Unlocking the full potential of generative AI starts with mastering communication with generative AI models through effective prompts. This chapter introduces you to the world of prompt management, a cornerstone for crafting intelligent, context-aware, and goal-specific AI responses. It guides you from crafting the ideal initial prompt to refining it, showing that effective prompts can become precise tools.

In a retail chatbot scenario, starting with a simple instruction like Help customers politely can result in unclear answers. By refining the approach—looking at feedback, adjusting the wording, and improving context—the chatbot evolves into a helpful and polite assistant that can manage various customer questions. Likewise, prompts designed for e-commerce recommenders change broad suggestions into personalized recommendations by including specific details.

This chapter emphasizes the importance of thoughtful prompt design, optimization, and version control, as well as the ability to adapt prompts dynamically to meet changing business needs. You will learn how to create, test, and deploy strong prompts. You will also keep versioned documentation for scalability. This will be done using Amazon Bedrock Prompt Management tools.

This chapter provides practical examples and workflows. It helps you improve your generative AI applications. The goal is to create consistent, efficient, and user-focused interactions. This approach encourages innovation in AI-driven experiences.

# **15.1 Overview of Prompt Management**

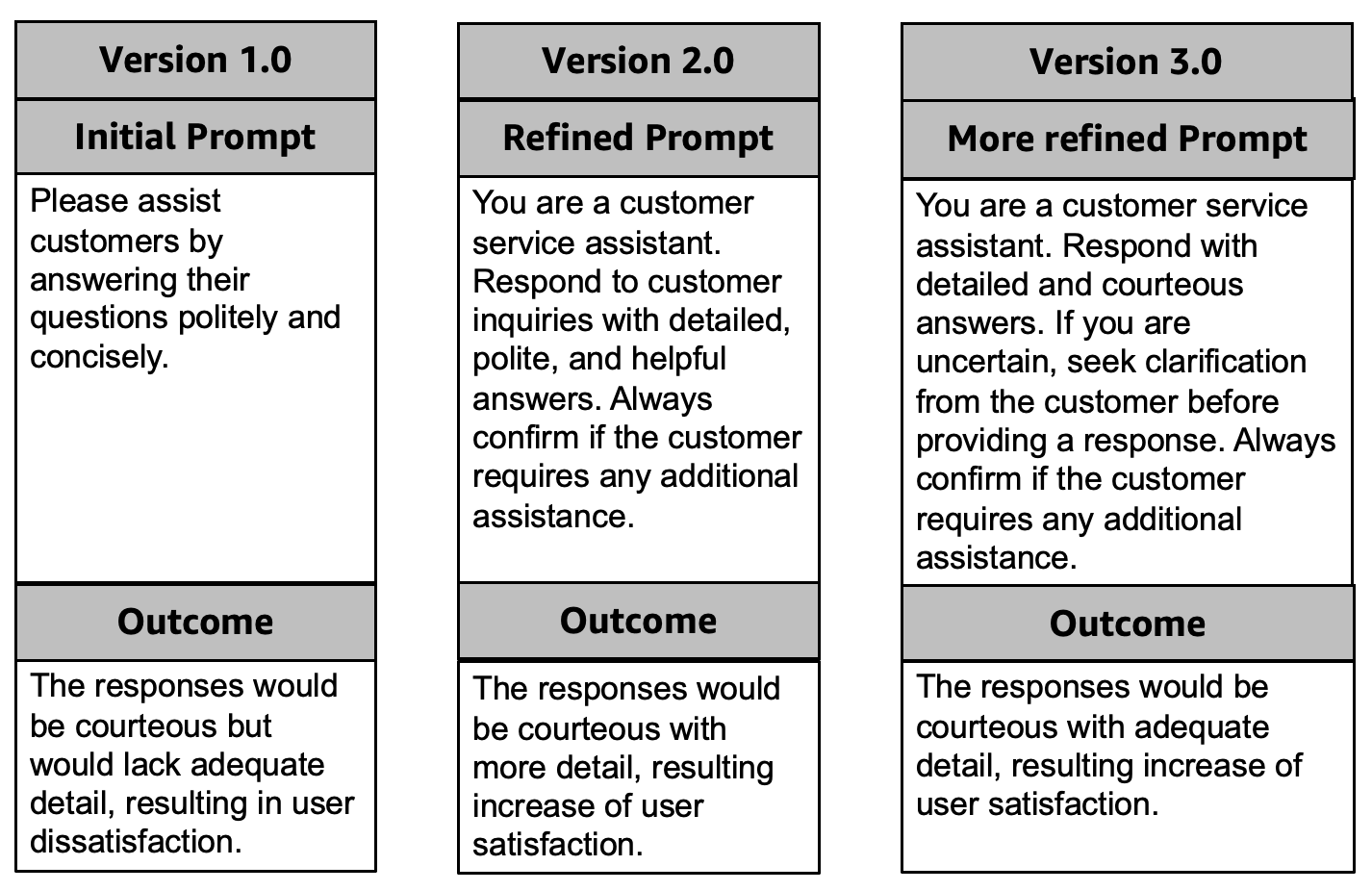
Prompt engineering represents a cutting-edge approach. The generative AI model will produce more accurate results if your prompt is clear. You should start with an initial prompt and refine it continuously based on the use case needs and best practices of prompt engineering. So, powerful prompt management is essential for your generative AI applications. Consider a scenario where a retail company uses an AI virtual chatbot to handle customer service queries. You can begin with an initial prompt (version 1.0) such as "**Please assist customers by answering their questions politely and concisely.**". The responses could be courteous but would lack adequate detail, resulting in user dissatisfaction.

