ISSN: 2473-4901
v10 n6126

Generative Conversational AI: Design AI Dialogue Systems Using Object-Oriented Methodology

Thuan L Nguyen
Advanced Data Analytics – Toulouse Graduate School
The University of North Texas
Denton, Texas, USA

Abstract

While still an emerging technology, natural language processing for generative AI has significantly progressed in developing AI dialogue systems that can serve various applications, showing an increasingly sophisticated natural language understanding. In this paper, to help designers and developers design and build these systems efficiently and effectively, the author proposes a new approach based on the Speech Act Theory and using the popular Object-Oriented methodology. The new approach aims to design dialogue flows as object-oriented classes. Each dialogue flow consists of data (entities and entity types) and intents (or actions), similar to attributes (data: what it has) and methods (or behaviors: what it can do) of object-oriented classes. The design focuses on these two significant parts of each dialogue flow: entities - entity types and intents (or actions). Then, the paper demonstrates how to use the new approach to design one of the main dialogue flows of a simple real-world conversational system for a retailer. Finally, the paper shows how to use Google's Dialogflow CX to build the designed conversational system, focusing on this dialogue flow, and test it.

Keywords: Generative Conversational AI, AI Dialogue System, Generative AI Agent, Design AI Dialogue Systems, Build Generative AI Agents, Object-Oriented Methodology

1. INTRODUCTION

Conversational AI, powered by large language models (LLMs), enables natural human-computer interaction, which led to significant progress, including generating human-like text. Popular LLMs, like those behind ChatGPT, Claude, and Gemini, show impressive language abilities (Wang & Singh, 2023). These pre-trained models also make chatbot and conversational agent development simple and convenient. Conversational AI can now have broad impacts on business, healthcare, education, and entertainment.

Various considerations are crucial in AI dialogue system development (Wang & Singh, 2023). Developers must prevent models that perpetuate uncontrolled conversations and disinformation (Lucy & Bamman, 2021; Shah et al., 2019).

The aim is to create sound AI dialogues that foster connection and empathy, which requires safe and effective conversation management. This paper presents a new method for designing business-oriented AI dialogue systems using object-oriented (OO) methodology and the Speech Act theory.

For a conversational system, the new approach proposes that the structure of a dialogue flow consists of two parts: entities – entity types (data) and intents (actions), which are similar to attributes (data: what it has) and methods (behaviors: what it can do) of an OO class.

The paper only focuses on designing these core components of a dialogue flow: Entities - entity types and intents (or actions) As a result, the discussion does not include some other aspects of the OO design process like use case analysis. The paper demonstrates how to design one of the main dialogue flows of a simple real-world conversational system for a retailer, build, and test it using Google's Dialogflow CX.

ISSN: 2473-4901
v10 n6126

2. SPEECH ACT THEORY

British philosopher John Austin's book "How to Do Things with Words" revolutionized language understanding, emphasizing its power in actions and relationships (Austin, 1962). Speech Act Theory's core idea is that every utterance performs three acts:

1. **Locutionary act:** The act of saying something, with its literal meaning.
2. **Illocutionary act:** The intended action behind the utterance (request, promise, and more).
3. **Perlocutionary act:** The actual effect on the listener or reader, which may not match the intention.

Speech Act Theory's insights are crucial in conversational AI and dialogue systems for intent recognition, response generation, dialogue management, and contextual understanding (Mambron, 2020; Sabisa, 2014; Stanford Encyclopedia of Philosophy, 2020).

3. OBJECT-ORIENTED METHODOLOGY

OO Classes: Data and Behaviors
In object-oriented programming (OOP), a class acts as a blueprint for creating objects. Objects and classes are the fundamental building blocks of OOP systems thanks to the transformative combination of data (attributes) and behavior (methods) within objects (Booch, 1994; Stroustrup, 1991).

- **Data (Attributes):** Representing the "What Does It Have?", attributes define an object's characteristics and information. For example, a "Customer" object might have attributes like name, address, email, and order history. Attributes determine an object's state.

- **Behaviors (Methods):** Representing the "What Can It Do?", methods define an object's actions and behaviors. A "Customer" object might have methods for placing orders, modifying orders, or viewing order history.

The strength of OOP lies in this combination of data and behaviors within a class. This provides a strong foundation for creating reliable, maintainable, and scalable software.

4. AI DIALOGUE SYSTEM: MAIN CONCEPTS

AI Agent
An AI dialogue system can be designed and implemented as an AI Agent to communicate with the end-user using natural languages. The system or AI agent can be compared to a human representative in a business's call center or customer service department.

Dialogue Flows: Overview
In the real world, an AI dialogue system must be able to handle many types of conversations for different purposes and goals, each involving a specific topic. A simple conversation of several utterances can handle the greetings. However, a dialogue for a customer to ask about or buy a car is much more complex. A complete conversation typically consists of multiple dialogue flows. Dialogue flows are the building blocks of an AI dialogue system.

For example, a conversation between a customer who calls a retailer to ask about and buy some clothes:

1. Customer: Hello, good morning!
2. Sales representative: Hello, thanks for calling. What can help you?
3. Customer: I want to ask about some T-shirts. Are there any sizes medium and for men?
4. Sales representative: Sure! We have many. How about color? What color do you want?
5. Customer: Green or orange.
6. Sales representative: Yes. We have.
7. Customer: I want to buy two of them, in green, please.
8. Sales representative: You got them. To confirm, you want to buy two green T-shirts, for men, medium size. Is that correct?
9. Customer: Yes. … (the customer pays for the T-shirts) …
10. Sales representative: What else can I help you?
11. Customer: That is all. Thanks. Bye.
12. Sales representative: Thanks very much for the business. Bye.

In the above conversation, there are three different dialogue flows, or at least two:

- (1) – (2): Dialogue flow for initial greetings
- (3) – (9): Main dialogue flow for the customer to ask about and buy clothes

- (10) – (12): Dialogue flow to conclude the dialogue (NOTES: This can be merged with (3) – (9))

Entities and Entity Types: Overview
In AI dialogue systems, "entities" can represent the data associated with an intent. For example, given the utterance "I want to buy two green T-shirts," the intent is "to buy some clothes," and the data or entities are (1): Number of T-shirts: Two; (2): Product: T-Shirt; (3): Colors: Green.
In OO programming, there are two groups of data types:

- System or built-in data types such as Integer or Floating-point offered by default by the system, e.g., a compiler or an IDE (Integrated Development Environment).
- User-defined data types defined by the user.

Similarly, to design and develop an AI Dialogue System, both system-built-in and user-defined entities and entity types are needed.

- System or built-in entities and entity types offered by default by AI conversational development systems, such as Google Dialogflow CX.
- User-defined entities and entity types defined by designers or developers as necessary to create an AI dialogue system.

For entity types, the author proposes **five categories of entity types:** (1) Basic entity types, (2) Complex entity types, (3) Enumerated List, (4) Enumerated List Map, and (5) Struct Composite.

Basic Entity Types
Like primitive or primary data types in popular OO programming languages, the paper proposes a category of basic entity types that includes all commonly used entity types, such as integers and floating-point, to name a few. The basic entity types are often system-defined. These entity types may include the following ones (but are not limited to): **Number** (any numeric values: cardinal, ordinal, integer, float, fractions, iterations, …), Integer, Float, Cardinal, Ordinal, Fraction, Iterations (e.g., once, twice, three times, …), Phone numbers (e.g., 1234567890 …).

Complex Entity Types
The following complex entity types can be introduced and used to design and develop an AI dialogue system: **Any** (any numbers, texts, characters, any entity), **String** (any length, including alphanumeric ones, similar to strings in Java or C++), **Color**, **Date and Time**, **Duration** (number + duration units like seconds, minutes, or hours, …), **Percentage** (%) (Number + percentage Units - percent), **Temperature** (Number + Celsius / Fahrenheit degrees), **Currency** (Number + currency name (e.g., $100.00, 100 Euros)).

Enumerated List Entity Types
This paper proposes a new category of entity type, **Enumerated List** (**Enum-List)**. The **Enum-List** entity type combines common properties of the "**enum**" data type in Java/C++ and "**list**" in Python.

Most Enum-List entity types are user-defined, although some can be defined and supported by the system by default. An Enum-List entity type can be defined as a list of values of which each is an entity name, as follows:

Enum-List <entity type name> = { value 1, value 2, … }

For example:
Enum-List Clothes {Shirt, T-shirt, Pants, Skirt, Hat, Shoes, Jacket, Sweater, …}

Enumerated-List Map Entity Types
This paper proposes another new category of entity type, **Enumerated List Map (Enum-List Map)**. The Enum-List Map entity type has all the features and properties of the Enum-List entity type. The only difference is that each value of the Enum-List Map entity type is mapped to another list of values each of which is also an entity name. An Enum-List Map entity type can be defined as follows:

```
Enum-List Map <entity type name> = {
value 1 → sub-value 11, sub-value 12, …;
value 2 → sub-value 21, sub-value 22, …;
                    …
}
```

For example:

```
Enum-List-Map Size {
        S → Small, Little
        M → Medium, Average, Middle;
        L → Large, Big, Grand;
}
```

Struct Composite Entity Types
This paper proposes another new category of entity type, the **Struct Composite**. This entity type has features and properties like the **structure data type** in **C++**. Struct Composite entity types are likely defined by the user, i.e.,

user-defined entity types. However, a system can define some by default for the user's convenience.

For example:

```
Struct Composite Address {
        Integer streetNumber;
        String streetName;
        String cityName;
        String stateName;
        String zip-code;
}
```

Intents: Overview

A meaningful conversation requires at least an initial intent or intention to start (Austin, 1962; Searle, 1969; Searle, 1983; Henderson & Brown, 1997). Without any intent, meaningful conversations cannot begin. Furthermore, the **flow of intents** or intentions, i.e., **illocutionary force** in his Speech Act Theory, drives conversations forward. When this flow stops, so does the dialogue.

In essence, intent or intention in each utterance is the driving force for successful conversations. Therefore, handling intents or intentions at each turn should be the top priority when designing AI dialogue systems.

Utterances: Data and Intents

Some utterances in dialogue only have intents, or actions, without data, like "Let's go!" However, most conversations, especially in business, require utterances with both intent and data.

For instance, a customer saying "I want to buy..." to a retailer is incomplete without specifying what to buy. A complete utterance combines an intent with the data on which the action is performed.

Routine Dialogue Flows

An AI dialogue system must handle some routine dialogue flows that are always needed in any dialogue. For example, we always start our dialogues with some greetings that can be grouped into a routine dialogue flow named Default Greetings Flow. Another type of routine dialogue flow can be the one to complete and end a dialogue.

Usually, a routine dialogue flow only has intents, i.e., actions, but no data, i.e., entities and entity types. Therefore, a **routine flow** for conversational AI can be viewed as an **interface** with only **methods**, i.e., behaviors or actions, in an OO programming language like Java.

Data-Intent Dialogue Flows

In addition to the routine dialogue flows, a complete significant conversation must include another type of dialogue flow, data-intent dialogue flows. Each data-intent flow handles a specific topic and may be needed to fulfill a system requirement. For example, an AI dialogue system for an online shopping store may need to fulfill conversational requirements for three topics:

1. **Products-Info flow**: For customers to get information on some products
2. **Order-Process flow**: For customers to buy/order some products
3. **Customer-Services flow**: For customers to get help with orders, canceling, and returning products

A data-intent dialogue flow has intents and data, i.e., entities and entity types. These flows can be matched to OO classes with attributes (data) and methods (actions or behaviors) in OO programming languages like Java or C++. In special cases, a data-intent flow can have only data but without any intents.

