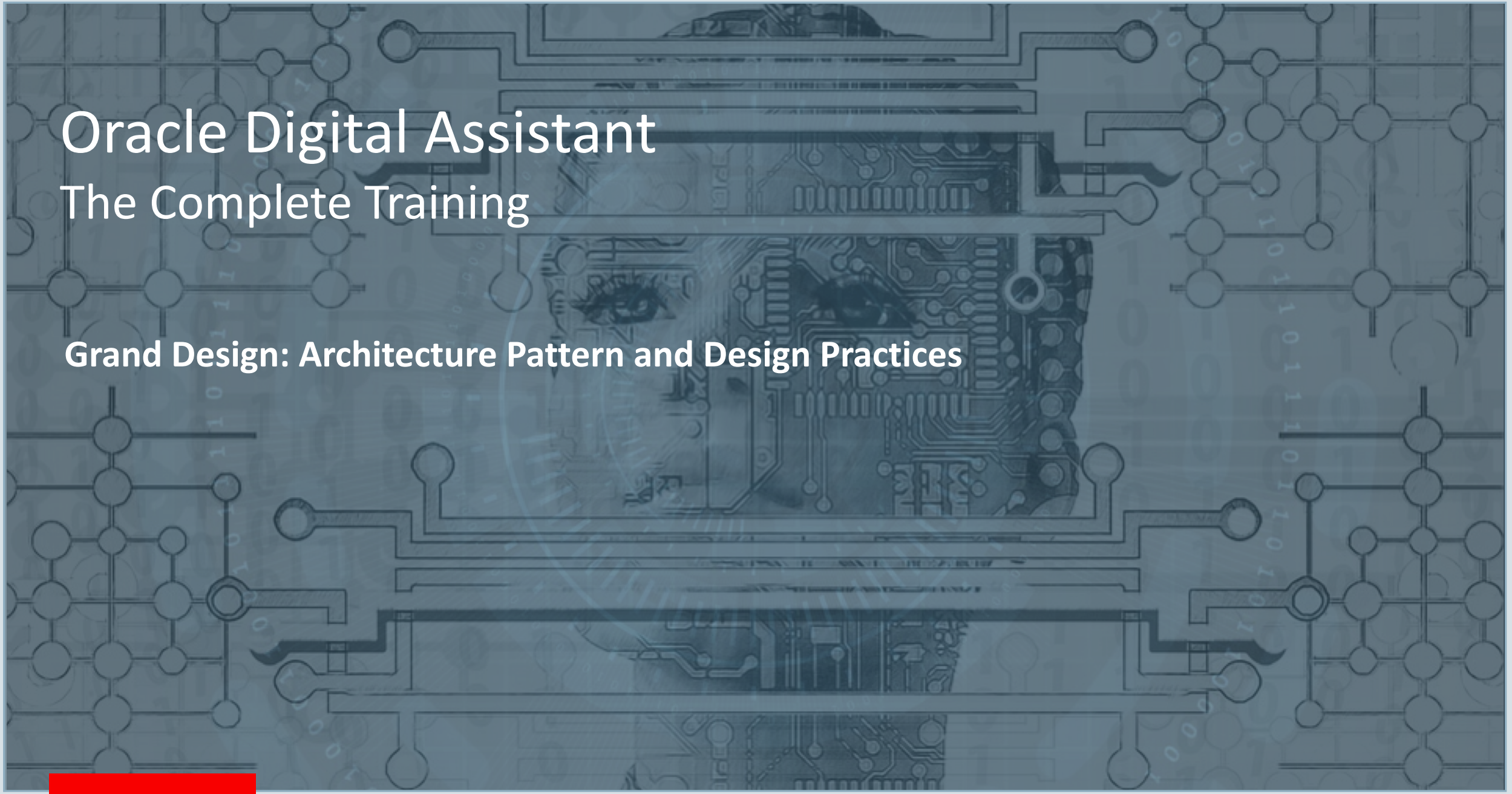

ORACLE®

# Oracle Digital Assistant

## The Complete Training

**Grand Design: Architecture Pattern and Design Practices**



ORACLE

Image courtesy of pixabay.com

# Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

# Topic agenda

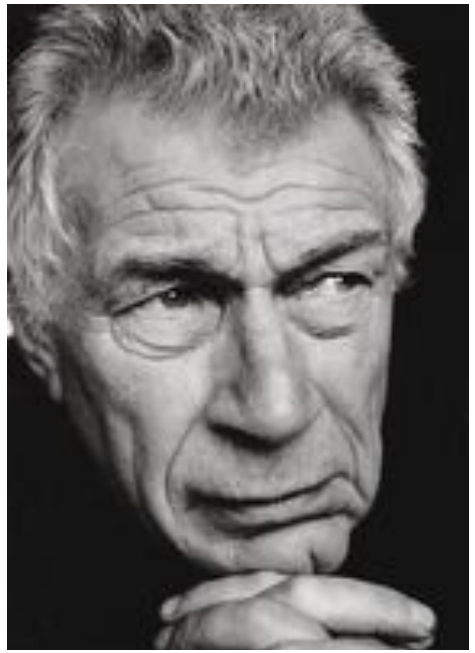
1. Architecture matters
2. Skill patterns
3. Digital assistant pattern
4. Skill Parameters
5. QnA

# Topic agenda

1. Architecture matters
2. Skill patterns
3. Digital assistant pattern
4. Skill Parameters
5. QnA

# Artificial intelligence alone doesn't build the bot

- Chatbots automate user interaction on the conversational channel
  - Requires good conversation design skills
  - Uses a sound mix of conversational AI and dialog flow to assist and guide users
- Building chatbots is a software development project
  - Full development lifecycle
- Artificial intelligence does not replace good design practices
  - It is just a tool for you to use



"The way we see things is affected by what we know or what we believe."

– John Peter Berger, English art critic, novelist and author

The **more** you know, the better you build

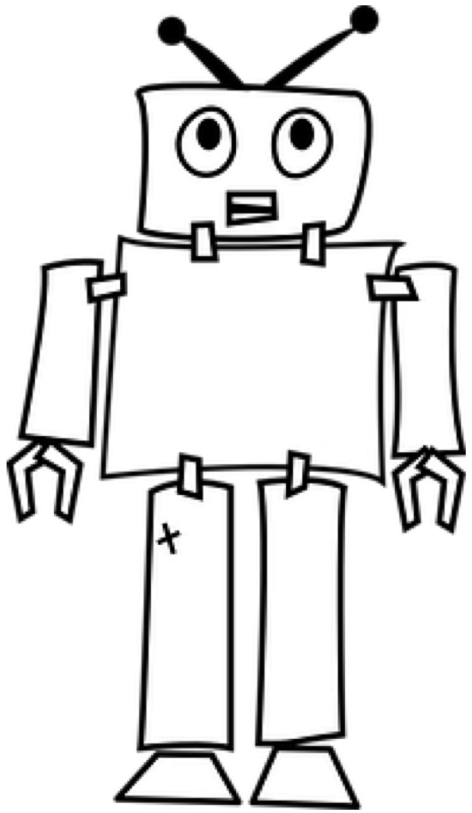


Image courtesy of pixabay.com

# Topic agenda

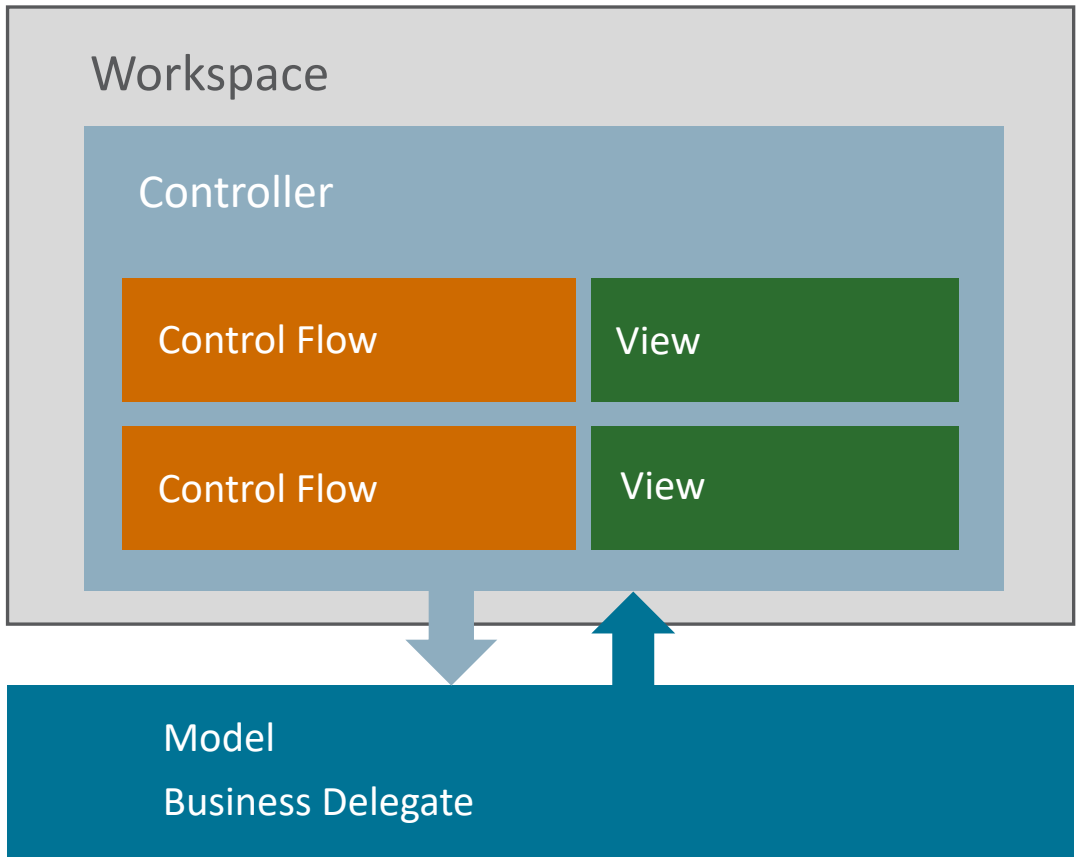
1. Architecture matters
2. Skill patterns
3. Digital assistant pattern
4. Skill Parameters
5. QnA