To address these issues, you start refining the prompt by analyzing prompt interactions. By examining metrics such as user satisfaction scores and response length, you identify areas for improvement. By incorporating user feedback, you revised the prompt (version 2.0) like, “**You are a customer service assistant. Respond to customer inquiries with detailed, polite, and helpful answers. Always confirm if the customer requires any additional assistance.**”. You observed that the responses would be courteous with more detail, resulting in an increase in user satisfaction.

You then want to refine the responses to be more courteous and provide adequate detail, which could increase user satisfaction. Therefore, you refined further the prompt (version 3.0) with the limitation of the context window of the generative AI model, like, “**You are a customer service assistant. Respond with detailed and courteous answers. If you are uncertain, seek clarification from the customer before providing a response. Always confirm if the customer requires any additional assistance.**”.

The iterative refinement resulted in notable improvements in customer satisfaction metrics. The virtual chatbot began generating responses that were not only polite and detailed but also adaptive to varying contexts. By maintaining versioned documentation for prompts and responses, you created a robust foundation for scaling and adapting the system in the future.

This example highlights the importance of systematic prompt refinement, demonstrating that the right prompt management can drive continuous improvements in AI performance and user experience.



*Figure 15-1 Example of prompt for a retail virtual chatbot*

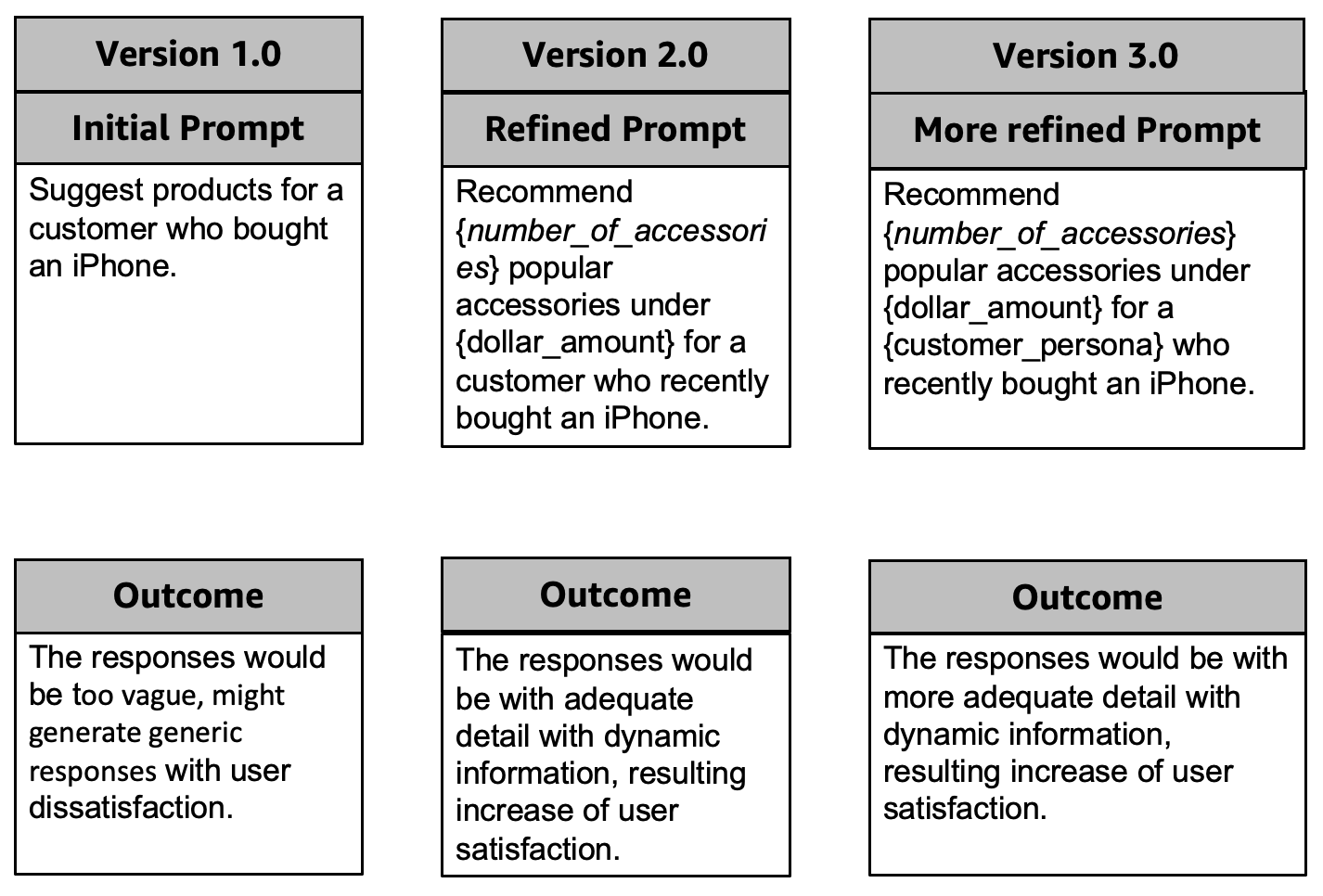
Consider another scenario where an ecommerce company uses an AI virtual chatbot for product recommendations for their customers.

You can start with an initial prompt (version 1.0) such as “**Suggest products for a customer who bought an iPhone.**”. The responses may be too generic, as the prompt itself is too vague.

To address these issues, you start refining the prompt by analyzing prompt interactions. By examining metrics such as user satisfaction scores and response length, you identify areas for improvement. Incorporating user feedback, you revised the prompt (version 2.0) to say, “**Recommend {number\_of\_accessories} popular accessories under {dollar\_amount} for a customer who recently bought an iPhone.**”. You observed that the responses would be clear, specific, and limited to a price based on two parameters: the number of accessories and the dollar amount.

The next step is to provide responses with sufficient detail, which will enhance user satisfaction. So, you refined further the prompt (version 3.0) like, “**Recommend {number\_of\_accessories} popular accessories under {dollar\_amount} for a {customer\_persona} who recently bought an iPhone.**”.

Now, you might want to store both the version 3.0 prompts in a prompt library for future use and refinement based on business changes. The company's virtual chatbot improved customer satisfaction metrics through iterative refinement, generating polite, detailed, and adaptive responses. Maintaining versioned documentation ensured system stability, highlighting the importance of prompt management for generative AI performance and user experience.



