# Semantic Kernel Part 4: Agents

This series started with the core concepts of Semantic Kernel (SK), then presented some hands-on coding, followed by more advanced examples, including how to implement a basic RAG pattern and automatic function calling. At each step, I discussed how SK treats artifacts like prompts as source code and how each feature builds on those that came before. In this installment, I’m going to cover agents, an emerging approach to accomplishing complex tasks and you’ll see how they build on what came before them and how they can be treated as source code.

## Why Agents?

With RAG and automatic function calling, you can do some really amazing and advanced things. So, what do agents add and why should you learn about them? An agent is nothing more than a customized Large Language Model (LLM) that may also have access to some specific capabilities and/or data. Creating an agent could be as simple as prefacing prompts to an LLM with something like, “You are a helpful assistant who ensures that math calculations are done correctly and produces a correct answer.” By providing this customization, you’re narrowing the responsibilities of the LLM and creating an expert of sorts in a certain area. An assistant created this way will, in fact, be much better at producing correct results when math is involved.

Semantic Kernel calls this type of agent an assistant agent. It’s one of two types of agents that SK supports. You might be thinking that you could just add some verbiage like that into your prompt and get the same result, and you’d be right. Remember that I said agents can also be given access to specific capabilities? What if I allowed my agent to have access to MATLAB, or gave it the ability to create and execute Python by giving it access to functions? That would make the agent even better and more accurate with math.

Yes, you could also do this without an assistant, but you can start to see how complex things can get with this approach. For example, if it looks like I need the LLM to do math, I must add this blurb of instructions to my prompt and allow access to the MATLAB, Python creation, and execution functions. And you must follow this pattern for every specialty skill you might need to use, every time you need to use it. Following this approach, things will get messy, fast.

## Assistant Agents

Agents bring some sanity by allowing you to create pre-configured experts and treat them as assets. The expert is configured with specific instructions, access to specific functions and capabilities, and perhaps even access to specific knowledge. Then you can simply use the agent when you need to. For example, if the agent isn’t supposed to just be good at math, but also able to calculate New Jersey state income taxes, you could, in addition to giving it access to MATLAB and Python execution, make a series of tax tables available to it and provide it with certain formulas.

Pre-configuring agents like this makes them modular and easy to use. To use the NJ state income tax assistant agent, you only need to specify to use it in your SK code; you don’t have to create it from scratch, configure it, and provide the tax tables and formulas every time you need to use it. Imagine creating dozens, or even hundreds of such specialized agents, and only having to decide which one to use, and not how to re-create, configure and test it in every instance. A developer or team could be assigned responsibility for the agent and could handle its enhancement, its testing, etc. The agent can be treated like source code, much the same way you treat a software library.

Let’s create a very basic assistant agent with Semantic Kernel and then use it. Agents are still in alpha as I’m writing this, so you’ll likely have to check the **Include prerelease** checkbox in VS’s NuGet Package Manager to see them. I’m using version 1.25.0-alpha of the Microsoft.SemanticKernel.Agents.Core and Microsoft.SemanticKernel.Agents.OpenAI packages in this article. There may be breaking changes in later versions. The former provides abstractions for using agents and the latter is a concrete implementation that allows me to use them on OpenAI and Azure OpenAI. If you use Azure OpenAI as I am, ensure that the region your model is deployed in supports agents or you’ll get errors about invalid endpoints (the agent endpoints). There are other implementations for other platforms, and I expect more to be added as agent functionality nears release.

Now you can create your first agent definition. To keep things simple, I’m only going to provide instructions, ignoring function calling and custom data for now. My prompt will create an assistant agent that writes songs in the style of Bob Dylan.

By providing this customization, you’re narrowing the responsibilities of the LLM and creating an expert of sorts in a certain area.

[AI Query]

You are Bob Dylan, the famous songwriter and musician. Write a song about the provided topic in the style of a folk song.

Now I’ll create my agent:

await OpenAIAssistantAgent.CreateAsync(

 kernel: kernel,

 clientProvider: provider,

 definition: new(\_deployment)

 {

 Name = "BobDylan",

 Instructions = bob

 });

**Listing 1** shows the complete source code for this example.

I can use my agent by creating a thread on the agent, priming it with a user prompt, and handling the response like this:

string threadId = await agent.CreateThreadAsync();

await agent.AddChatMessageAsync(threadId,

 new ChatMessageContent(AuthorRole.User,

 "waiting for a phone call"));

await foreach (ChatMessageContent message

 in agent.InvokeAsync(threadId))

 Console.WriteLine(message);

Sample output using GPT-4o looks like this:

[AI Response]

(Verse 1)

In a room dim-lit, I sit and wait,

For a ring from a friend or a lover's fate,

The silence, oh it beckons, the world stands still,

As my heart beats on, against its will.