# About skills

- Skills are units of work
  - Assist users in completing a conversational task
  - Access remote services and backends
  - Do not make assumptions about the existence of user scope variables
- Scope of a skill can be
  - single use case
  - multiple use cases
  - complete business solution
- Supports modularization

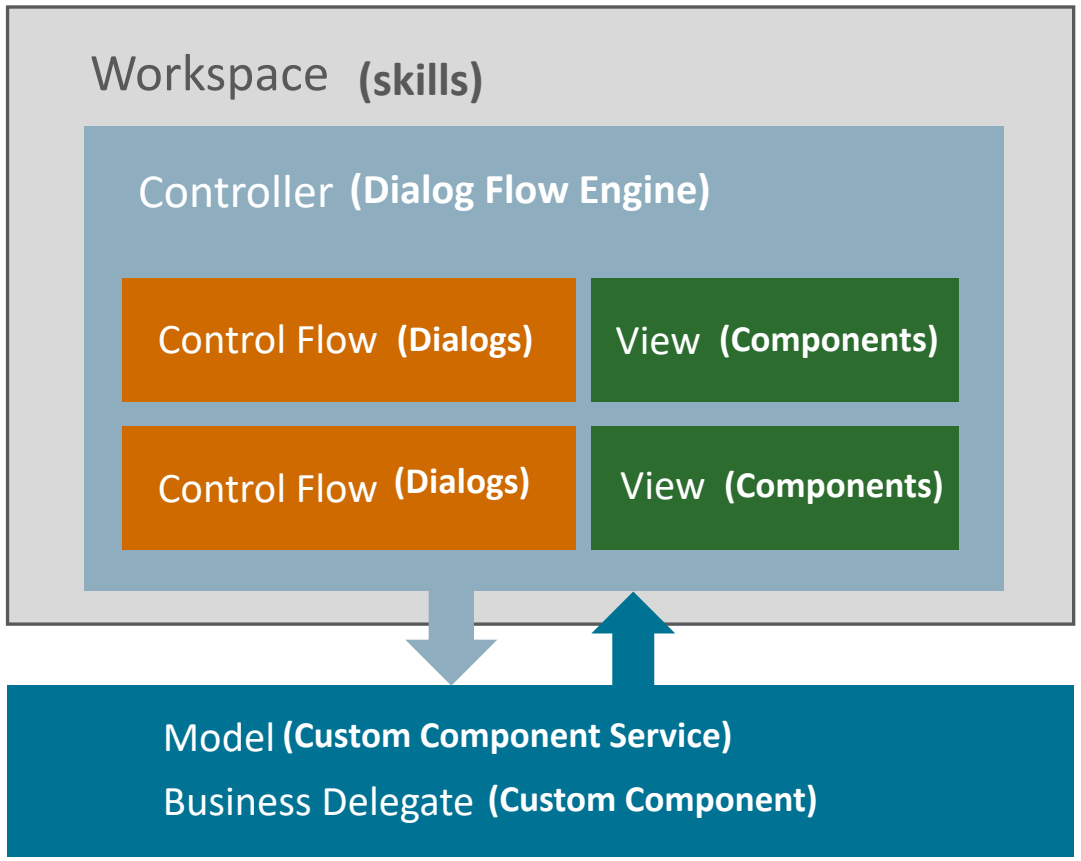
# Thinking in patterns - MVC

- Workspace
  - Holds project code, libraries
- Controller
  - Navigates UI and holds state
- View
  - Renders application response
- Model
  - Data and business service access
- Business delegate



# Thinking in patterns – Oracle Digital Assistant

- Skill (Workspace)
  - Holds conversations, intents, utterances and custom logic
- Dialog Flow Engine (Controller)
  - Navigates between dialogs, holds state
- Components (View)
  - Renders bot responses
- Custom Component Service (Model)
  - Data and business service access
- Custom component (Business delegate)



**Good design** starts with sound decisions

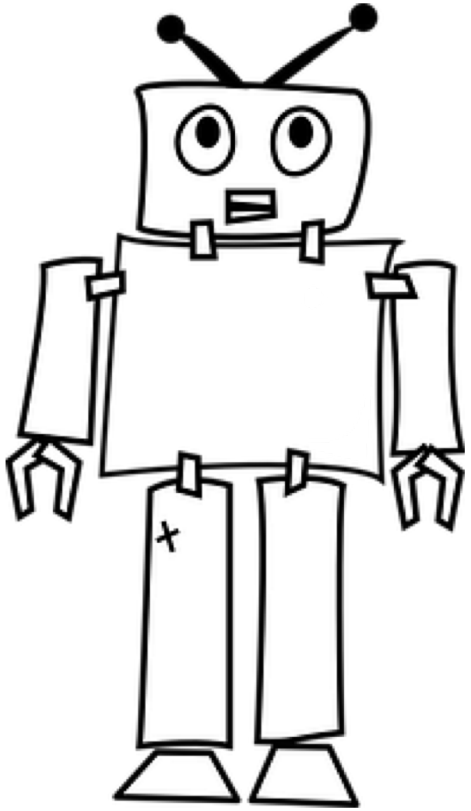


Image courtesy of pixabay.com

# Architectures

## All-in-one

- Skill as complete business solution
  - Finance, Pizza, Retail
- Skills as stand-alone solutions
- Reusability not a primary concern
- Skills are most likely built by different teams
  - Risk of inconsistent behavior and look

## Part-of-a-whole

- Skill as part of a whole
  - Member registration, course booking, message board, meeting organizer
- Small skills built with reuse in mind
- Very likely built by same team
  - Easier to enforce design principles and guidelines for a consistent appearance

Breaking down a monolithic skill into smaller skills **improves modularity**. It does not eliminate possible ambiguity in the utterances.