*Figure 15-2 Example of prompt for an ecommerce virtual chatbot*

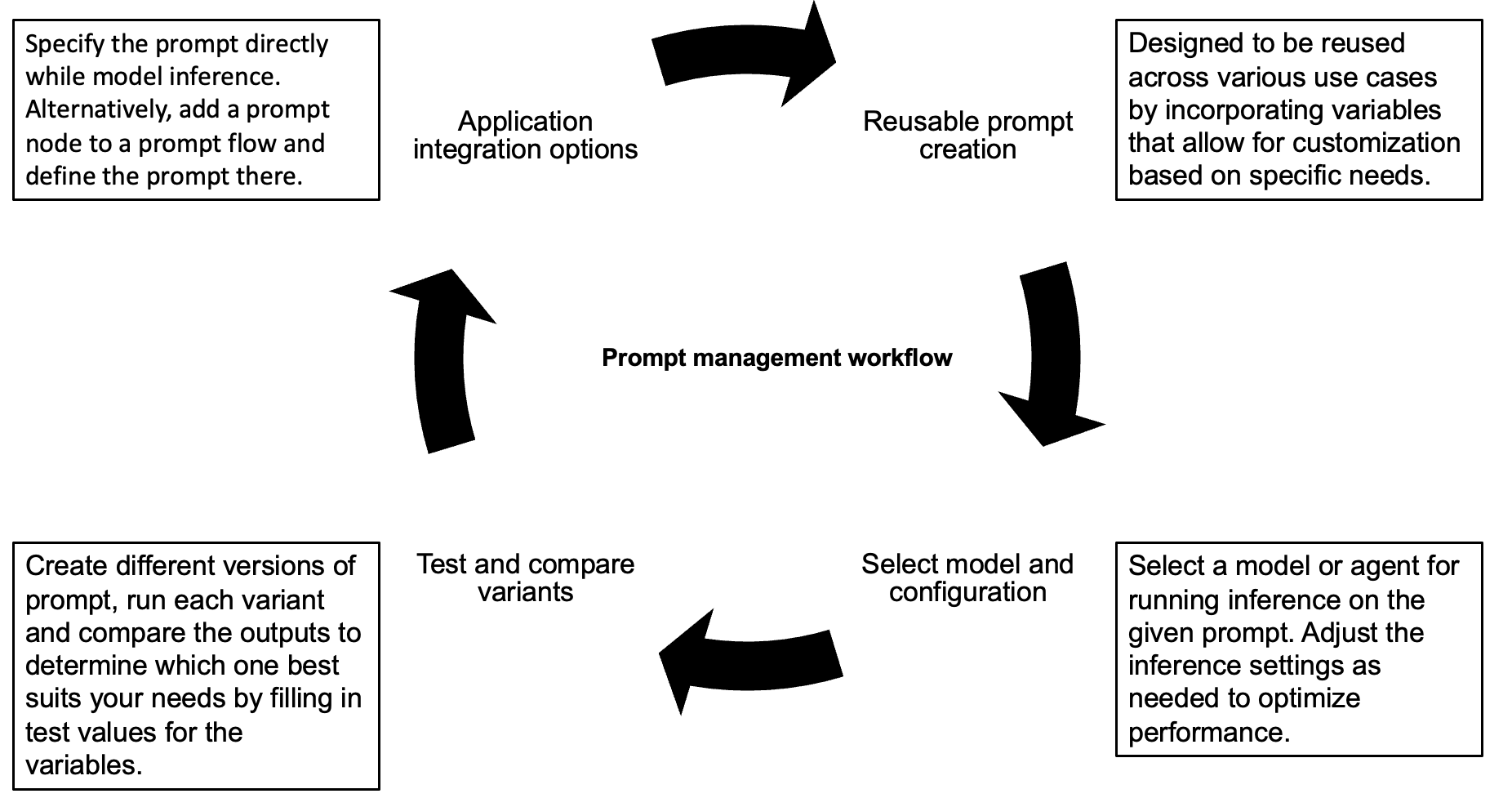
Generative AI prompt management focuses on the organization and optimization of prompts used with AI models. This process aims to achieve consistent, accurate, and goal-specific outputs. By managing prompts effectively, it ensures that models produce high-quality responses, minimize errors, and adapt efficiently to different tasks. Let you drive deep on the key components of prompt management.

* **Design of prompt**: Writing prompts that are clear, specific, and aligned with the task's objective.
* **Optimization of prompt**: Refining prompts through testing and iteration to improve model-generated outputs.
* **Version control**: Keeping track of prompt versions to maintain consistency and manage changes.
* **Dynamic prompting**: Modifying prompts dynamically based on input or context.
* **Monitoring and evaluation**: Measuring the effectiveness of prompts and refining them based on performance metrics.