Data-Intent Flows: Intents

In object-oriented programming languages such as Java or C++, a class combines two major components: data (attributes) and behaviors (methods). As discussed above, similarly, a dialogue flow can have data, i.e., entities and entity types, and intents (actions). As a result, the author proposes to mimic the structures of methods in an OO class to design intents belonging to a data-intent dialogue flow of an AI dialogue system.

Data-Intent Flows: Intents: Signature

In a Java or C++ class, a method has a signature consisting of a returned data type, one or more optional modifiers, the method name, and a list of parameters embedded between a pair of parentheses. Each parameter is a variable defined by its data type.

The author proposes an intent signature that consists of an access modifier ("public," "protected," or "private"), the intent name, and a list of parameters embedded between a pair of parentheses. Similarly, each parameter is an entity defined by its associated entity type. An intent signature is slightly different from that of an OO class method: An intent signature does not have a returned entity type where an OO class method has a returned data type.

The signature of an intent can be defined as follows:

<Optional Modifier> <Intent Name> (… List of entities with entity types …);

For example, the signature of a customer's intent to buy shirts of some color, size, and group (men, women, …) can be defined and designed as follows:

```
order (Product product, Number numItems, Color color, Size size, Group group);
```

Data-Intent Flows: Intents: Definition
A method of an OO class is defined with its signature and a body of lines of code. Similarly, the complete structure of an intent definition consists of its signature and a body that contains training phrases used to train the AI dialogue system to learn how to detect, match, and fulfill the intent.

```
<Optional Modifier> <Intent Name> (… List of entities …) {
    Training phrase 1
    Training phrase 2
    Training phrase 3
    …
}
```

For example, a customer's intent to buy shirts of some color, size, and group (men, women, …) can be defined as follows:

```
order (Product product, Number numItems, Color color, Size size, Group group) {

    // Training phrases
    I want to buy two green shirts.
    I need a T-shirt for men, large, and orange.
    We need three medium hats for ladies.
        …
}
```

Significance of Intent Definitions
The most crucial task for a virtual agent in an AI dialogue system is gathering all necessary information to fulfill an intent once it's identified. For instance, when a customer wants to buy shirts, the system must collect information like the quantity, color, size, and group. If the customer says, "I want to buy two shirts for men," the system already has some information but needs to prompt for color and size.

A clearly defined intent signature guides the system's actions. For ordering shirts, the signature could be:

```
order (Product product, Number numShirts, Color color, Size size, Group group);
```

This list of parameters tells the system what information to gather from the customer. The virtual agent will continue prompting the customer until all five parameters have been collected so that the system can fulfill the intent and complete the order transaction successfully.

Data -Intent Dialogue Flows: Inheritance
Inheritance, a key principle in object-oriented (OO) methodology, allows classes to inherit attributes and methods from their superclass, depending on access modifiers like "public" or "protected."

This principle can be applied to data-intent dialogue flows. A child flow can inherit "protected" entities, entity types (data), and intents (actions) from a parent flow if a parent-child relationship is established.

For example, a super flow can be defined as follows:

```
Data-Intent Abstract DI-Super-Flow {

Protected <entity type> {… }
…

Protected <intent> (…) { … }
…

}
```

If all the entities, entity types, and intents of the super flow are defined with the modifier "**protected**," its child flows can inherit all properties it has, including entities, entity types, and intents as shown in the following example.

```
Data-Intent Products-Info: DI-Super-Flow {
        …
}
```

An **inheritance relationship** (Parent-Child) between this flow and the super flow has been established. The **colon ':'** in "**Products-Info: DI-Super-Flow"** indicates the **inheritance**, similar to what is used in OO programming languages like Java and C++.

The inheritance principle in dialogue flows reduces duplication, especially in entity and entity type definitions and declarations.

5.AI DIALOGUE SYSTEM: DESIGN

AI Dialogue System: Design: Overview
This section demonstrates designing a simple AI dialogue system for a retail store to showcase

how to design an AI dialogue system using Speech Act Theory and OO methodology in a real-world business application.

The store aims to create a system for 24/7 customer interaction across the USA. For simplicity, it is assumed that clothes have only **three properties**: **color, size, and group** (men, women, children).

The design follows the standard system development life cycle (SDLC): Planning, System Analysis, System Design, and System Implementation/Deployment. These phases may overlap, and agile methodology can be used for faster feedback and improvement.

SDLC: Phase I: Planning

In Phase I, the organization that wants to build a new AI dialogue system should identify all business values of the targeted system. For example, a retailer wants an AI dialogue system that enables it to do business 24/7, serve its customers better, improve and expand its business, and increase revenues and profits, to name a few. In addition to identifying business values, the organization should also complete other planning activities to deliver a project plan that includes the work plan, the project staffing, the technical planning, and the project management scheme.

SDLC: Phase II: System Analysis and Requirements

In Phase II, the organization should perform the system analysis to collect critical system information such as system requirements and system concepts. Generally, the system analysis aims to answer the following questions: (1) Who will use the new AI dialogue system? (2): What will the new AI dialogue system do? (3): Where will the new AI dialogue system be used? (4): When will the new AI dialogue system be used?

In addition, the organization can gather the following operational requirements for the new system:

1. System Requirement 1: Customers can use the system 24/7 to ask about products.
2. System Requirement 2: Customers can use the system 24/7 to order products online.
3. System Requirement 3: Customers can use the system 24/7 to get online customer service.

SDLC: Phase III: Design: Routine Flows

For routine dialogue flows that often include only intents, the author proposes to define them as a structure similar to the interface of the Java programming language in which an interface only contains methods and optional constants (the particular case of data) but not variables (typical case of data).

Default Greetings Flow is a routine dialogue flow with only intents (to greet) but not data, i.e., no associated entities or entity types. This routine flow can be defined as follows:

```
Routine Default-Greetings-Flow {
        // Only one intent for greetings; no data
        void sayHello () {
                // Training phrases
                Hi.
                …
        }
}
```

SDLC: Phase III: Design: Data-Intent Flows

Like OO classes that have attributes (data) and methods (actions/behaviors), a data-intent dialogue flow requires entities and entity types (data) and intents (actions). In this example of an AI dialogue system for a retailer, based on the system requirements, the designer can sketch the system concepts of four main dialogue flows, including a default routine flow and three data-intent flows, each for one system requirement: (1) Products Info, (2) Order Process, (3) Customer Services as follows:

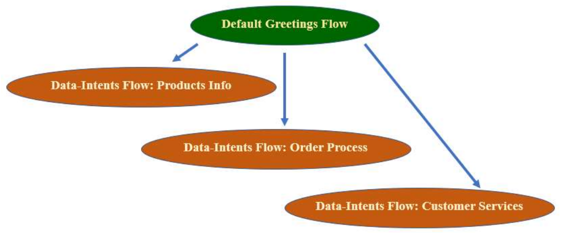

Default Greetings Flow
Data-Intents Flow: Products Info
Data-Intents Flow: Order Process
Data-Intents Flow: Customer Services

Figure 1: Four Main Dialogue Flows

All three data-intent flows have many common user-defined entities and entity types. For example, all needs product entities such as shirts, t-shirts, skirts, and others. The flows also needs entities representing sizes, and more. Instead of defining these custom entities and entity types for each data-intent flow, it is much better to use the inheritance principle to define a super flow with all common entities and entity types, from which each data-intent flow – as a child flow – can inherit. The design is much cleaner, better, easier to understand, and much more efficient without unnecessary duplication of data or intents.

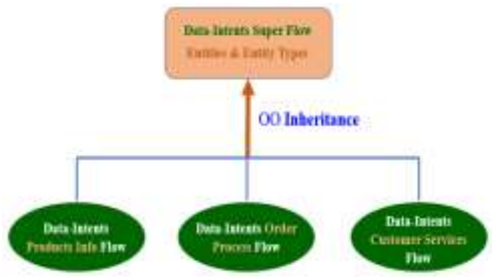

Data-Intents Super Flow
Entities & Entity Types
OO Inheritance
Data-Intents Products Info Flow
Data-Intents Order Process Flow
Data-Intents Customer Services Flow

Figure 2: Data-Intent Super and Child Flows

<u>NOTES</u>:
--) A larger image of Figure 1 is displayed in Appendix B
--) The definition details of all the data-intent super flows are presented in Appendix C.

The data-intent super flow has only data, i.e., entities and entity types, and no intents. It should be declared with the modifier "abstract" to indicate that it will not be implemented as a normal data-intent dialogue flow, similar to abstract superclasses in OO programming languages. The super flow can be declared as follows:

<u>NOTES</u>:
*--) The declaration details of all data-intent dialogue flows, including the super flow, are presented in **Appendix C**.*

```
Data-Intent Abstract DI-Super-Flow {
// Define common entities and entity types of
data-intent flows

Protected Enum-List Product { Shirt, … }
Protected Enum-List-Map Yes-No { … }
Protected Enum-List-Map Size { …}
Protected Enum-List-Map Group { …}
Protected Struct Composite Order { …}

}
```

The data-intent super flow, named "**DI-Super-Flow**," is specified with the flow type of "Data-Intent." It only has entities and entity types, without any intents. All child flows can inherit these entities and entity types.

```
Data-Intent Products-Info: DI-Super-Flow {
        // Intents
        askProduct (Product product, …){

        // Training phrases
        Do you sell T-shirts for women?
        …

        }

} // End of Data-Intent Products-Info
```

The data-intent flow, "**Products-Info**," a child of the super flow, handles the scenarios when a customer wants to ask for information about some products. This data-intent flow has

- All entities and entity types inherited from the super flow (do not need to be displayed)
- Only one intent: askProduct (…)

```
Data-Intent Order-Process: DI-Super-Flow {
        // Intent to order only one product type
        order (Product product, … {
        // Training phrases
        I want to buy two green shirts for men.
        …
        }

        // Intent to order more than one
        // product type in the same utterance
        orderComplex (…) {
        // Training phrases
        I want to buy two green shirts for men
        and one hat for me.
        …
        } // End of orderComplex

} // End of Data-Intent Order-Process
```

The data-intent flow, "**Order-Process**," also a child of the super flow, has the following properties:

- All entities and entity types inherited from the super flow
- Two intents: order (…) {…} and orderComplex (…) {…}

```
Data-Intent Customer-Services: DI-Super-
Flow {
        askOrder (Number orderNum) {

                // Training phrases
                When is my order delivered?
                …
        }

        changeOrder (Number orderNum) {

                // Training phrases
                I want to change my order.
                …
        }

        cancelOrder (Number orderNum) {

                // Training phrases
                I want to cancel my order.
                …

        }

} // End of Data-Intent Customer-Services
```

The data-intent flow, "**Customer-Services**," also a child of the super flow, has the following properties:

- *All entities and entity types inherited from the super flow (do not need to be displayed)*
- Three intents: askOrder (…) {…}, changeOrder (…) {…}, cancelOrder (…) {…}

SDLC: Phase III: Design: State Machines
Similar to the state machine of a software program, the designer should create one state machine to document all the steps or states of each data-intent dialogue flow representing a conversational flow or session. For this paper, only the state machine of the data-intent dialogue flow Order Process is created as an example. The details of the state machine are discussed in Appendix D.

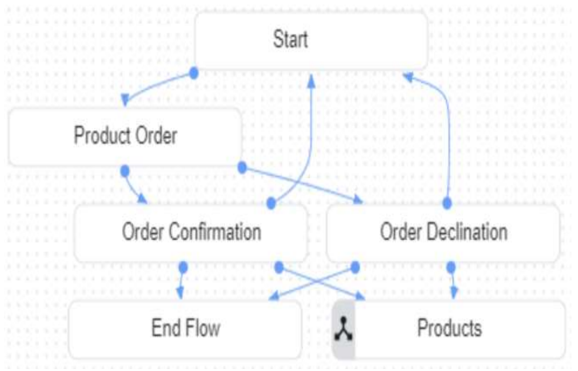


Figure 3: State Machine of Order Process

SDLC: Phase IV: Implementation
The enterprise cloud service Dialogflow CX of Google Cloud Platform (GCP) is used to implement the design of the retailer's AI dialogue system, which is used as an example to demonstrate how the system is built, tested, and deployed.