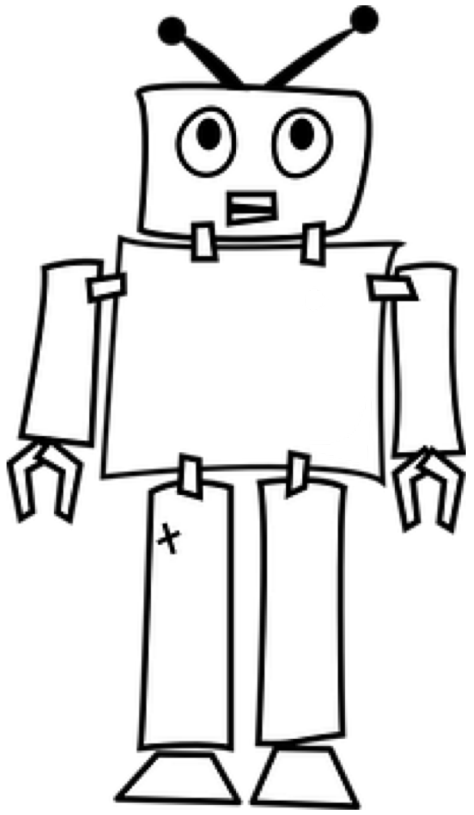
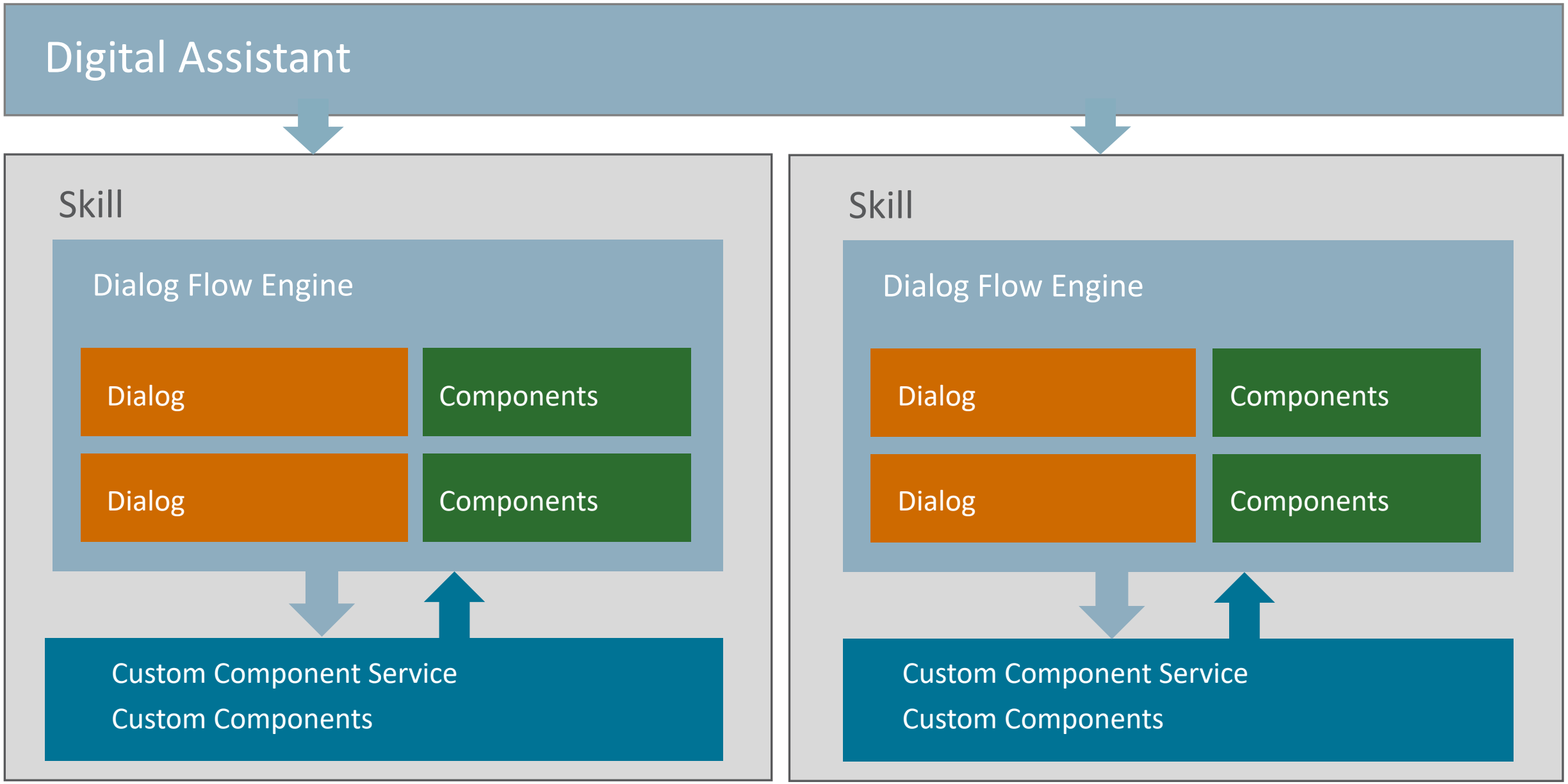
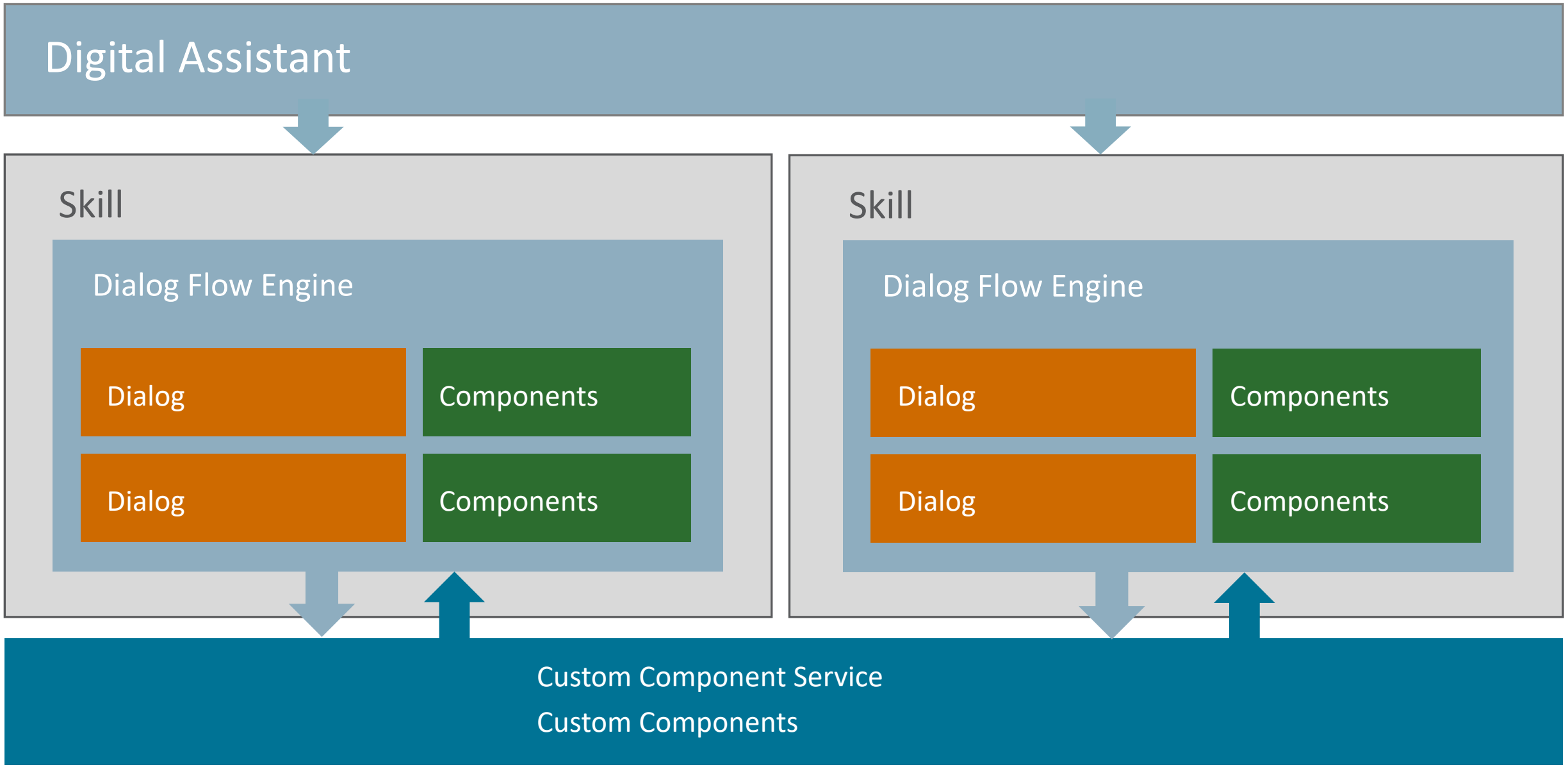