Let you explore why it's important to have a prompt management strategy when building generative AI applications.

* **Consistency**: Helps maintain uniformity in responses across different use cases or users.
* **Efficiency**: Reduces time spent debugging or refining poorly constructed prompts.
* **Scalability**: Allows organizations to use prompts effectively across a range of applications.
* **Adaptability**: Helps manage diverse user needs and evolving tasks by tailoring prompts.
* **Cost optimization**: Reduces unnecessary compute costs from retries or incorrect outputs.

Amazon Bedrock allows you to create and save custom prompts through its prompt management feature. This functionality helps save time by enabling the application of the same prompt across various workflows. When creating a prompt, you can choose a model for inference and adjust the inference parameters as needed. You can also include variables in the prompt to customize it for various use cases. When testing your prompt, you can compare various versions to find the one that works best for your needs. As you refine your prompt, you have the option to save different versions. To incorporate a prompt into your application, you can use Amazon Bedrock Prompt flows. You will learn Amazon Bedrock Prompt flows in Chapter 16.



*Figure 15-3 Example of a general prompt management workflow*

Since you already understand the definition of a prompt, you should be familiar with three additional important definitions for this chapter.

* **Variable**: A placeholder can be added to prompts for flexibility. It allows for the inclusion of variable values during testing. This feature is useful both in development and at runtime.
* **Prompt variant**: You can modify the prompt's configuration, including its message and model settings. By creating various prompt variants, you can test their effectiveness. Once you've identified a successful variant, you can save it for future use.
* **Prompt builder**: The Amazon Bedrock console features a tool for creating, editing, and testing prompts. This tool offers a visual interface for users to work with different prompt variants.

# **15.2 Lifecycle of Prompt Management**

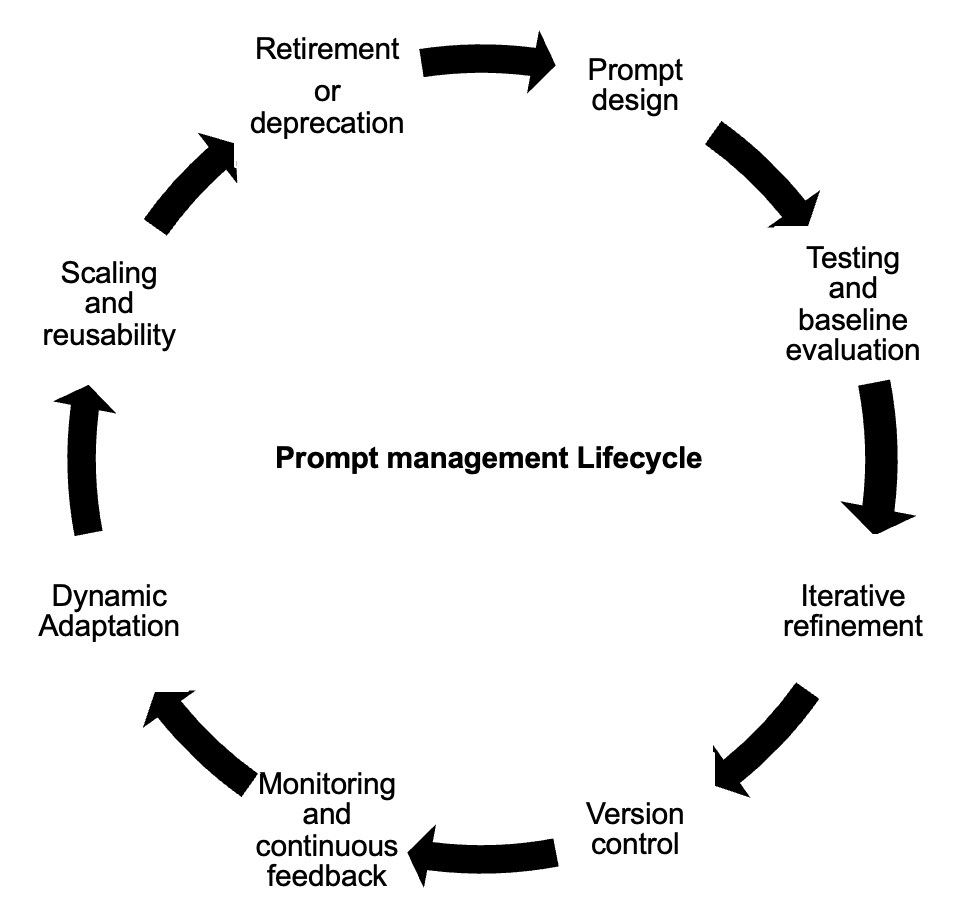
The prompt management lifecycle offers a systematic way to create, optimize, and maintain prompts for generative AI systems. This approach ensures prompts are consistent, efficient, and adaptable, leading to high-quality interactions in various applications. Additionally, it facilitates scalability and minimizes troubleshooting, aligning AI performance with changing user and business requirements.