The step-by-step details of the implementation and deployment of the AI dialogue system are discussed in the next section.

6.AI DIALOGUE SYSTEM: IMPLEMENTATION

Dialogflow CX: Overview
Generative conversational AI is revolutionizing business-customer interactions. Evolved from an earlier version (Dialogflow ES), the Google Dialogflow CX employs a state machine approach for flexible conversational flows and uses modular flows and pages for scalability. Each page represents a state with its data, and every dialogue starts with the "Start Page."

Dialogflow CX provides a robust understanding engine for natural languages, accurately interpreting intents and managing context. Integrated with other Google Cloud products, the conversational AI development system enables task execution and personalized experiences. The system supports multi-channel interactions via websites, voice calls, and messaging platforms.

Dialogflow CX offers a built-in "Default Start Flow" and numerous system entities and entity types for convenient use. Designers and developers can also create user-defined ones.

NOTES:
--) All the implementation steps are discussed in detail, including visual explanation with figures presented in Appendix E.

Create Virtual Agent
To build an AI dialogue system with Dialogflow CX, we first create an agent like a human call center agent.

Dialogflow CX offers a built-in "Default Start Flow" for initiating conversations. This flow begins with a "Start Page" for a default "Default Welcome Intent." Customization of the virtual agent's greeting responses is possible.

Build Data-Intent Dialogue Flows
As discussed above, there are three data-intent dialogue flows: Products-Info (or Products), Order-Process, and Customer-Services. However, the demonstration focuses on only one data-intent dialogue flow – the Order Process flow.

Create User-Defined Entities and Entity Types
Based on the signature of the Order intent of the Order-Process flow:

```
order (Product anItem, Number numItems, Color color, Size size, Group group);
```

We must have **entities** defined for five entity types: Products, Number, Color, Size, and Group. DialogFlow CX provides two built-in system entity types, **@sys.number-integer** and **@sys.color** that can be used for Number and Color. Three others – Product, Size, and Group plus OrderNumber must be defined as custom entity types.

Build Data-Intent Order-Process Flow
Its intent "Order.Propducts" must be created with its training phrases.

Create Page "Product-Order"
For Dialogflow CX, each page represents a state or step of the state machine. It manages all the data required for the state, i.e., entities and entity types. Every dialogue flow starts with a default "Start Page." This page handles the customer's intent to order some products.

This page includes a form to manage the data, i.e., entities and entity types or **parameters** required for the state or step of the state machine. As mentioned above, the intent requires five parameters (Product, Number of Items, Color, Size, and Group). These five required parameters guide the virtual agent in determining whether it has all the information needed to fulfill the intent or whether it needs to gather more information.

The page also includes information, i.e. routes, about the next page or flow to which this page (or state) will transition when the system moves to the next state in the state machine. Based on these routes, Dialogflow CX automatically builds the state machine of the dialogue flow.

Test Retailer's AI Dialogue System
The designer and developer of the AI dialogue system can test their newly built system either using Test Agent or Dialogflow Messenger.

6. EVALUATION

To quickly evaluate the new design approach of entities – entity types and intents of a dialogue system, the author asked 25 graduate students to employ the new method in designing entities / entity types and intents after they had done the same thing without using the new approach while developing a conversational AI. Then, the author performed a brief oral survey on the students focusing on the following aspects:

1. Does the new design method help the user to understand the role of entities / entity types w.r.t. intents more easily and quickly?
2. Does the new design method help the user more accurately capture and then formally represent the relationships of entities / entity types with intents?
3. Does the new design method help the user to design each intent and its training phrases faster and better in a dialogue system?

The results of the survey showed 18 students (72%) say "Yes" with Question 1, 20 students (78%) say "Yes" with Question 2, and 17 students (68%) say "Yes" with Question 3.

In summary, the results of the survey were positive, suggesting the effectiveness of the new design method in aiding their understanding of entities and entity types, agreeing that it facilitated the accurate capture and formal representation of relationships, and acknowledging its role in expediting the design of intents and their training phrases.

7. CONCLUSION

Speech Act Theory (SAT) and Object-Oriented Methodology (OOM) are powerful synergies for designing AI dialogue systems. The theory understands language as actions, focusing on intentions and social dynamics. The methodology provides the structure for organizing these complex conversational components in a modular, reusable, and scalable way.

Integrating SAT and OOM enhances naturalness and adaptability in designing and developing conversational AI systems. By understanding the illocutionary force of user input via the theory, the virtual AI agent can respond more appropriately and empathetically, tailoring responses based on context for more coherent conversations. The methodology translates this into a modular, maintainable system, with speech acts encapsulated as objects for flexible development. Formulating dialogue flows and designing intents using OOM structures enables deeper insight into the system architecture. Additionally, SAT opens up new metrics for evaluating AI dialogue systems, focusing on achieving the intended effect of user communication for greater satisfaction.

Challenges include the complexity of Speech Act Theory, which requires designers and developers to understand it for practical application. Further studies need to perform a more detailed comparison of this approach with traditional design approaches, which would be beneficial.

In conclusion, the new design approach, which combines Speech Act Theory and Object-Oriented methodology, has the potential to significantly enhance the user's understanding of entities and entity types, as well as their relationships with intents. As conversational AI continues to evolve, this blend of linguistic theory and software engineering has the potential to inspire a new wave of innovation in the field of conversational AI, leading to more natural and effective interactions.

8. REFERENCES

Alphatbet Inc. (2024). Google Cloud Platform (GCP): Dialogflow CX Documentation. Retrieved August 20, 2024 from https://cloud.google.com/dialogflow/cx/docs.

Alphatbet Inc. (2024). Google Cloud Platform (GCP): Dialogflow CX Documentation – System Entities Reference. Retrieved August 20, 2024 from https://cloud.google.com/dialogflow/cx/docs/reference/system-entities.

Austin, J. L. (1962). How to do things with words. Oxford University Press, New York, NY.

Booch, G (1994). Object-Oriented Analysis and Design with Applications (2nd Ed.). The Benjamin/Cummings Publishing Company, Inc. Redwood City, CA.

Henderson, G. E. & Brown, C. (1997). Glossary of Literary Theory. Retrieved August 20, 2024 from https://resources.saylor.org/wwwresources/archived/site/wp-content/uploads/2011/04/Speech-Act-Theory.pdf.

Lucy, l. & Bamman, D. (2021). Gender and Representation Bias in GPT-3 Generated Stories. In Proceedings of the Third Workshop on Narrative Understanding, pages 48–55, Virtual. Association for Computational Linguistics.

Mambron, N. (2020). Speech Act Theory. Retrieved August 20, 2024 from https://literariness.org/2020/10/11/speech-act-theory/.

Sbisà, M. (2014). Austin on Language and Action. In: Garvey, B. (eds) J.L. Austin on Language. Philosophers in Depth. Palgrave Macmillan, London. https://doi.org/10.1057/9781137329998_2.

Searle, J. R. (1969). Speech Acts: An essay in the philosophy of language. Cambridge University Press, Cambridge.

Searle, J. R. (1983). Intentionality: An essay in the philosophy of mind. Cambridge University Press, Cambridge.

Searle, J. R., Kiefer, F., & Bierwisch, M. (1980). Speech act theory and pragmatics. Springer Netherlands, Heidelberg.

Shah, D., Schwartz, H. A., and Hovy, D. (2019). Predictive biases in natural language processing models: A conceptual framework and overview. Retrieved August 20, 2024 from https://doi.org/10.48550/arXiv.1912.11078.

Stanford Encyclopedia of Philosophy (2020). Speech Act. Retrieved August 20, 2024 from https://plato.stanford.edu/entries/speech-acts/.

Stroustrup, B. (1991). The C++ Programming Language. Addison-Wesley Publishing Company (2nd Ed.). Reading, Massachusetts, USA.

Wang, Y. & Singh, L. (2023). Adding Guard Rails to Advanced Chatbots. Retrieved August 20, 2024 from https://www.researchgate.net/publication/371537176_Adding_guardrails_to_advanced_chatbots.

Appendices and Annexures

APPENDIX A
System Analysis – System Requirements

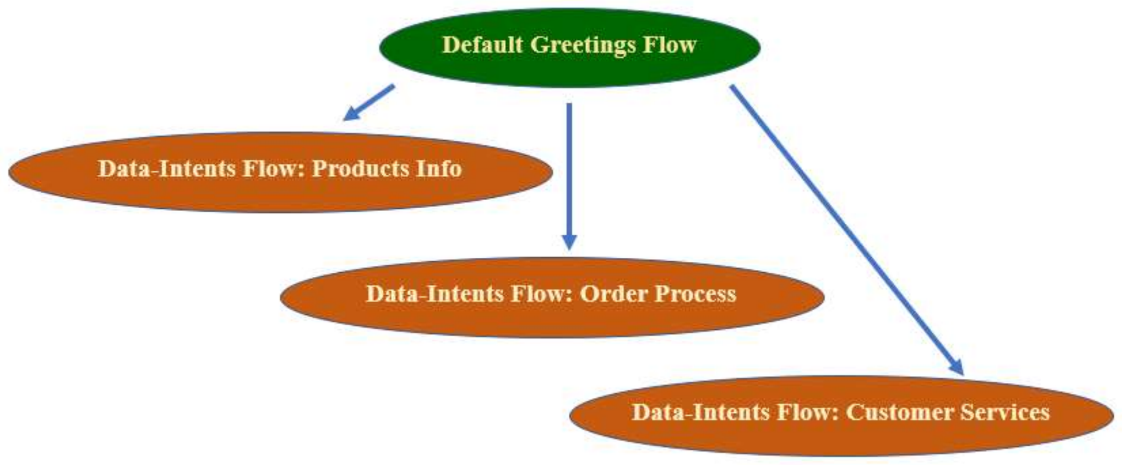

Default Greetings Flow
Data-Intents Flow: Products Info
Data-Intents Flow: Order Process
Data-Intents Flow: Customer Services

Figure 1: AI Dialogue System: Retailer Example: Three System Requirements

ISSN: 2473-4901
v10 n6126

APPENDIX B
Data-Intent Super Flow and Its Child Flows: Inheritance Relationship

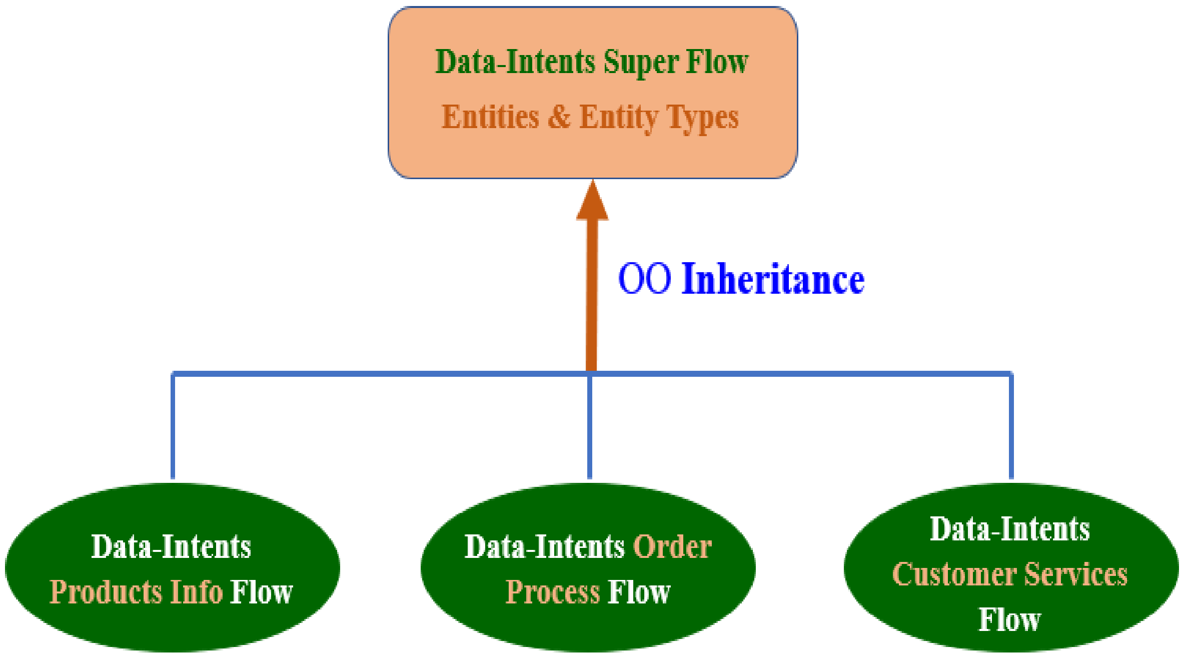

Data-Intents Super Flow
Entities & Entity Types
OO Inheritance
Data-Intents Products Info Flow
Data-Intents Order Process Flow
Data-Intents Customer Services Flow