(Chorus)

Oh, I'm waiting for a phone call, seems like forever,

Through the night and the stormy weather,

Words unspoken, dreams on hold,

Stories untold, oh, the nights grow old.

(Verse 2)

The clock hands sway, in their timeless dance,

Holding onto hope, midst the sweet expanse,

Of memories we shared, and dreams we weave,

In this tapestry, of love and grief.

(Chorus)

Oh, I'm waiting for a phone call, seems like forever,

Through the night and the stormy weather,

Words unspoken, dreams on hold,

Stories untold, oh, the nights grow cold.

(Bridge)

Will the voice on the line, bring joy or pain?

Will it ease my mind, or drive me insane?

In this solitude, where shadows creep,

In the echo of silence, promises sleep.

(Verse 3)

Bound by the yearning, of a heart so true,

In a world divided, caught in two,

I'll hold this vigil, till the dawn's embrace,

Hoping to hear, your tender grace.

(Chorus)

Oh, I'm waiting for a phone call, seems like forever,

Through the night and the stormy weather,

Words unspoken, dreams on hold,

Stories untold, oh, the nights grow bold.

(Outro)

So I sit and ponder, this love so deep,

And in the quiet, my patience keep,

For one small chime, a lifeline, dear,

To know you're out there, to bring you near.

As you can see, having an arsenal of pre-configured assistant agents can be a powerful tool.

## Chat Agents

But wait, there’s more! You can also create an agent that can interact with other agents to work collaboratively. That’s the idea behind chat agents, the other type of agents SK supports. Here’s one of the scenarios Microsoft uses as an example of chat agents. Suppose you want your AI to write software for you. Can it really do that? Yes! And no! Right now, it can write some basic software, but you’re not out of a job, at least not yet. In this example, I’ll ask the AI to create a simple calculator as a web application. First, you’ll create three specialized agents, a program manager, a software engineer, and a project manager with the following instructions:

[AI Query]

You are a Program Manager who will take the user requirements and create a plan for creating an app. The Program Manager understands the user requirements and will form detailed documents with requirements and costs.

[AI Query]

You are a Software Engineer, and your goal is develop a prototype web app using HTML and JavaScript (JS) by taking into consideration all the requirements from the Program Manager.

[AI Query]

You are a project manager who will review software engineer code, and make sure all client requirements are completed. Once all client requirements are completed, you can approve the request by just responding "approve".

Again, I’m creating a very basic scenario. This example has been tweaked from Microsoft’s original sample code to make it a bit more reliable and easier to digest. I could also give these agents access to functions and data to make them more effective and powerful, but they do surprisingly well without it. Notice that the project manager instructions end by telling it to respond with only the word “approve” when it’s satisfied with the results. You’ll see why that’s important soon.

You can also create an agent that can interact with other agents to work collaboratively.

Similar to how you created the assistant agent earlier, begin by creating the three chat agents:

ChatCompletionAgent ProgramManagerAgent

 = new()

 {

 Instructions = ProgamManager,

 Name = "ProgramManagerAgent",

 Kernel = kernel,

 };

ChatCompletionAgent SoftwareEngineerAgent

 = new()

 {

 Instructions = SoftwareEngineer,

 Name = "SoftwareEngineerAgent",

 Kernel = kernel

 };

ChatCompletionAgent ProjectManagerAgent

 = new()

 {

 Instructions = ProjectManager,

 Name = "ProjectManagerAgent",

 Kernel = kernel

 };

Next, create a new agent group chat among all the agents. If you were to just turn the agents loose with a prompt from the user, they might just end up chatting forever, which could be very expensive and frustrating for the user. So, before you start the chat, let’s put some restrictions on it and specify a termination strategy.

AgentGroupChat chat =

 new(ProgramManagerAgent,

 SoftwareEngineerAgent,

 ProjectManagerAgent)

{

 ExecutionSettings = new()

 {

 TerminationStrategy =

 new ApprovalTerminationStrategy()

 {

 Agents = [ProjectManagerAgent],

 MaximumIterations = 10,

 }

 }

};

Notice that you specify the agents to use in the constructor. You then set some ExecutionSettings. In this case, you only set the Termination Strategy by specifying that the ProjectManagerAgent will decide when the chat is done and you set MaximumIterations to 10, meaning that if you don’t get an approval after 10 attempts, go ahead and stop anyway.