#### **Prompt Design**

Prompt design is the foundational phase in the prompt management lifecycle. The main goal here is to clearly define the task the prompt is meant to achieve. During this phase, the initial prompt is drafted with a focus on clarity and specificity. It's crucial to collaborate with stakeholders to understand business objectives and ensure the prompt meets requirements. For instance, a prompt for a customer service chatbot could be: "You are a customer service assistant. Provide detailed, polite, and accurate answers to queries." This approach ensures that the virtual chatbot's responses are helpful and align with the company’s customer service standards.

#### **Testing and Baseline Evaluation**

Testing and Baseline Evaluation is the phase for assessing the initial performance of a prompt to ensure it meets objectives. During this phase, tests with sample inputs evaluate how well the prompt generates expected responses. Performance metrics like accuracy, relevance, and user satisfaction are measured to establish a baseline for comparison. This process identifies gaps in responses, such as ambiguity or lack of detail, and addresses specific your needs. By identifying these issues early, refinements can be made to enhance the prompt’s effectiveness before further deployment.



*Figure 15-4 Example of prompt management lifecycle*

#### **Iterative Refinement**

Iterative refinement aims to enhance prompts for better accuracy and relevance in outputs. The process involves gathering user feedback and analysing performance metrics, such as response relevance and task success rates. Insights from this analysis lead to modifications of the prompt to fix any identified issues. For example, a vague prompt like "**Answer customer queries**" can be improved to "**Provide detailed and polite answers. If unsure, ask clarifying questions before responding.**" This ongoing refinement ensures prompts evolve to provide more effective, context-aware, and precise responses.

#### **Version Control**

Version control is essential for managing prompt iterations by tracking changes and ensuring consistency across versions. The goal is to document each prompt version, detailing modifications for transparency and traceability. This process allows for easy auditing and the option to revert to previous versions if needed. Versions can be labelled like v1.0, v2.0, etc., with timestamps and summaries of changes between iterations. This practice helps teams systematically manage prompt evolution while keeping a clear record of adjustments over time.

#### **Monitoring and Continuous Feedback**

Monitoring and Continuous Feedback is essential for the long-term effectiveness of prompts. The goal is to track and evaluate prompt performance over time. This includes using a strategic approach to measure key metrics such as user satisfaction, response time, and errors. Real-time user feedback is also gathered to identify new challenges or areas for improvement. For instance, logs may show that the system has difficulty with ambiguous queries, leading to prompt refinements for better clarity and accuracy. This ongoing process ensures the generative AI's output remains relevant and of high quality.

#### **Dynamic Adaptation**

Dynamic adaptation involves modifying prompts to align with changing requirements and circumstances. The goal is to keep the prompt relevant and effective as situations evolve. This phase introduces contextual elements based on real-time conditions or specific user needs. External data, like user profiles or past interactions, is used to personalize responses for a tailored experience. For instance, when assisting a high-value customer, the prompt may be adjusted to prioritize detailed, personalized assistance. This approach enhances the chatbot's service quality, leading to improved customer satisfaction and engagement.

#### **Scaling and Reusability**

Scaling and reusability emphasize the use of effective prompts in various contexts to enhance their effectiveness. The goal is to generalize successful prompts for adaptability in similar tasks. A significant part of this phase involves developing a prompt library for organizational use, promoting consistency and efficiency. For instance, a successful customer service prompt can be adapted for an internal IT helpdesk chatbot, improving response consistency across departments.