Figure 2: Data-intent Super Flow and Its Child Flows

APPENDIX C
Data-Intent Super Flow and Its Child Flows: Detailed Definitions

DATA-INTENT DIALOGUE SUPER FLOW: DI-Super-Flow

NOTES:
*--) Name of the flow: **DI-Super-Flow.***
*--) The keyword "**Data-intent**" indicates the flow type.*
--) The super flow only has entities and entity types. It has no intents.

```
Data-intent Abstract DI-Super-Flow {

        Protected Enum-List Product {
                Shirt, T-shirt, Pant, Skirt, Hat, Shoes, Jacket, Sweater, …
        }

        Protected Enum-List-Map Yes-No {
                YES --> Yes, True, 1, OK, Correct, Fine, Good, Agree, Confirm, Do it;
                NO --> No, False, 0, Not OK, Incorrect, Bad, Disagree, Deny, Don't do it;
        }

        Protected Enum-List-Map Size {
                S → Small, Little;
                M → Medium, Average, Middle;
                L → Large, Big, Grand;
        }

        Protected Enum-List-Map Group {
                Any → Any;
                Men → Men, Man, Male, Guy;
                Women → Women, Woman, Lady, Female, Gal;
                Adult → Adult;
                Children → Children, Teens;
        }

        Protected Struct Composite Order {
                Number orderNumber// order number of the order
                Product item;           // a product item like shirt, or pants, or …
                Number numItems;        // number of items in this order
                Color color;            // Color of the items
                Size size;              // Size of the items
                Group group;            // Men or women or children, …
        }


} // End of Data-intent Abstract DI-Super-Flow
```

DATA-INTENT DIALOGUE FLOW: Products-Info

```
Data-Intent Products-Info: DI-Super-Flow {

        // Intents
        askProduct (Product product, Color color, Size size, Group group){
                // Training phrases
                Do you sell T-shirts for women?
                I want to get some pants for boys. Do you have them?
                We need some medium skirts for ladies?
                …

        }

} // End of Data-intent Products-Info: DI-Super-Flow
```

The data-intent flow, "**Products-Info**," is a child of the super flow. The **colon ':'** in "**Products-Info: DI-Super-Flow"** indicates **the inheritance relationship** between this flow and the super flow, similar to what is used in the OO programming language C++. This data-intent flow handles the scenarios when a customer wants to ask for information about some products. This data-intent flow has the following properties:

- *All entities and entity types inherited from the super flow (do not need to be displayed).*
- Only one intent: askProduct (…) {…}

DATA-INTENT DIALOGUE FLOW: Order-Process

```
Data-intent Order-Process: DI-Super-Flow {

        // Intent to order only one product type, color, size, and group.
        order (Product product, Number numItems, Color color, Size size, Group group) {
                // Training phrases
                I want to buy two green shirts for men.
                        Can I get one skirt for ladies, size small, and blue?
                May I purchase three large T-shirts, any color, for teens?
                …
        }

        // Intent to order more than one product type in the same utterance
        orderComplex (Product product_1, Product product_2) {

                // Training phrases
                I want to buy two green shirts for men and one hat for me.
                Can I get one skirt, two shirts, and three T-shirts?
        May I purchase hats, shoes, ties, and suits here?
        …
    } // End of orderComplex (Product product_1, Product product_2)

} // End of Data-intent Order-Process
```

The data-intent flow, "**Order-Process**," is also a child of the super flow. This data-intent flow has the following properties:

- *All entities and entity types inherited from the super flow (do not need to be displayed).*
- Two intents: order (…) {…} and orderComplex (…) {…}

NOTES:
*--) The virtual agent **needs to recognize only two types of product** in the same utterance to **match the intent orderComplex**, no matter how many types of product in the same utterance.*
*--) In this case, the virtual agent should generate the necessary prompts for the customer to guide him/her in completing the order of each product type at a time. In other words, the virtual agent should help the customer **split a complex intent into multiple simple intents**, each of only one product type.*

DATA-INTENT DIALOGUE FLOW: Customer-Services

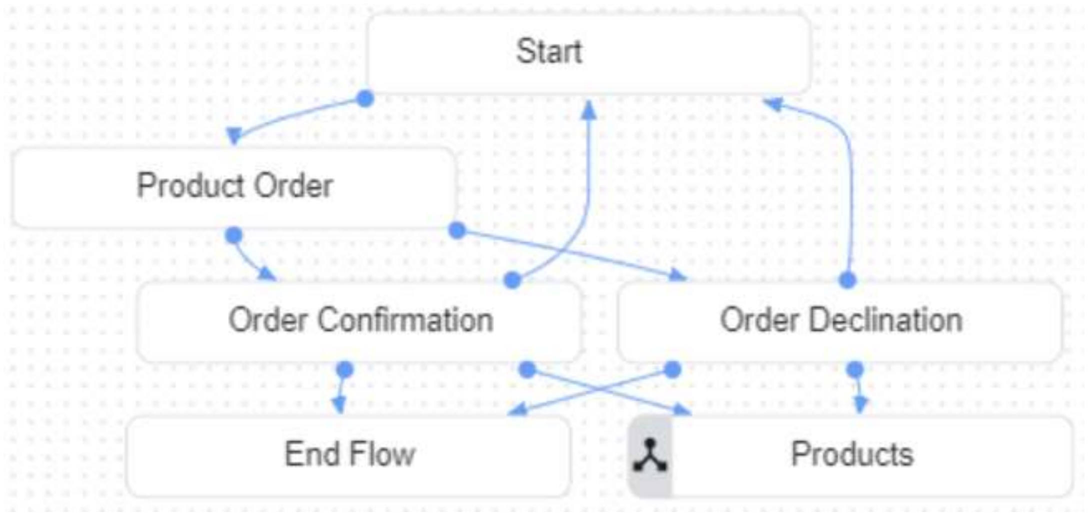
```
Data-Intent Customer-Services: DI-Super-Flow {

    askOrder (Number orderNum) {

        // Training phrases
        When is my order delivered?
        What did I order?
        When can I receive my items?
        …
    }

    changeOrder (Number orderNum) {

        // Training phrases
        I want to change my order.
        Can I change what I bought two days ago?
        Can I change the shirt I ordered and get a larger size?
        …
    }
    cancelOrder (Number orderNum) {

        // Training phrases
        I want to cancel my order.
        Can I cancel the order I made yesterday?
        Would you cancel the order of two shirts?
        …
    }

} // End of Data-intent Customer-Services
```

The data-intent flow, "**Customer-Services**," is also a child of the super flow. This data-intent flow has the following properties:

- *All entities and entity types inherited from the super flow (do not need to be displayed)*
- Three intents: askOrder (…) {…}, changeOrder (…) {…}, cancelOrder (…) {…}

APPENDIX D
Data-Intent Super Flow and Its Child Flows: Detailed Definitions

For each data-intent dialogue flow, the designer should create one state machine to cover all its steps or states of the flow. For this paper, only the state machine of the data-intent dialogue flow Order Process is created as an example.



1. Start
    a. The data-intent dialogue flow Order-Process starts with the Start step or state

2. Product Order
    a. At this step or state, first, the customer orders some products.
    b. The system tries to collect all required parameters to fulfill the customer's intent – to order some products.
    c. After the order has been taken, the system asks the customer to confirm what he/she has ordered to ensure the order has been taken correctly to fulfill his/her intent. Two scenarios can occur:
        i. The customer confirms the order.
            1. → Transition to the step of Order Confirmation
        ii. The customer declines the order.
            1. → Transition to the step of Order Declination

3. Order Confirmation
    a. At this step or state, the customer confirms his/her order
    b. After this step, the customer has completed his/her order and successfully his/her intent, and the dialogue flow ends.

ISSN: 2473-4901
v10 n6126

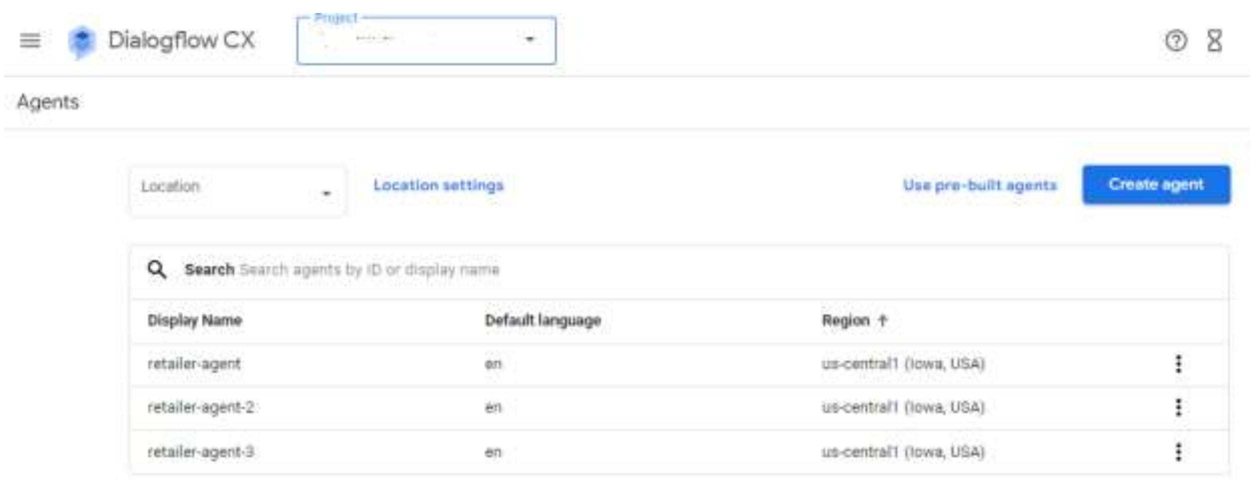
4. Order Declination
    a. At this step or state, the customer declines the order.
    b. After this step, two scenarios can occur:
        i. The customer completely stop the order. → The dialogue flow ends.
        ii. The customer may want to make another order.
            1. → Transition back to the start of the order flow.
        iii. The customer way want to ask for more information of some products.
            1. → Transition to another dialogue flow – the Products flow.