Although you don’t explicitly create a thread for the chat like you did with the assistant agent, you do something very similar and the chat object handles the thread for you.

chat.AddChatMessage(

 new ChatMessageContent(AuthorRole.User, input));

await foreach (var content in chat.InvokeAsync())

 Console.WriteLine(

 $"\n# {content.Role} –

 {content.AuthorName ?? "\*"}:

 '{content.Content}'");

In this case, you’re writing not just the final result, but the entire conversation among the three agents to the console so you can see how the agents collaborate to produce an answer. Unfortunately, the output is too long to include here. I’ve included it with the sample code (downloadable from the article page at [www.CODEMagazine.com](http://www.codemagazine.com/)), as well as a sample HTML page, complete with CSS and JavaScript produced by one of my test runs against GPT-4o that runs as expected. For now, I’ll just recap the conversation using the following user prompt. You can find the full code for this sample in the source code that accompanies this article.

[AI Query]

I want to develop a calculator as a web app. Keep it very simple, and get final approval from the project manager.

The Program Manager begins by responding to the user and creating a set of requirements for the app. For instance, under Functional Requirements, it writes:

[AI Response]

1. \*\*Basic Arithmetic Operations\*\*: The calculator should perform addition, subtraction, multiplication, and division.

It goes on to define project milestones and even estimates development costs. It then asks the Software Engineer agent if it needs any additional information and asks it to approve the requirements. The Software Engineer agent approves the requirements, creates a wireframe and mockup of the application, then asks the Project Manager if any changes are needed, and asks for approval. The Project Manager creates a checklist of the requirements, compares it to the prototype code, decides that the prototype meets the requirements, and approves, terminating the chat.

The chat output is never exactly the same after each run. Sometimes there are missing items or the need for changes detected, resulting in one agent going back to a previous agent to have it address the issues, and resulting in a longer conversation. What’s important is that the agents communicate among themselves to reach the goal.

What’s amazing to me is that with very little configuration and no access to functions or specialized data, the system generally does a very good job of developing the software asked for, as long as the request is basic. The requirements aren’t always complete and correct, and code doesn’t always run as written, but often, it does work. Even though this sample is far from perfect, it produces some truly useful output. One of the reasons for how well this particular example works is that writing a calculator app as a web page with HTML, CSS, and JavaScript is a pretty common programming assignment, so there’s a lot of source code for it, freely available on the web, to draw from.

The more complex and unique the ask, the more advanced the agents will have to become, and the more you’ll have to provide access to your own source code so the LLM can take advantage of it. In a larger context, you might also include agents to perform additional tasks, such as QA and testing.

Agents are a higher-level building block for you to use.

## Summary

As you can see, agents allow you to create larger and more complex systems while remaining somewhat sane. They allow you to abstract systems and build in a modular way. Agents can be quite powerful and they make expanding the capabilities of a system easier to manage. Agents are a higher-level building block for you to use.

Sponsored Sidebar

AI Executive Briefing

Experience the game-changing impact of **AI** through **CODE Consulting's** Executive Briefing service. Uncover the immense potential and wide-ranging benefits of **AI** in every industry. Our briefing provides **strategic guidance** for seamless implementation, covering crucial aspects such as infrastructure, talent acquisition, and leadership.

Discover how to effectively **integrate AI** and propel your organization into future success.

**Contact us today** to schedule your executive briefing and embark on a journey of AI-powered growth: [www.codemag.com/ai](http://www.codemag.com/ai)

Listing 1: Complete source code for assistant agent sample. (Bob Dylan)

private static async Task AssistantAgents()

{

 var builder = Kernel.CreateBuilder()

 .AddAzureOpenAIChatCompletion(

 \_defaultDeployment,

 \_endpoint,

 \_apiKey);

 var kernel = builder.Build();

 var bob = """

 You are Bob Dylan, the famous songwrite and musician.

 Write a song about the provided topic in the style

 of a folk song.

 """;

 var agent =

 await OpenAIAssistantAgent.CreateAsync(

 kernel: kernel,

 clientProvider: clientProvider,

 definition: new(\_defaultDeployment)

 {

 Name = "BobDylan",

 Instructions = bob,

 EnableCodeInterpreter = true,

 });

 string threadId = await agent.CreateThreadAsync();

 await agent.AddChatMessageAsync(threadId,

 new ChatMessageContent(AuthorRole.User,

 "waiting for a phone call"));

 await foreach (ChatMessageContent message in

 agent.InvokeAsync(threadId))

 Console.WriteLine(message);

}