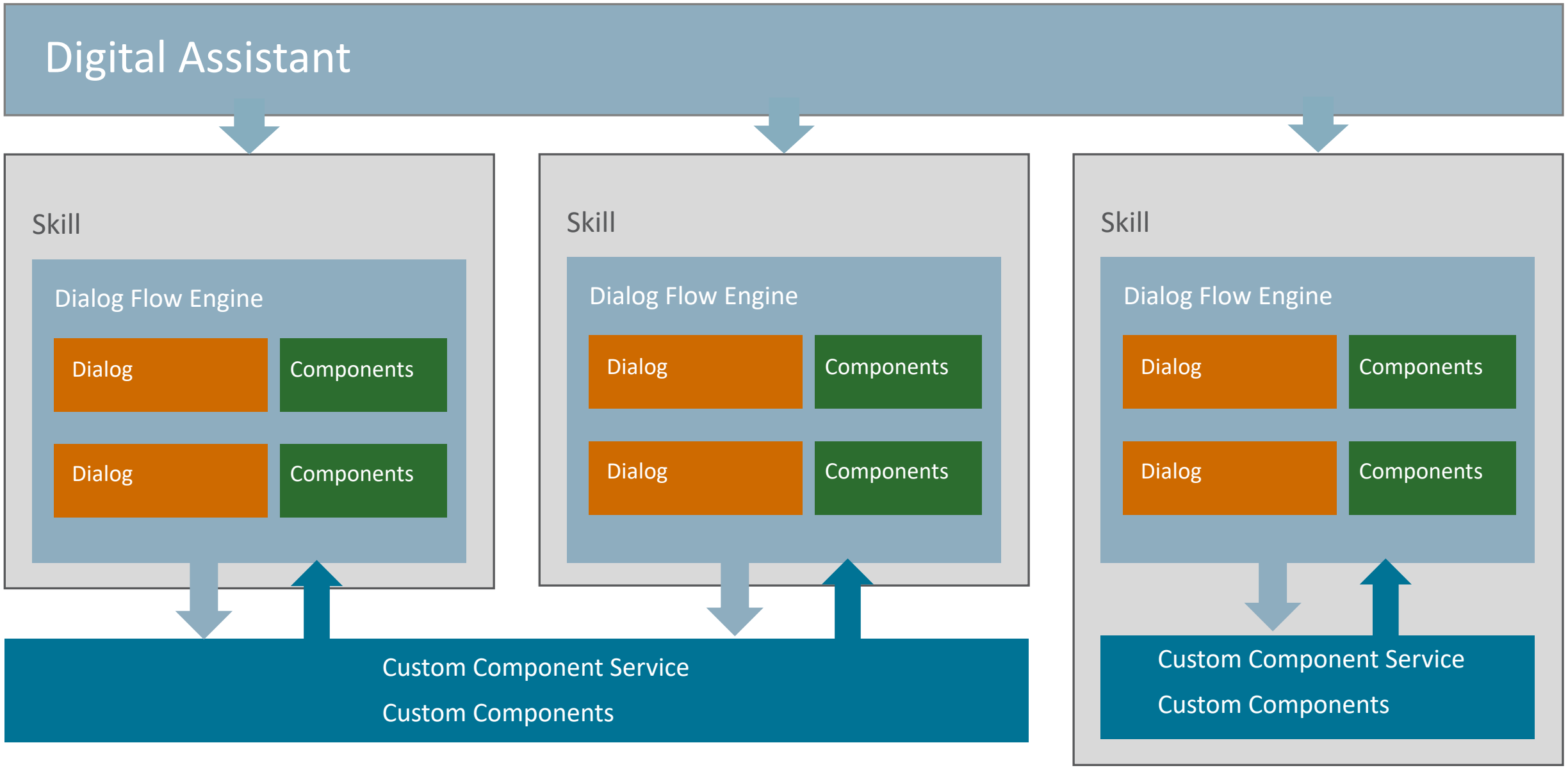
Image courtesy of pixabay.com

# Design Patterns

## Organizing Skills







# Design Patterns

## Custom Components

You build custom components for **backend integration** and to implement custom logic.

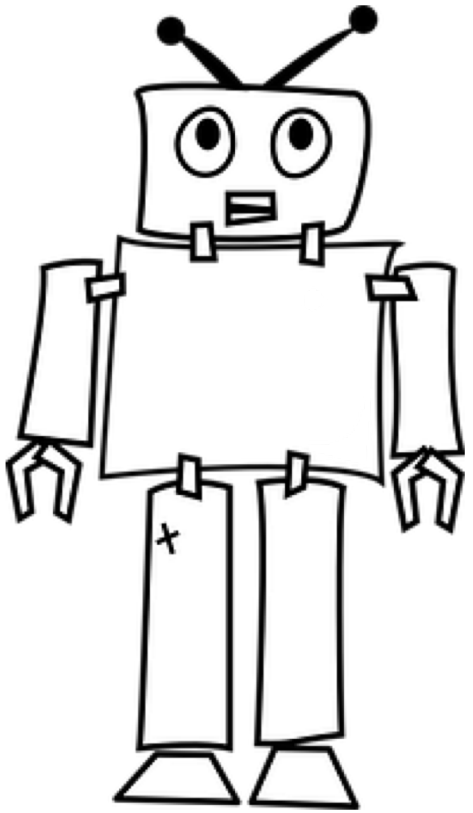


Image courtesy of pixabay.com

# Custom component deployment considerations

## Exclusive Component Service

### Local Component Container

- Skills can run different versions of a component
- Breaking a component in an update does not impact all skills but only those updated to the new version
- Error correction requires updating all deployments

- No credential store
- Component code exported with skill

## Shared Component Services

### Remote Node Servers

- Component source code resides on server
- Option to share code with other applications
- Remote server may act as a data and service integration layer
- Single point of development and maintenance

- Failure may impacts multiple skills

### Mobile Hub

- Multi channel backend service
- Provides platform services and declarative service connectors
- Secure container
- Storage options

### 3rd Party Node Servers

- Allow use of environment variables (configuration)

There is **no limit to the number of custom components** used in a skill. You can even use combinations of locally and remotely deployed components.

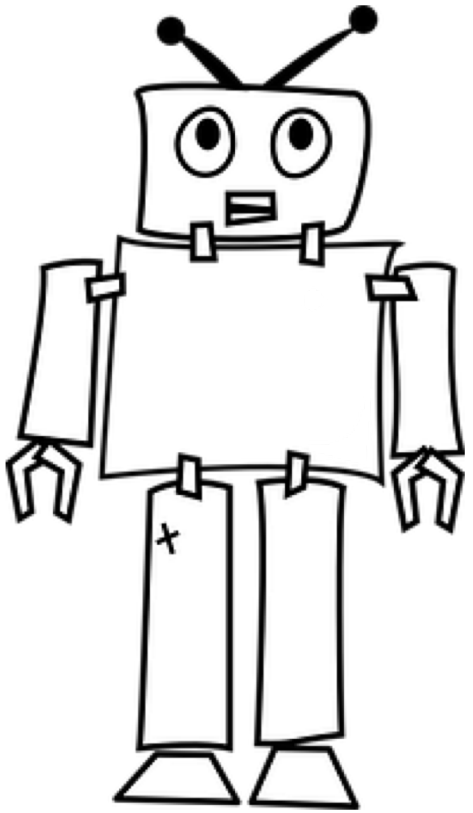


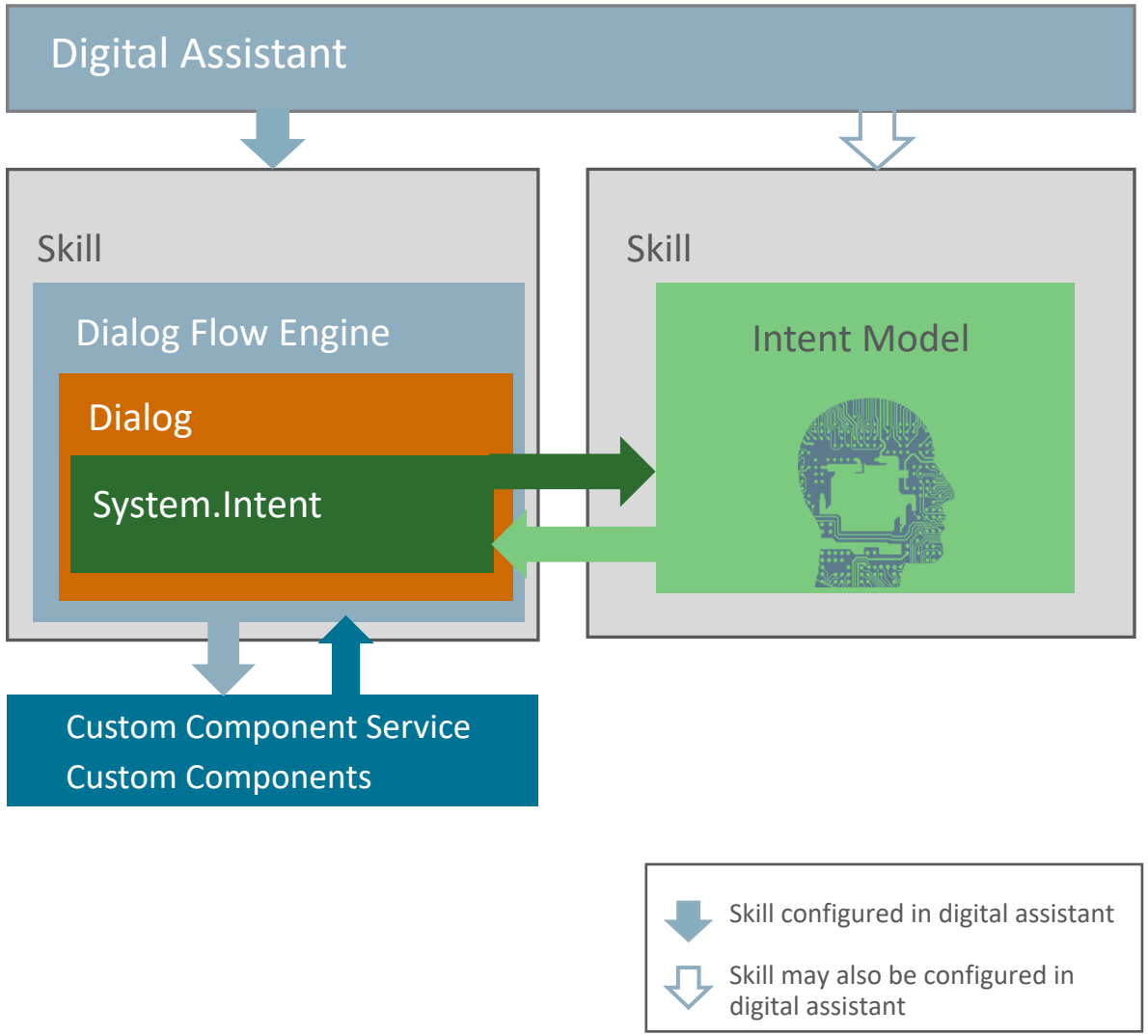
Image courtesy of pixabay.com

# Design Patterns

## Referencing Skills From Skills

# Interskill referencing

- A skill's `System.Intent` state may reference another skill's intent model
- Allows one skill to share another skill's intents
- Resolved intent name and entities are returned to calling skill
- Use cases
  - Common intents (train once)
  - Use of different intent engine



Interskill referencing does not execute the referenced skill's dialog flow. All intent handling happens in the skill that references the intent model of another skill.