ISSN: 2473-4901
v10 n6126

APPENDIX E
Data-Intent Super Flow and Its Child Flows: Detailed Definitions

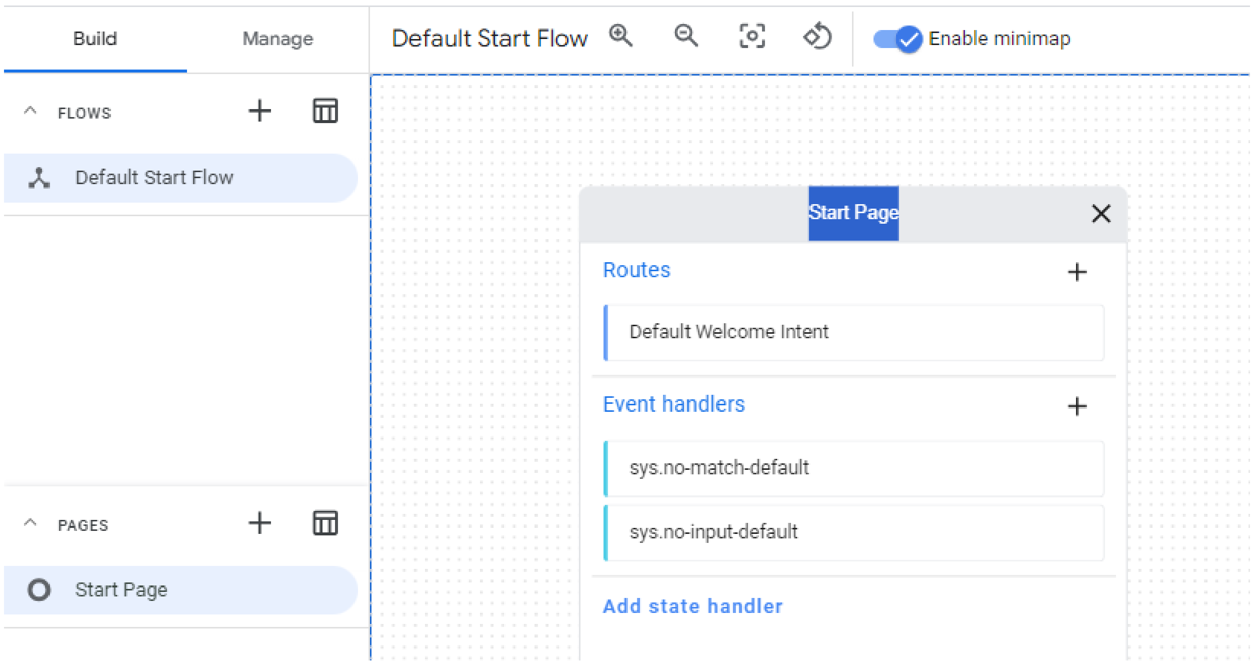
Create Virtual Agent

To build an AI dialogue system with Dialogflow CX, we first create an agent like a human call center agent.



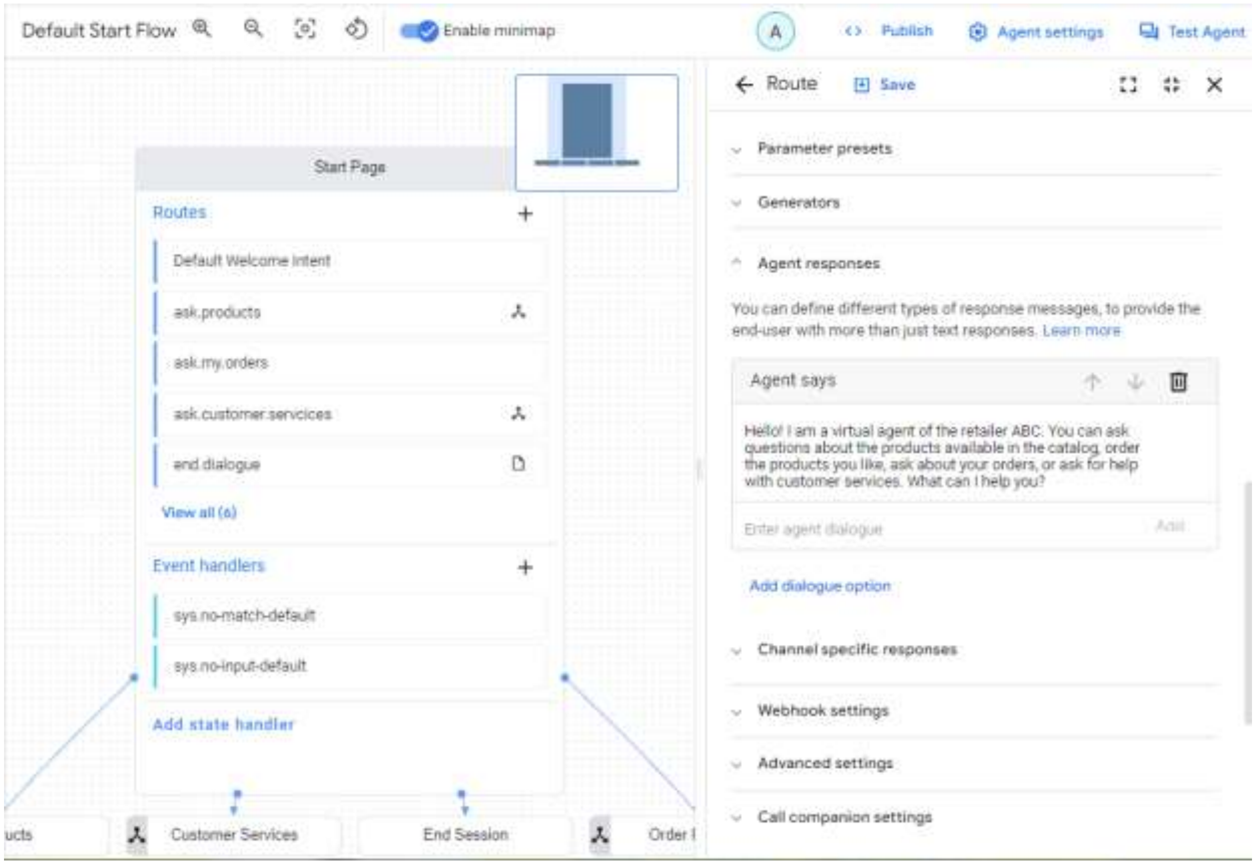
Display Name	Default language	Region ↑
retailer-agent	en	us-central1 (Iowa, USA)
retailer-agent-2	en	us-central1 (Iowa, USA)
retailer-agent-3	en	us-central1 (Iowa, USA)

Dialogflow CX offers a built-in "Default Start Flow" for initiating conversations. This flow begins with a "Start Page" for a default "Default Welcome Intent." Customization of the virtual agent's greeting responses is possible.



ISSN: 2473-4901
v10 n6126

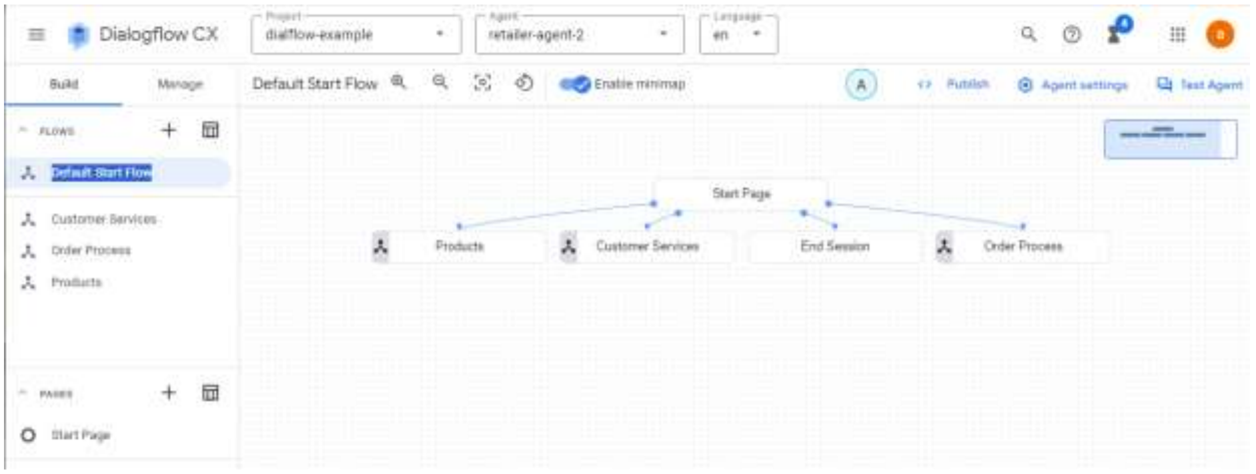
It is possible to customize the virtual agent's responses to the customer's greetings.



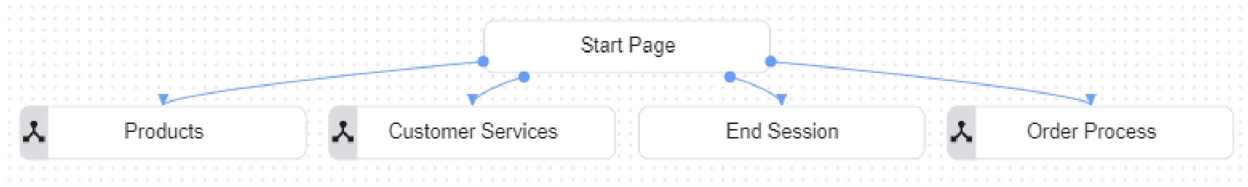
You can define different types of response messages, to provide the end-user with more than just text responses. Learn more

Hello! I am a virtual agent of the retailer ABC. You can ask questions about the products available in the catalog, order the products you like, ask about your orders, or ask for help with customer services. What can I help you?

Build Data-Intent Dialogue Flows
As discussed above, there are three data-intent dialogue flows: Products-Info (or Products), Order-Process, and Customer-Services. However, the demonstration focuses on only one data-intent dialogue flow – the Order Process flow.



ISSN: 2473-4901
v10 n6126

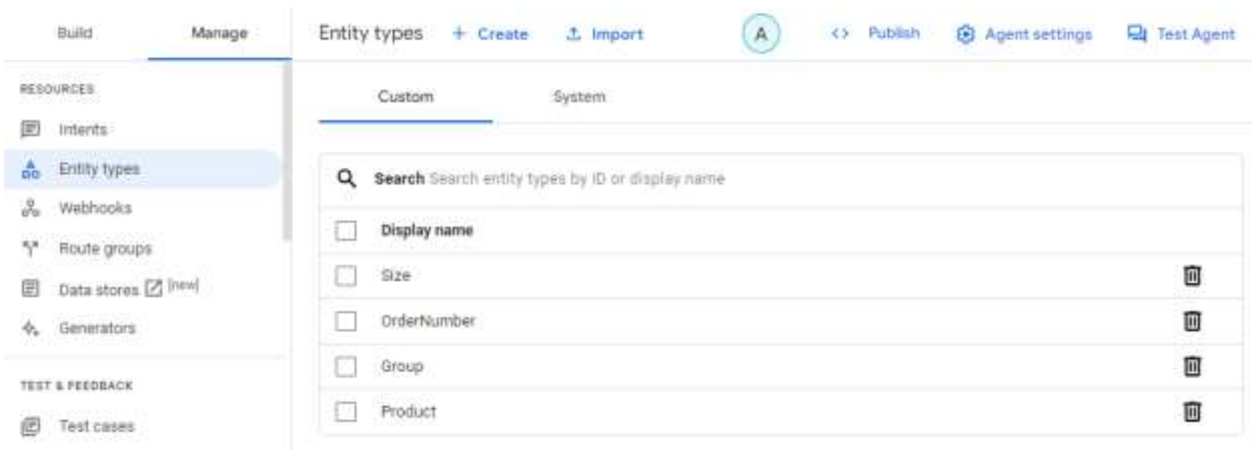


Create User-Defined Entities and Entity Types
Based on the signature of the Order intent of the Order-Process flow:

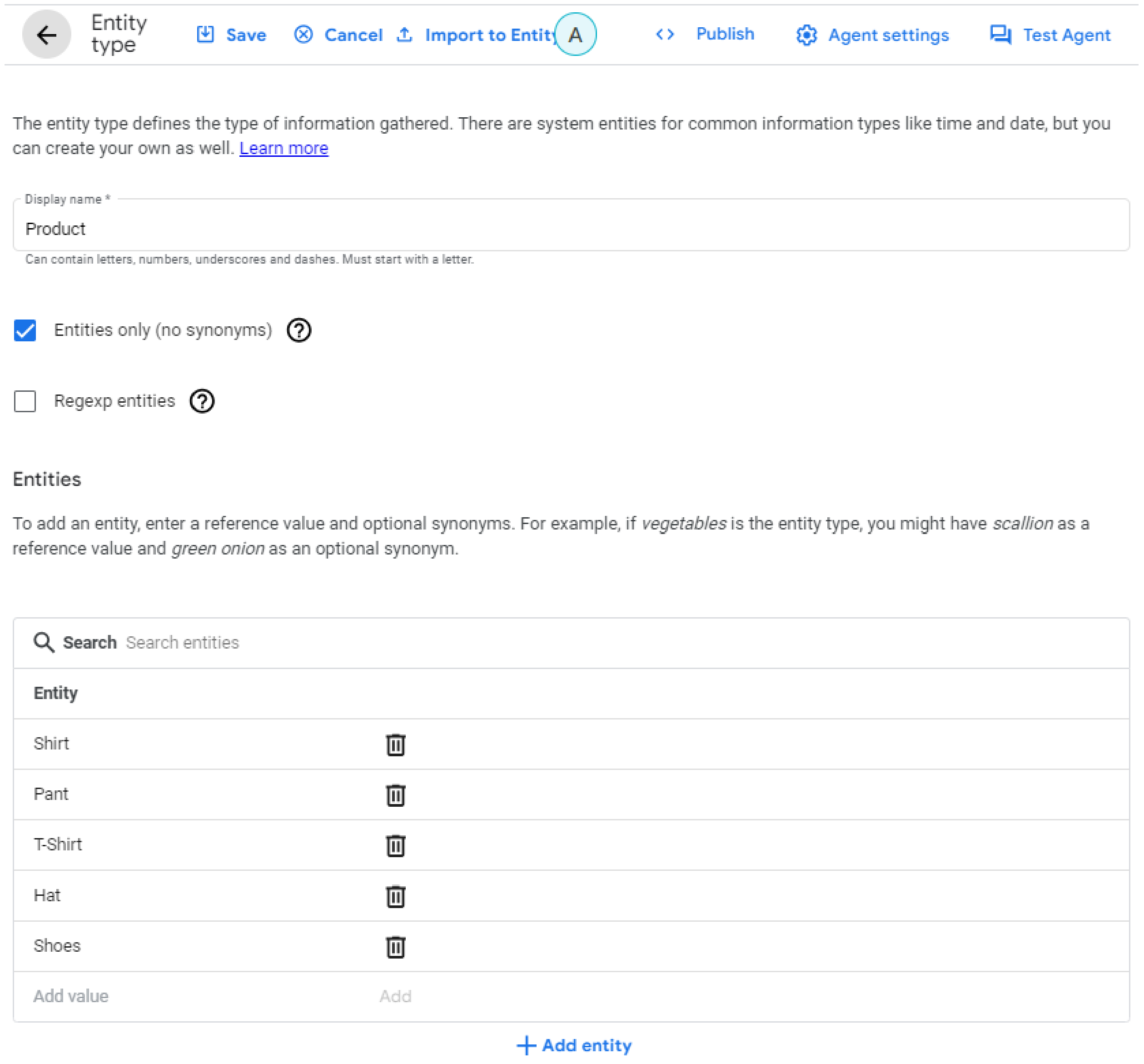
order (Product anItem, Number numItems, Color color, Size size, Group group);

There must be entities defined for five entity types: Products, Number, Color, Size, and Group. DialogFlow CX provides two built-in system entity types, **@sys.number-integer** and **@sys.color** that can be used for Number and Color. Three others – Product, Size, and Group plus OrderNumber must be defined as custom entity types.

Also, OrderNumber is another entity type that must be defined to represent the values of order numbers. The custom entities, i.e., values of order numbers, should be alphanumeric strings that can be defined as Regexp entities



Display name
Size
OrderNumber
Group
Product



The entity type defines the type of information gathered. There are system entities for common information types like time and date, but you can create your own as well. Learn more

Display name *
Product
Can contain letters, numbers, underscores and dashes. Must start with a letter.

[✓] Entities only (no synonyms)

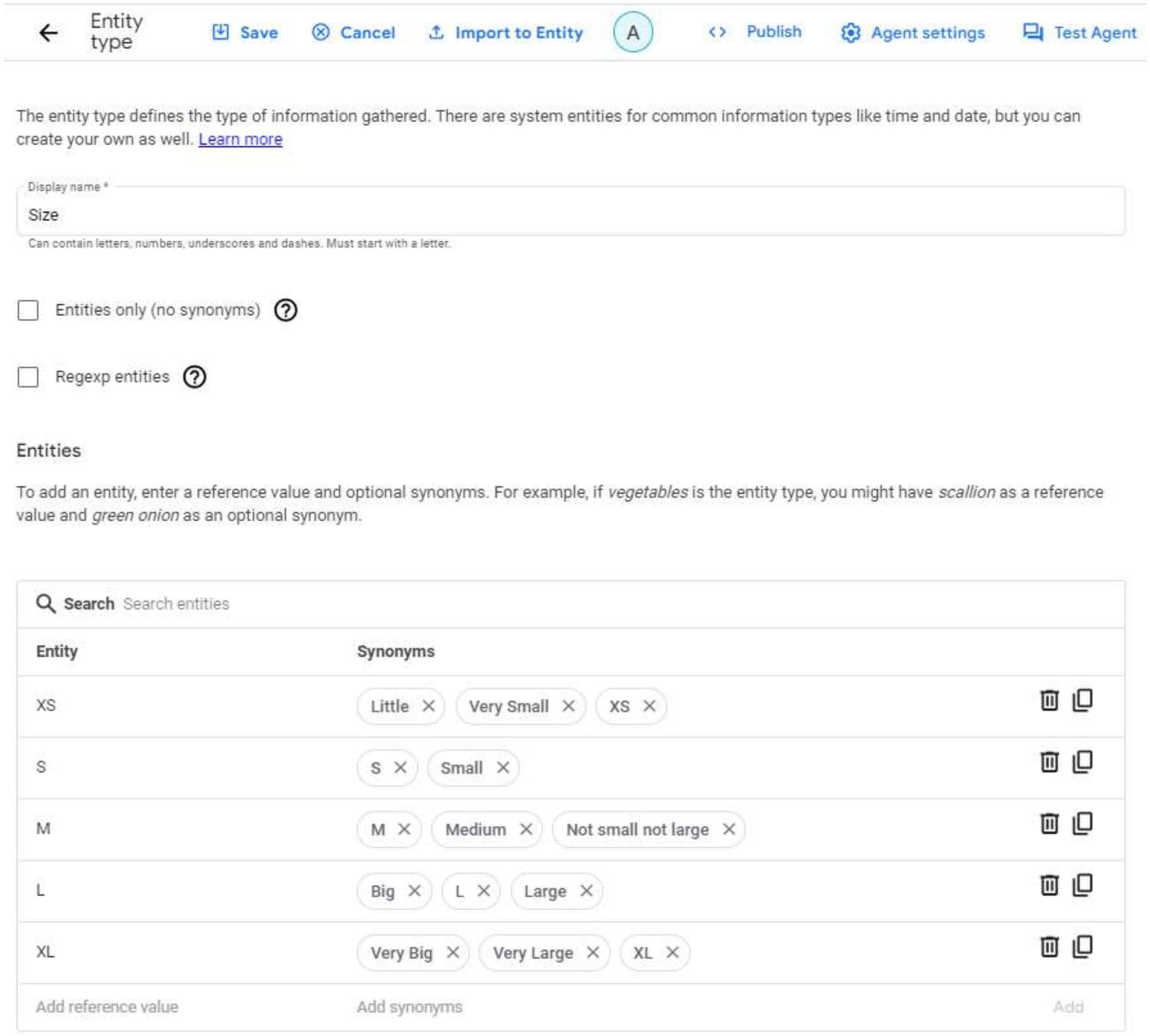
[ ] Regexp entities

Entities

To add an entity, enter a reference value and optional synonyms. For example, if *vegetables* is the entity type, you might have *scallion* as a reference value and *green onion* as an optional synonym.

Search Search entities

Entity
Shirt
Pant
T-Shirt
Hat
Shoes
Add value



The entity type defines the type of information gathered. There are system entities for common information types like time and date, but you can create your own as well. Learn more

Display name *
Size
Can contain letters, numbers, underscores and dashes. Must start with a letter.

☐ Entities only (no synonyms)

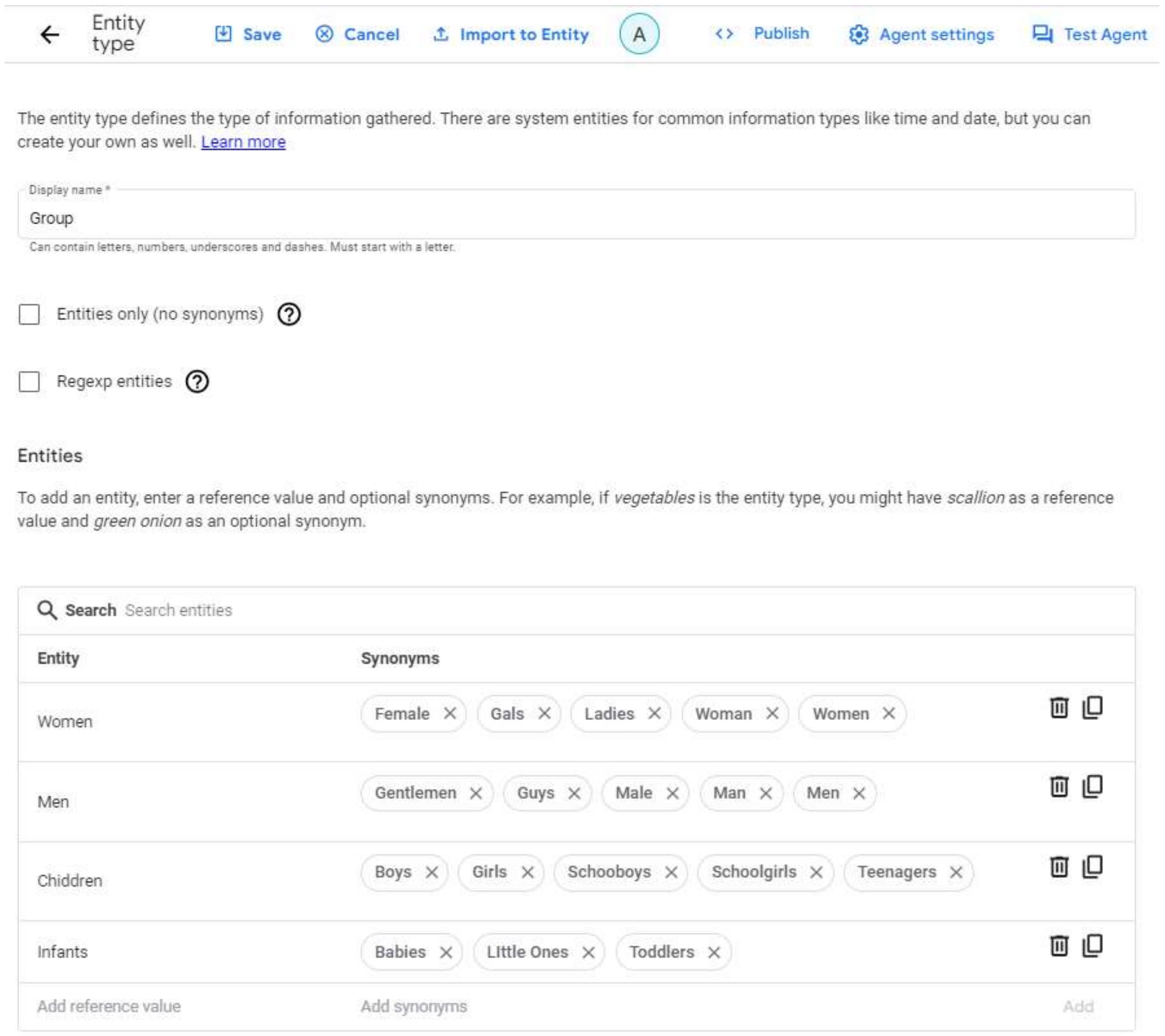
☐ Regexp entities

Entities

To add an entity, enter a reference value and optional synonyms. For example, if *vegetables* is the entity type, you might have *scallion* as a reference value and *green onion* as an optional synonym.