#### **Retirement or Deprecation**

Retirement or deprecation is the last stage in the prompt management lifecycle. This phase aims to phase out outdated or ineffective prompts. The goal is to retire prompts that no longer fulfil their intended purpose or have been superseded by better options. It also includes archiving previous versions for historical reference, analysis, or compliance. By retiring obsolete prompts, organizations can focus on using only the most relevant and efficient prompts. Additionally, maintaining a record of past iterations aids in future insights or regulatory requirements.

All the above phases of the prompt management lifecycle follow a flywheel for continuous improvement based on the industry and use cases.

# **15.3 Sample Application: Building Amazon Bedrock Prompt Management**

To get the GitLab details, refer to the appendix section of this book. In GitLab, locate the repository named **genai-bedrock-book-samples** and click on it.

Inside the **genai-bedrock-book-samples** repository is an AWS CloudFormation template that resides in the **cloudformation** folder. If you already executed the AWS CloudFormation template in Chapter 3 and didn't delete the stack afterward, you can skip the paragraph highlighted in grey below.

The task requires the execution of an AWS CloudFormation template, which should be performed once for all exercises in this book. A detailed guidance on how to manually execute the AWS CloudFormation template can be found in a file called **README** located within a directory named **cloudformation**. For more information about AWS CloudFormation template refer <https://aws.amazon.com/cloudformation/>.

**Disclaimer**: It is advisable to delete the AWS CloudFormation template if you are not actively participating in any exercises for some longer duration. Clear instructions for deleting the AWS CloudFormation template are provided within the README file itself.

However, in the **genai-bedrock-book-samples** folder there’s another subfolder titled **chapter15**. The **README** file within **chapter15** folder provides clear instructions on launching a **Notebook** on Amazon SageMaker.

|  |  |
| --- | --- |
| **File Name** | **File Description** |
| simple\_prompt\_mgmt.ipynb | 1. Create a prompt using Prompt management 2. Modify a prompt using Prompt management 3. Create a version of a prompt in Prompt management 4. Retrieve detail of prompt 5. Testing the prompt with simple way 6. Delete a version of a prompt in Prompt management   **Dependency**:  simple-sageMaker-bedrock.ipynb at Chapter 3 should work properly. |

# 3.8 Bedrock Interaction Sample Application

**Disclaimer**: Charges will apply upon executing above files. Therefore, it is important not to forget to clean up the kernel after studying the topic. Refer to the clean-up section for instructions on how to properly clean up the kernel.

# **15.4 Creating a Prompt**

You can create prompts using prompt management in Amazon Bedrock. This serve as flexible templates for working with generative AI models. These prompts are structured inputs that help direct how the model responds. You can customize them by using dynamic placeholders, which are indicated with double curly braces (like {{variable}}). When you set up a prompt, you can specify the message, choose a model or agent for inference, and tweak settings such as temperature and top-p sampling to influence creativity and diversity. For models that work with the Converse API, you can enhance the prompt with system instructions, conversation history, and tools for better responses.

You initially create prompts as drafts using either the Bedrock Console or the API. In the console, you can name the prompt, add descriptions, and optionally encrypt it with a customer-managed KMS key. The user-friendly Prompt Builder helps configure templates and test different variants. Using the API, the CreatePrompt request allows you to define fields such as names, variants, and template configuration, resulting in a draft prompt with a unique ARN and ID. You can also add optional fields for tagging, descriptions, and ensuring idempotency.

You can configure, test, and compare prompts across variants after creation to refine outputs. By leveraging Bedrock’s robust prompt management capabilities, developers can streamline the integration of advanced generative AI features into their applications.

You will find one example notebook in section 15.3. You can also navigate the AWS documentation for the details. Refer to

https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-management-create.html.

# **15.5 Testing a Prompt**

Testing a prompt is essential to make sure it works well before using it in production. Amazon Bedrock provides various ways to test prompts using prompt management through the AWS Management Console and APIs. In the console, you can go to the Prompt Management section, pick or modify a prompt, and enter temporary test values for the variables. You can choose a model configuration and test it in the Test Window Pane. You can even refine your prompts until you are satisfied and meet the business context. Furthermore, you can create a version snapshot for easier integration into production workflows.