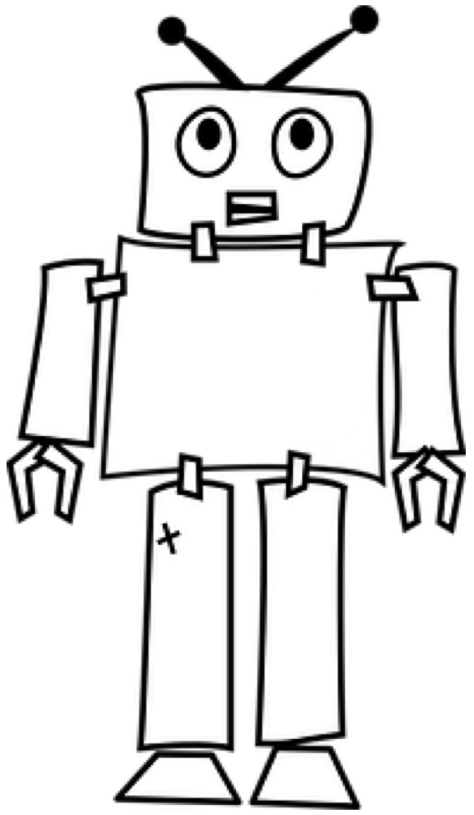
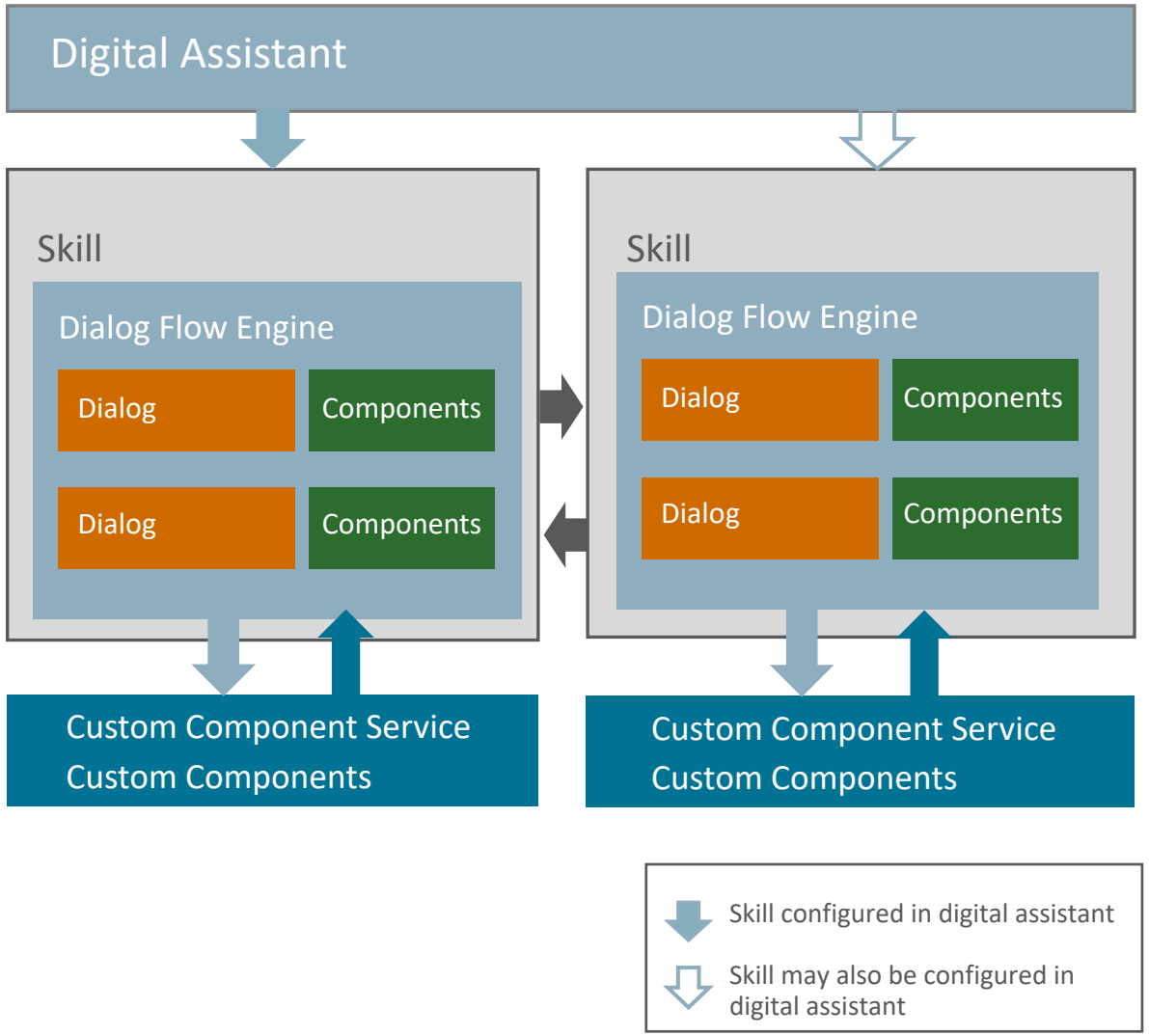


Image courtesy of pixabay.com

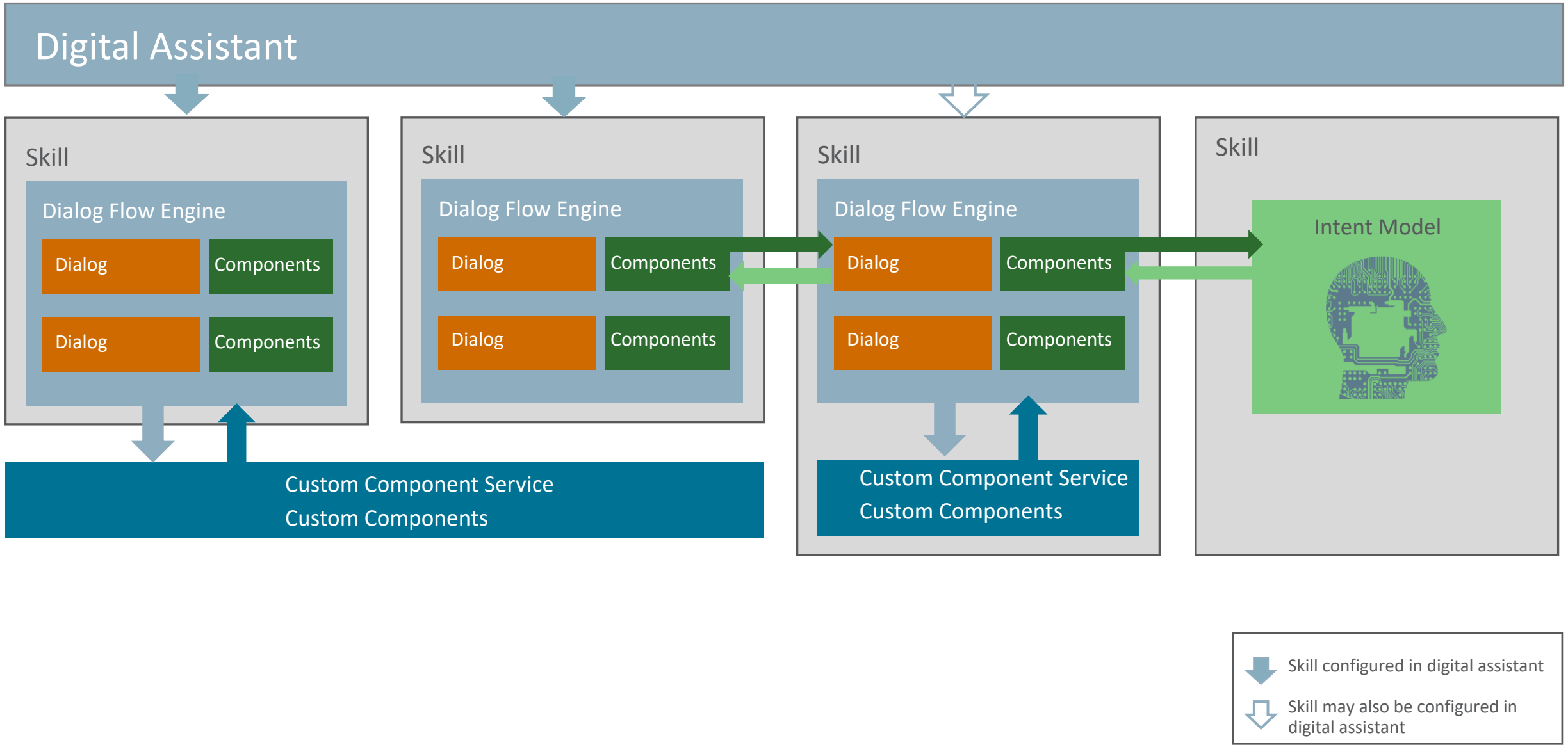
# Skill calling skill

- Calling skill
  - Uses `System.InvokeBot` component
  - Passes input parameters
  - Defines dialog flow state to call
  - Can receives return values and action
- Called skill
  - Optionally has input parameters defined and mapped to variables
  - May return action string