Search Search entities

Entity	Synonyms
XS	Little, Very Small, XS
S	S, Small
M	M, Medium, Not small not large
L	Big, L, Large
XL	Very Big, Very Large, XL
Add reference value	Add synonyms



The entity type defines the type of information gathered. There are system entities for common information types like time and date, but you can create your own as well. Learn more

Display name *
Group
Can contain letters, numbers, underscores and dashes. Must start with a letter.

☐ Entities only (no synonyms)

☐ Regexp entities

Entities

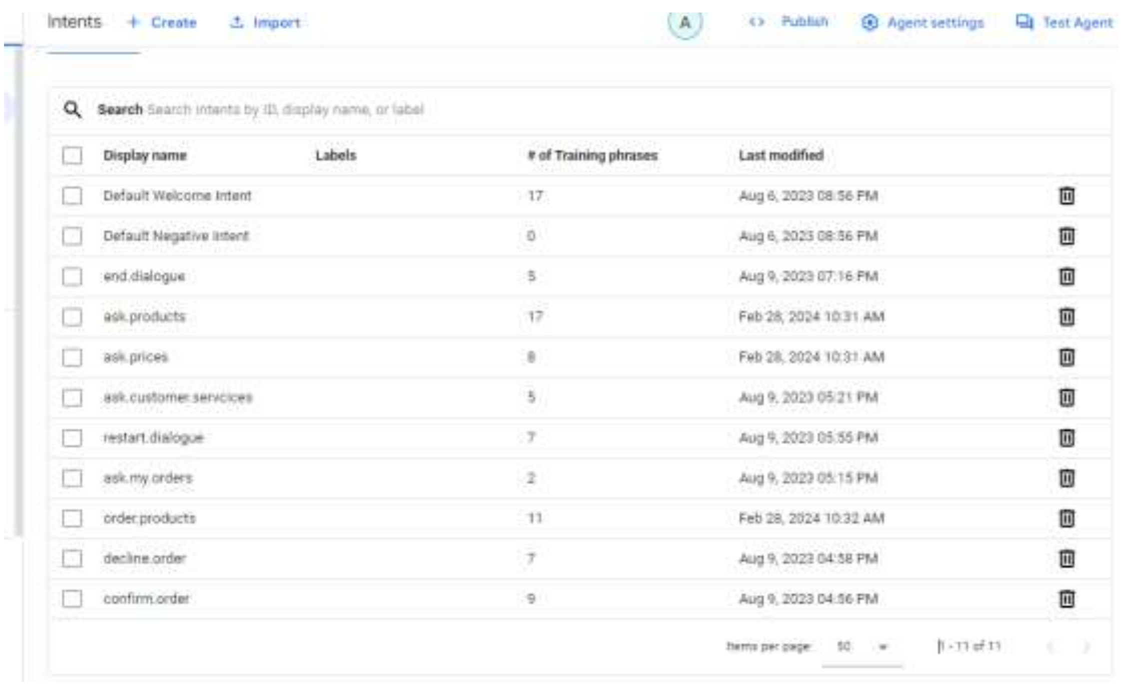
To add an entity, enter a reference value and optional synonyms. For example, if *vegetables* is the entity type, you might have *scallion* as a reference value and *green onion* as an optional synonym.

Search Search entities

Entity	Synonyms
Women	Female, Gals, Ladies, Woman, Women
Men	Gentlemen, Guys, Male, Man, Men
Chiddren	Boys, Girls, Schooboys, Schoolgirls, Teenagers
Infants	Babies, LIttle Ones, Toddlers
Add reference value	Add synonyms

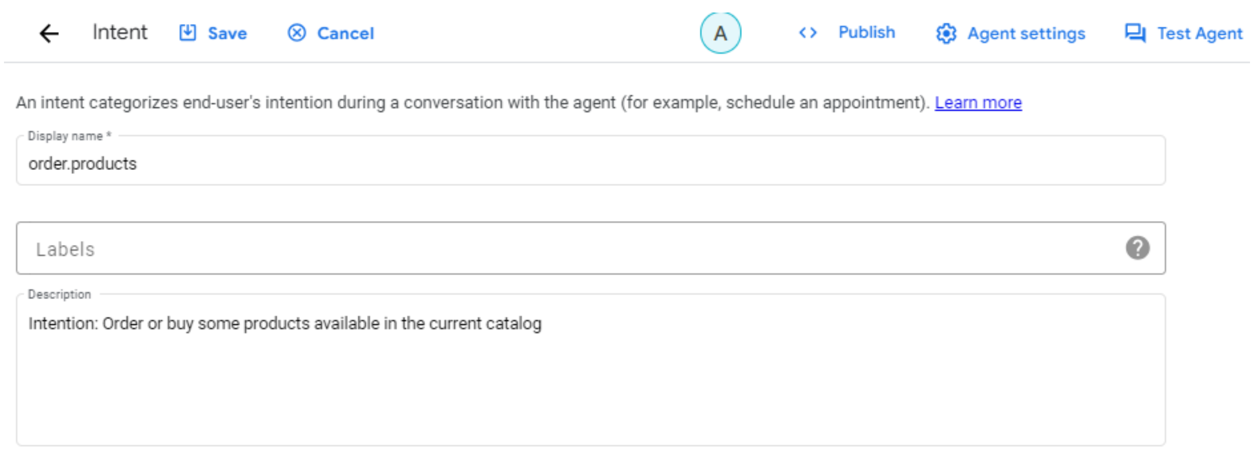
Build Data-Intent Order-Process Flow

The intent “Order.Propducts” must be created with its training phrases.



Display name	Labels	# of Training phrases	Last modified
Default Welcome Intent		17	Aug 6, 2023 08:56 PM
Default Negative Intent		0	Aug 6, 2023 08:56 PM
end.dialogue		5	Aug 9, 2023 07:16 PM
ask.products		17	Feb 28, 2024 10:31 AM
ask.prices		8	Feb 28, 2024 10:31 AM
ask.customer.services		5	Aug 9, 2023 05:21 PM
restart.dialogue		7	Aug 9, 2023 05:55 PM
ask.my.orders		2	Aug 9, 2023 05:15 PM
order.products		11	Feb 28, 2024 10:32 AM
decline.order		7	Aug 9, 2023 04:38 PM
confirm.order		9	Aug 9, 2023 04:56 PM

Items per page: 50
1 - 11 of 11



An intent categorizes end-user's intention during a conversation with the agent (for example, schedule an appointment). Learn more

Display name *
order.products

Labels

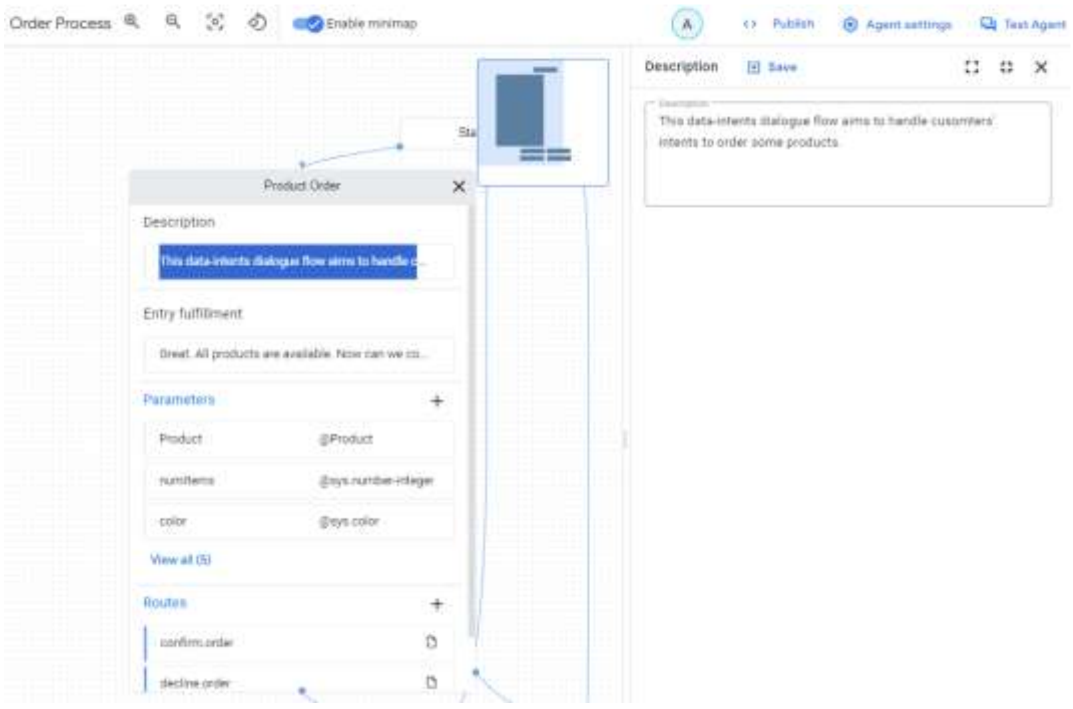
Description
Intention: Order or buy some products available in the current catalog

Create Page "Product-Order"
For Dialogflow CX, each page represents a state or step of the state machine. It manages all the data required for the state, i.e., entities and entity types. Every dialogue flow starts with a default "Start Page." This page handles the customer's intent to order some products.

This page includes a form to manage the data, i.e., entities and entity types or **parameters** required for the state or step of the state machine. As mentioned above, it requires five parameters (Product, Number of Items, Color, Size, and Group). These five required parameters guide the virtual agent in determining whether it has all the information needed to fulfill the intent or whether it needs to continue gathering more information.

ISSN: 2473-4901
v10 n6126

The page also includes information, i.e. routes, about the next page or flow to which this page (or state) will transition when the system moves to the next state in the state machine. Based on these routes, Dialogflow CX automatically builds the state machine of the dialogue flow.



Description

This data-intents dialogue flow aims to handle c...

Entry fulfillment

Great. All products are available. Now can we co...