You can test APIs with Bedrock's runtime endpoints. You can send requests like InvokeModel, InvokeModelWithResponseStream, or Converse to test a prompt. Just specify the ARN in the modelId parameter to test a prompt directly. You can also test prompts in a prompt flow. This involves embedding prompts in nodes using APIs like CreateFlow or UpdateFlow, and then running them with InvokeFlow. Furthermore, you can use the console or InvokeAgent requests to test with agents, allowing for dynamic and scalable prompt evaluation. However, Chapter 16 contains a specifically designed notebook for testing prompts with prompt flows. These methods ensure that the prompt meets business objectives, aligns with configurations, and is ready for deployment.   
You will find one example notebook in section 15.3. You can also navigate the AWS documentation for the details. Refer to

<https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-management-test.html>.

# **15.6 Managing a Prompt**

Modifying a prompt is important for managing prompts. It helps you adjust them to meet changing business needs or user feedback. With the Amazon Bedrock console, users can easily change a prompt's name, description, or settings using a user-friendly interface. To make changes to a prompt, you go to the Prompt Management section. Select the prompt and corresponding version you want to edit. You can modify the overview and detailed settings with the prompt builder. If you need to update prompts programmatically, you can use the UpdatePrompt API for modifications. You can specify the fields to maintain or alter, ensuring precise control over prompt behaviour, by sending a request to an Amazon Bedrock runtime endpoint. This dual approach, console and API, ensures flexibility and scalability in managing prompts, whether for minor edits or significant updates to configuration settings.

You will find one example notebook in section 15.3. You can also navigate the AWS documentation for the details. Refer to

https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-management-modify.html.

# **15.7 Deploying a Prompt**

Amazon Bedrock allows for reliable and flexible integration of generative AI into applications through prompt deployment. Deployment starts with a draft prompt. This allows you for adjustments to be made to fit particular use cases. After finalizing the prompt, you create a version, capturing a snapshot of the draft at that point. This versioning system makes it easy to manage different configurations and switch between versions as needed for various applications.  
Using the Amazon Bedrock console, you can create prompt versions by navigating to the "Prompt Management" section, selecting the draft, and choosing Create Version. Furthermore, the CreatePromptVersion API allows programmatic creation by specifying the prompt's identifier, returning the version's ID and ARN.

Versioning allows for transparency and control. You can use the console or GetPrompt API to see, compare, and delete versions. Comparing versions side-by-side shows configuration differences, which helps with optimization. Once deployed, prompts integrate into applications via the InvokeModel API, using the version's ARN for inference requests.

Version control ensures applications use the most appropriate prompt configuration while enabling iterative enhancements and compliance tracking. This structured deployment approach simplifies scaling and maintains high-quality AI interactions.

You will find one example notebook in section 15.3. You can also navigate the AWS documentation for the details. Refer to

<https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-management-deploy.html>.

Even, you can compare between two versions of the same prompt or two different prompts with the above strategy. Refer to

<https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-management-version-compare.html>

# **15.8 Summary**

You will learn how prompt management is an important strategy for improving generative AI applications. It will discuss how clear and iterative prompt engineering can significantly enhance the performance of AI models. Using relatable examples such as virtual chatbots in retail and e-commerce, it illustrates how refining prompts boosts user satisfaction, response quality, and task-specific accuracy. You will learn about the prompt management lifecycle. This includes important steps like creating prompts that align with goals, testing them to set performance standards, improving them using user feedback, keeping track of changes with version control, and adjusting them as needs change. It highlights the importance of adjusting prompts for different situations. It also suggests getting rid of old prompts to maintain efficiency and relevance.   
You will explore the role of Amazon Bedrock prompt management in this lifecycle in detail, showcasing features like prompt builders, versioning tools, and testing APIs that streamline prompt creation, management, and deployment. By incorporating best practices such as versioned documentation, dynamic placeholders, and prompt flows, organizations can achieve greater consistency, efficiency, and cost-effectiveness in their AI systems.