# Skill calling skill vs. skill exposure in digital assistant

- Skill-calling-skill
  - Weakly implementation of encapsulation
  - Intents and utterances not classified in digital assistant intelligent router
  - Allows skill chaining
    - Skills can be called in specific order based on use case requirement
    - E.g. skill A calls skill B calls skill C
  - Promotes modularization
- Skill exposed on digital assistant
  - Skill directly accessed through digital assistant routing
  - Skill intents and utterances classified by digital assistant router



# Topic agenda

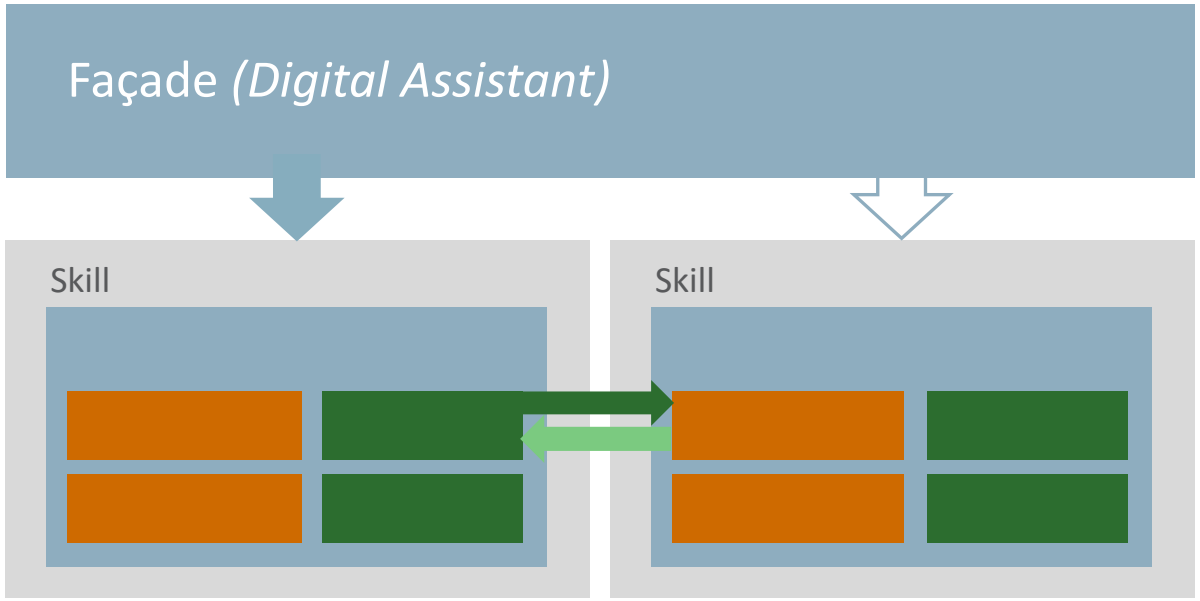
1. Architecture matters
2. Skill patterns
3. Digital assistant pattern
4. Skill Parameters
5. QnA

# About digital assistant

- Front-end bot that redirects user messages to one of its configured skills
  - Routing is based on intents and utterances, context and direct addressing
- Disambiguates user messages if required
- Exposed on one or many messenger channels
  - Messenger payload differences handled by configured channel connector
- Design time for digital assistant designers
  - Create chatbots by orchestrating individual skills
    - Configuration only (no coding)
  - Digital assistant designers may or may not be the skill developer

# Thinking in patterns: Digital Assistant

- Façade
  - Single entry point for a bot
  - Skills "hidden" from user view
  - Skills can be added, removed or updated without re-deployment



**Define a clear goal for your digital assistant** that is different from "all you can eat".

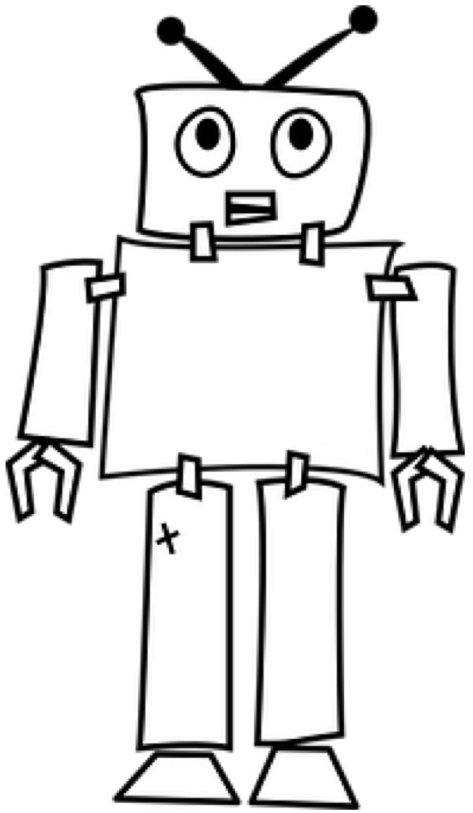
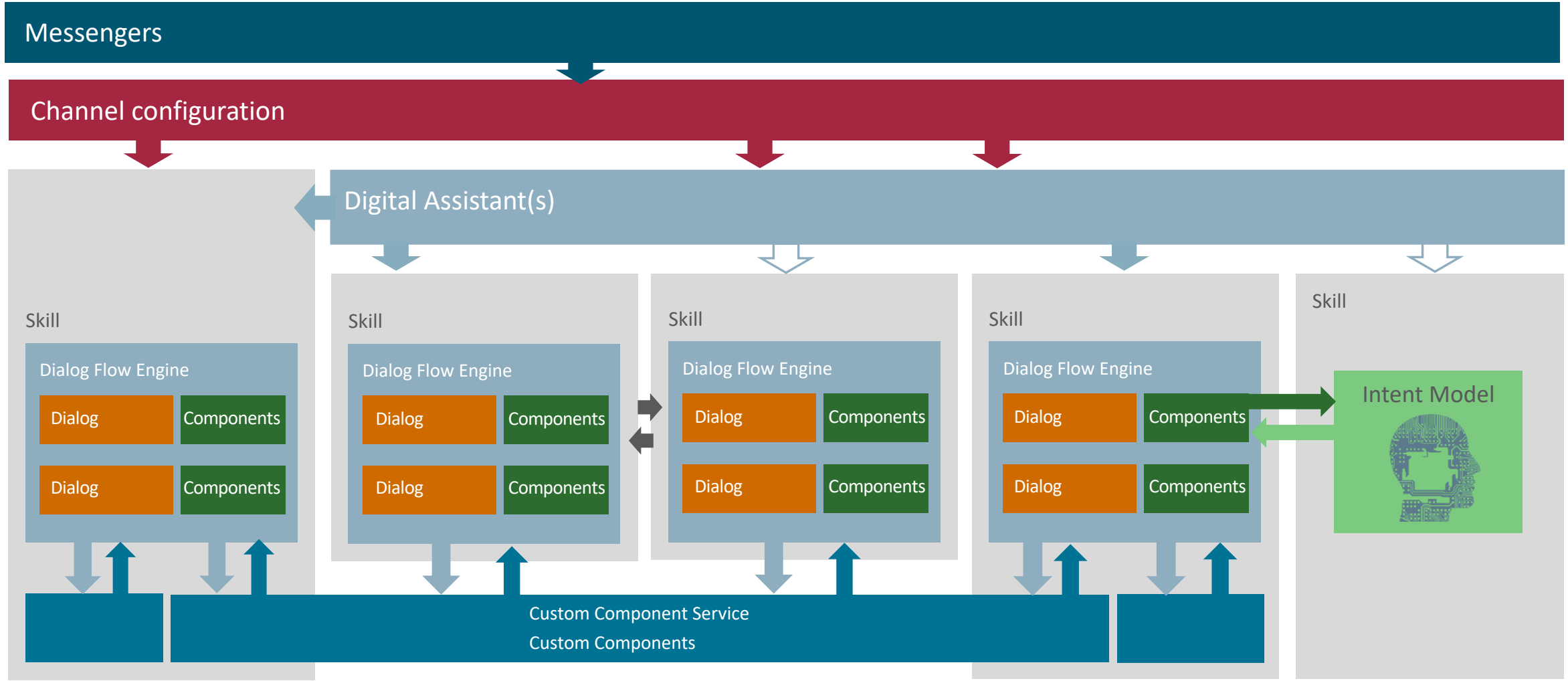


Image courtesy of pixabay.com

# Oracle Digital Assistant architecture possibilities



Identify an architecture that best suits the usecase to be implemented.

