understudy for Gisting Evaluation):
 - Measures recall : how much of the reference text is captured by generated output.
 - Used for summarization, report generation, and explanation tasks.
 - Higher ROUGE → better content coverage.

Ethical and Responsible Prompting

- Prompts can unintentionally induce bias, hallucination, or data leakage.
- Always validate model outputs against ethical standards and data governance policies.
- Avoid prompts that:
 - Encourage harmful or biased outputs.
 - Reveal private or copyrighted data.
- Part of responsible AI project management.

End of Prompt Engineering