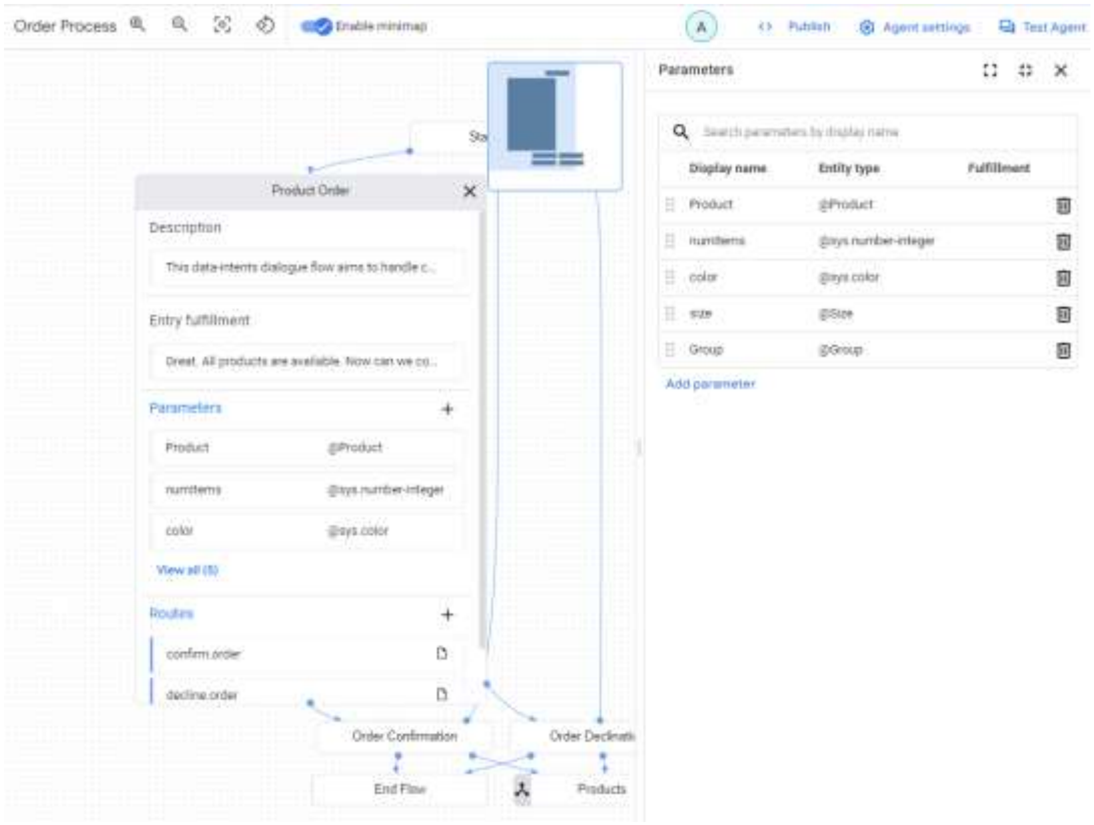
Product	@Product
numItems	@sys.number-integer
color	@sys.color

confirm.order

decline.order

Description

This data-intents dialogue flow aims to handle cusomers' intents to order some products.



Description

This data-intents dialogue flow aims to handle c...

Entry fulfillment

Great. All products are available. Now can we co...

Product	@Product
numItems	@sys.number-integer
color	@sys.color

confirm.order

decline.order

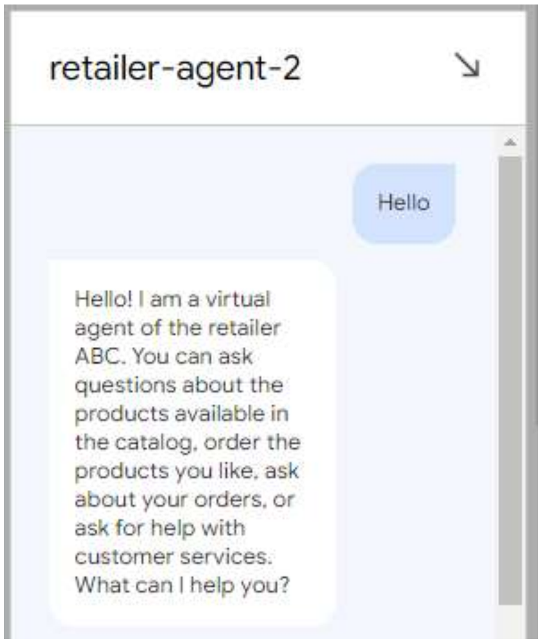
Parameters

Search parameters by display name

Display name	Entity type	Fulfillment
Product	@Product	
numItems	@sys.number-integer	
color	@sys.color	
size	@Size	
Group	@Group	

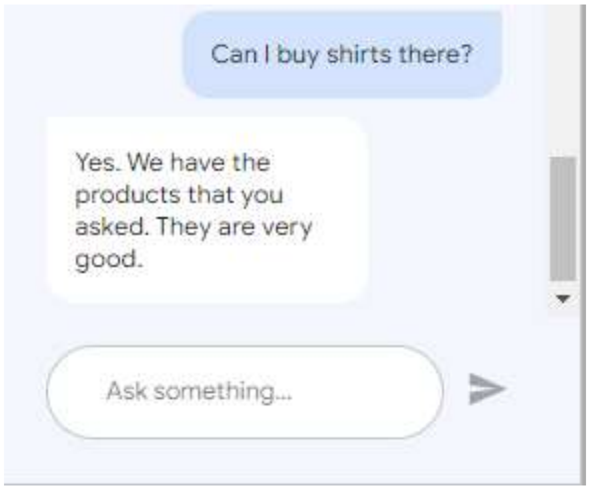
Test Retailer's AI Dialogue System

The designer and developer of the AI dialogue system can test their newly built system either using Test Agent or Dialogflow Messenger.



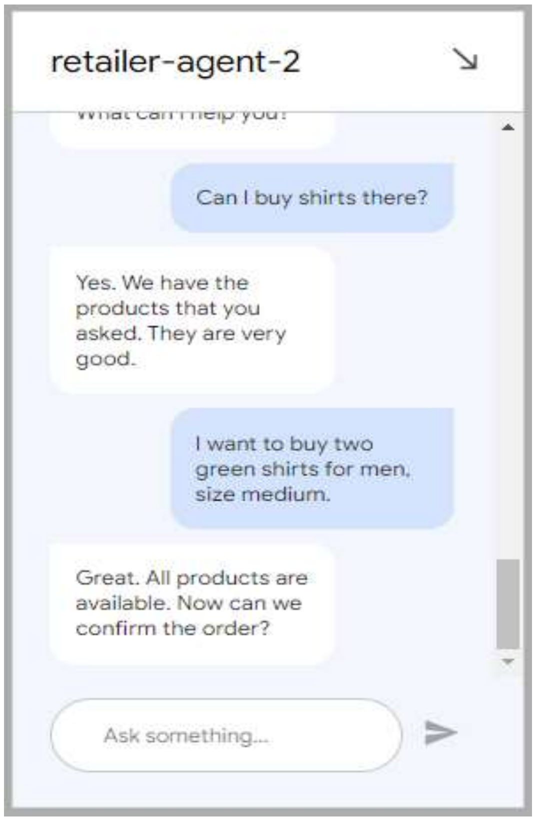
Hello

Hello! I am a virtual agent of the retailer ABC. You can ask questions about the products available in the catalog, order the products you like, ask about your orders, or ask for help with customer services. What can I help you?



Can I buy shirts there?

Yes. We have the products that you asked. They are very good.



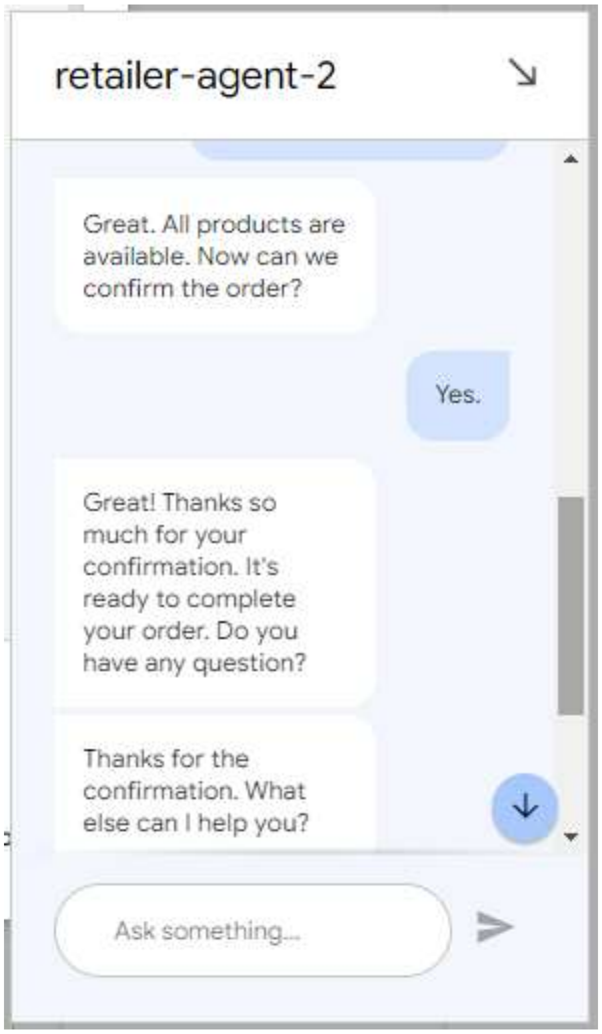
What can I help you?

Can I buy shirts there?

Yes. We have the products that you asked. They are very good.

I want to buy two green shirts for men, size medium.

Great. All products are available. Now can we confirm the order?



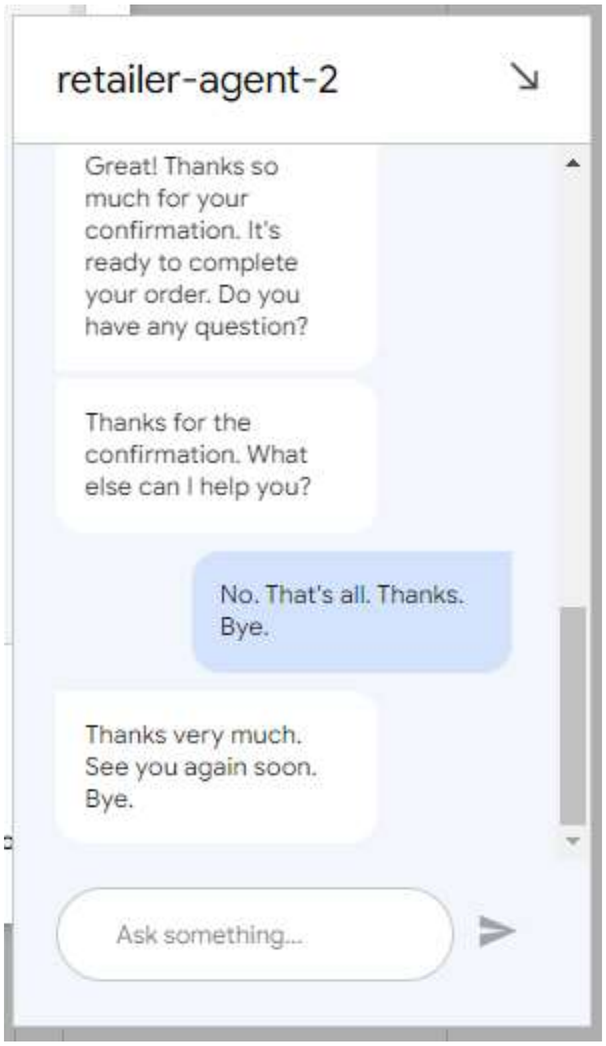
retailer-agent-2

Great. All products are available. Now can we confirm the order?

Yes.

Great! Thanks so much for your confirmation. It's ready to complete your order. Do you have any question?

Thanks for the confirmation. What else can I help you?



retailer-agent-2

Great! Thanks so much for your confirmation. It's ready to complete your order. Do you have any question?

Thanks for the confirmation. What else can I help you?

No. That's all. Thanks. Bye.

Thanks very much. See you again soon. Bye.