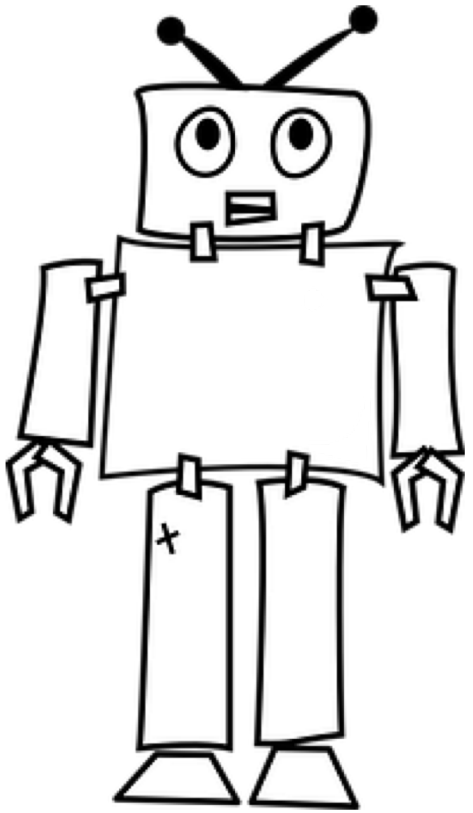


Image courtesy of pixabay.com

# Topic agenda

1. Architecture matters
2. Skill patterns
3. Digital assistant pattern
4. Skill Parameters
5. QnA

You can "**remote control**" **skills** through skill parameters

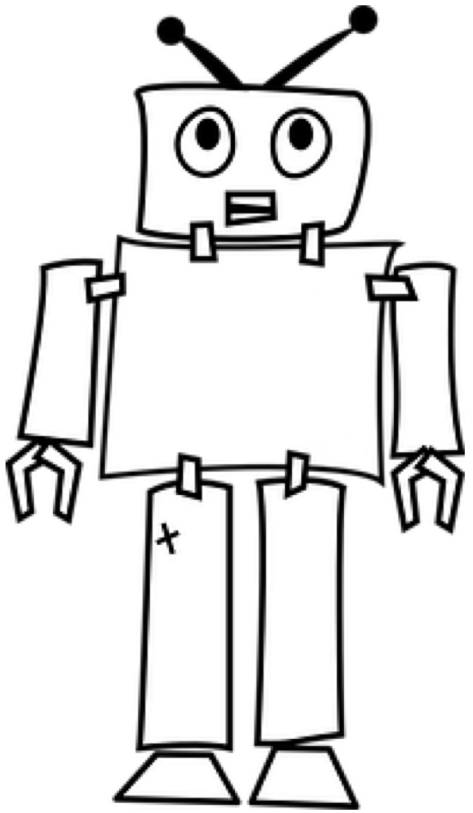
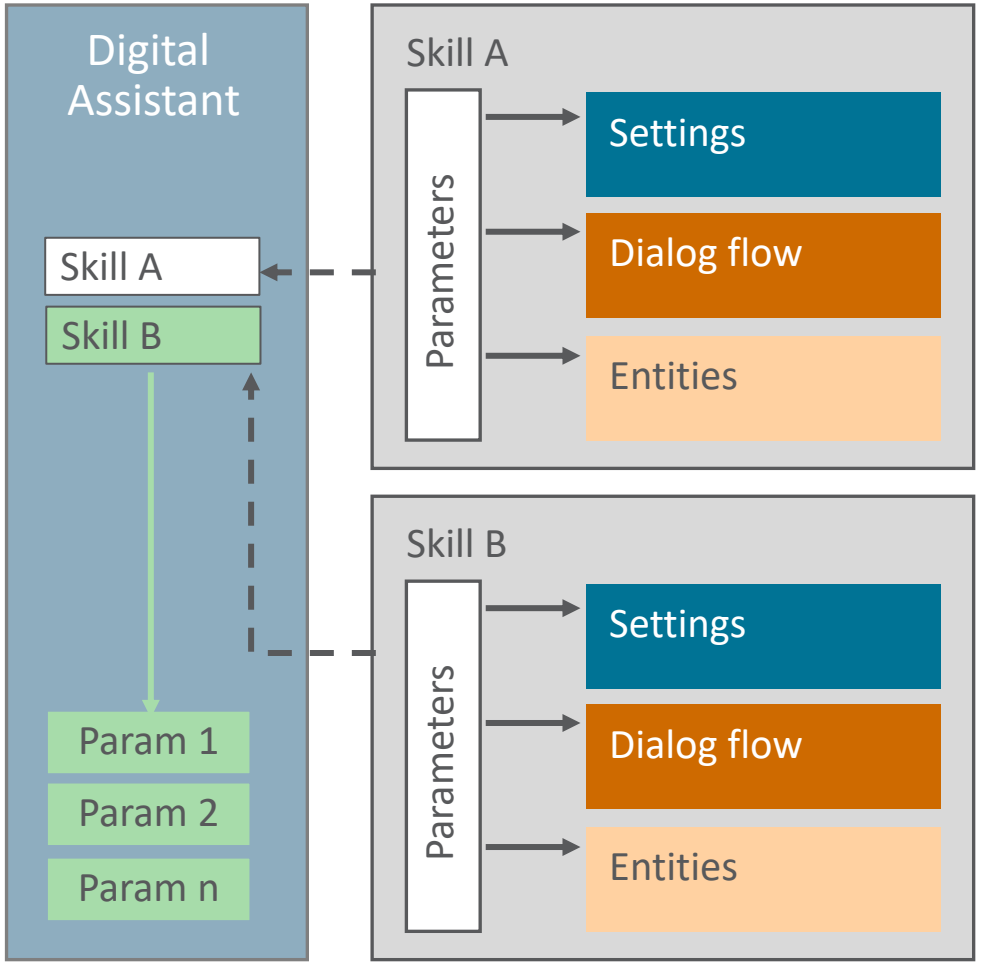


Image courtesy of pixabay.com

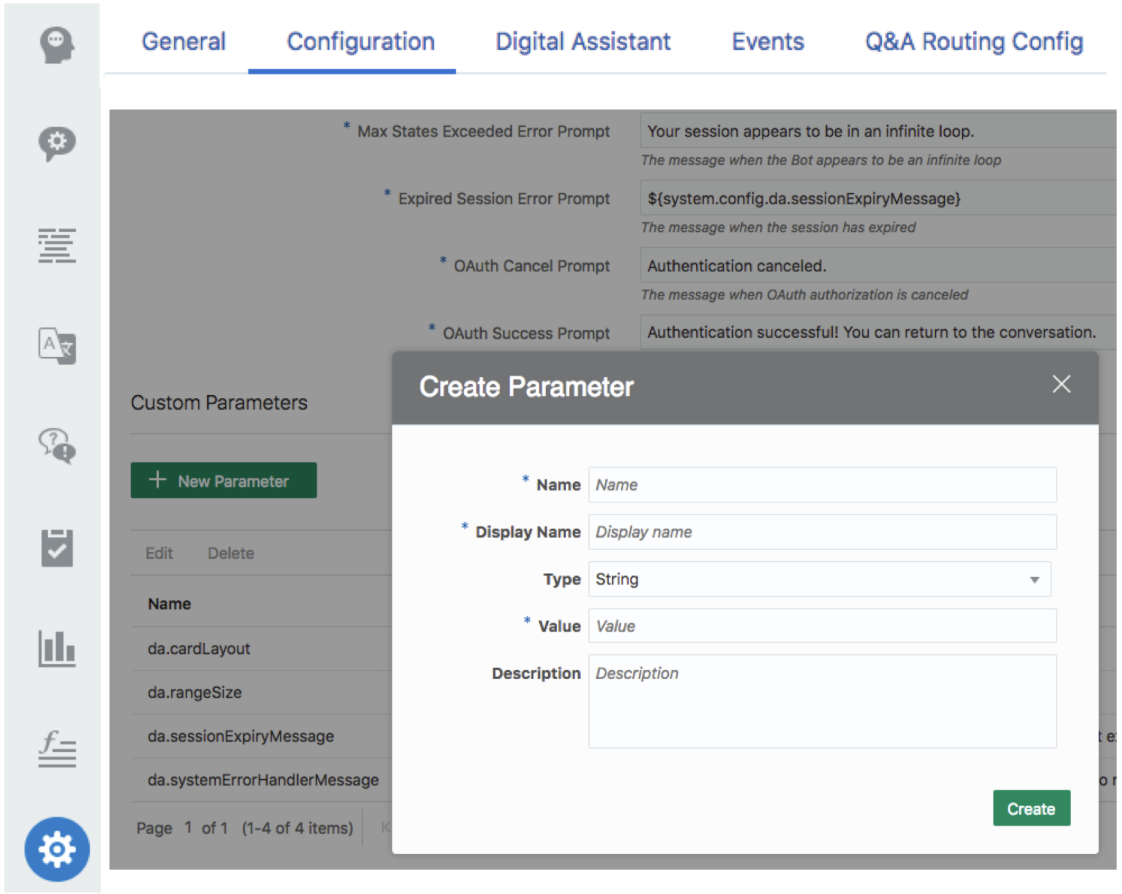
# Exposing skill parameters to digital assistant

- Improves user experience through consistency
  - Consistent messages
  - Similar behaviors and looks
- Defines a contract between a skill and the digital assistant
  - Improves reusability



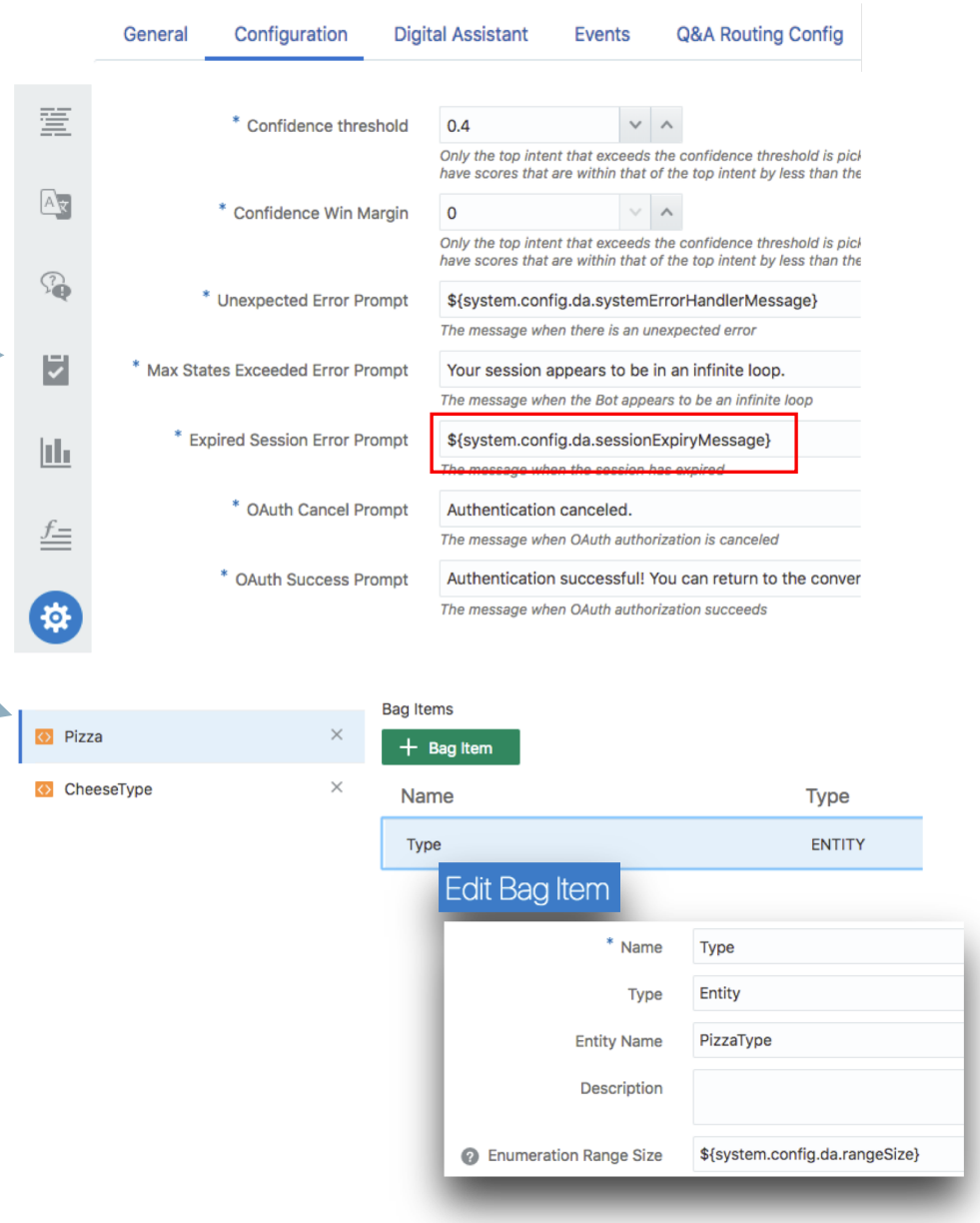
# Defining custom parameters for a skill

- Custom parameters are created in the *Settings* panel of a skill
- Supported data types are string, integer, float and boolean
- Parameter name must start with "*da.*" prefix to be visible in digital assistant
- Digital assistant exposes parameters in skill configuration panel



# Accessing parameters in skill

- Skill parameters are accessible from
  - Skill's settings panel
  - In Entities
  - In dialog flow
- Read access
  - `${system.config.<name>}`
- Write access
  - `System.SetVariable`
    - variable: "system.config.<name>

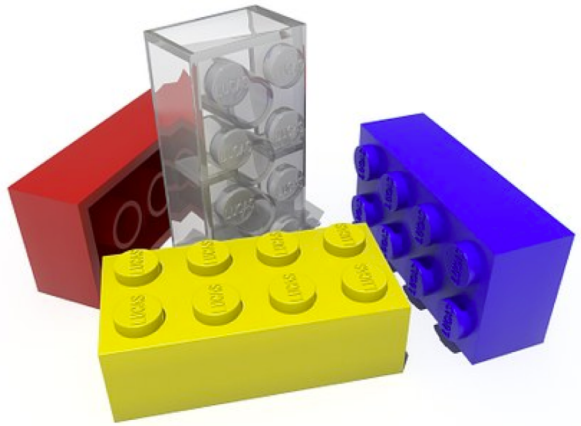


# Topic agenda

1. Architecture matters
2. Skill patterns
3. Digital assistant pattern
4. Skill Parameters
5. QnA

# Options for using QnA

- Each skill has its on QnA as required
  - Modular and simplest option
- Interskill QnA referencing
  - Calling QnA defined in a different skill
    - E.g. route to a specific QnA based on conversation context
  - Using `System.Intent` and `System.Qna` properties botName, botVersion, qnaBotVersion and qnaBotName
- Separate QnA only skill used by digital assistant
  - FAQ (QnA skill) may be shown upfront in welcome message
  - QnA maintained in single place without impacting other skills

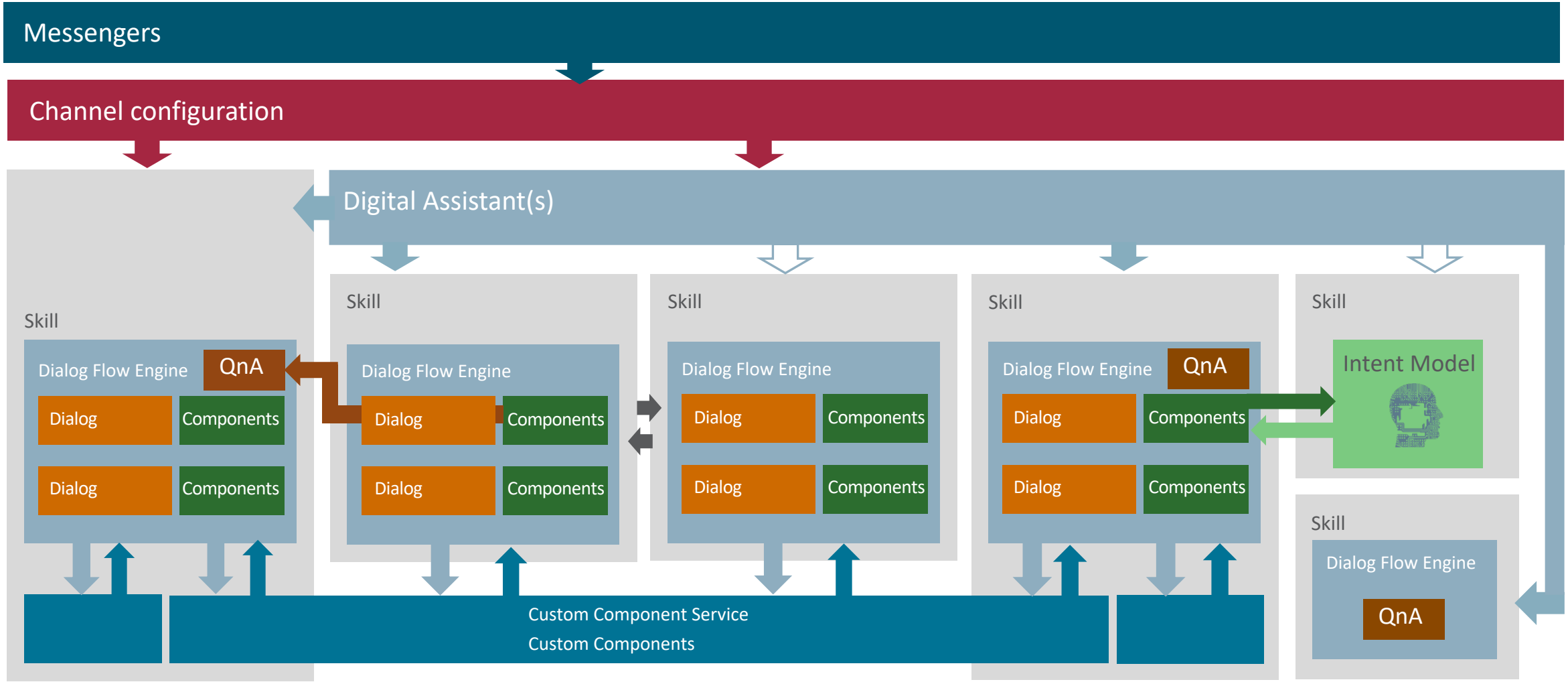


# Comparison of approaches

- QnA defined in each skill
  - QnA content considered in digital assistant routing
- Interskill QnA referencing
  - More control to the user the way he/she wants to invoke QnA
  - Ability to show different QnA for different users
  - QnA content not considered in digital assistant routing

- Creating separate QnA only skill and consuming in DA
  - QnA skill will be shown upfront in welcome message!
  - No additional code required since the routing will be managed by DA
  - QnA questions and answers contribute to classification in routing

# Oracle Digital Assistant: the completed image



# Integrated Cloud

## Applications & Platform Services


ORACLE®