
ORACLE®

Oracle Digital Assistant

The Complete Training

Dialog Flow

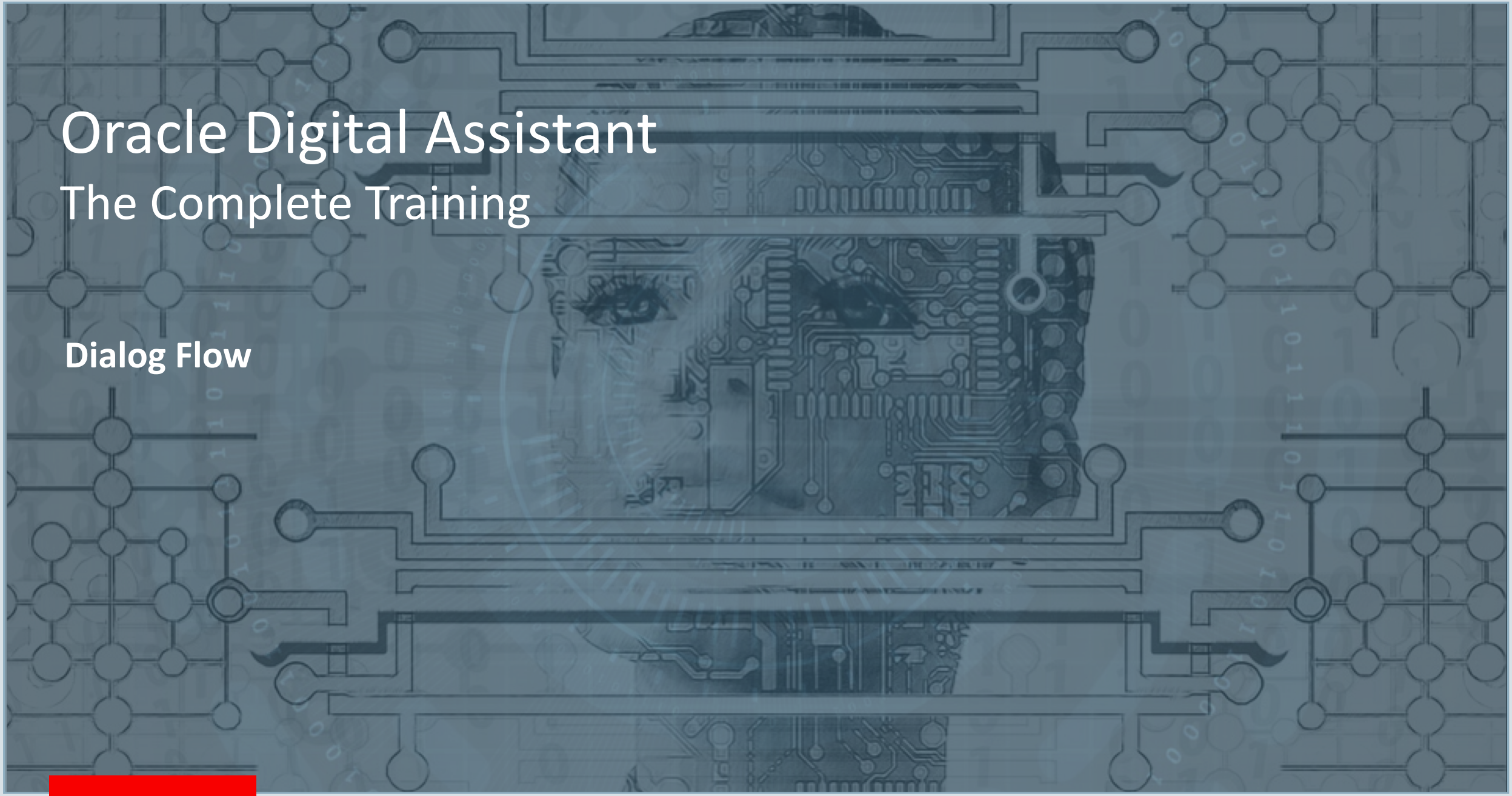


Image courtesy of pixabay.com

Safe Harbor Statement

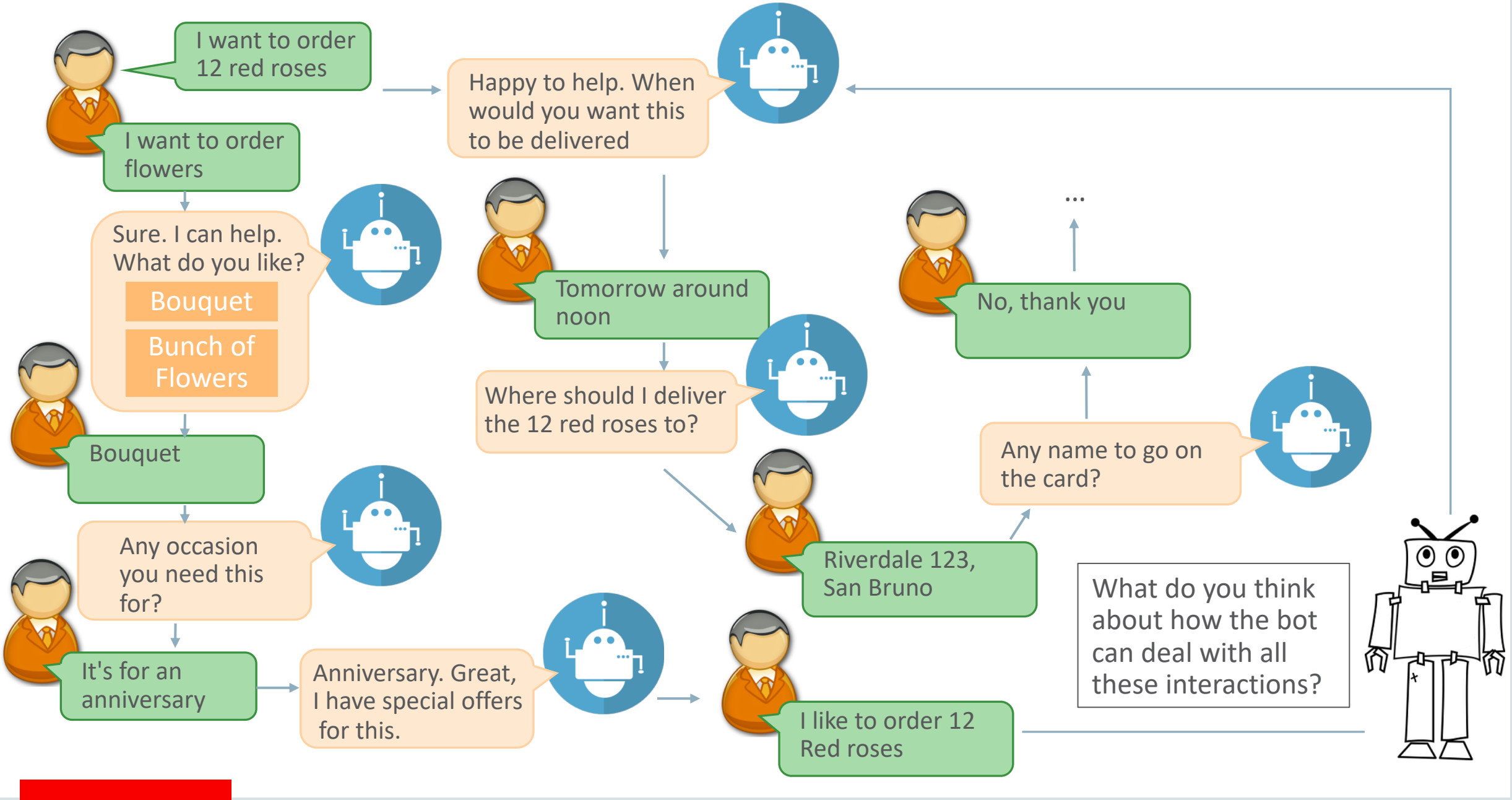
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development,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

Topic agenda

1. Conversation flow
2. Dialog flow
3. Variables
4. Components
5. Navigation
6. Embedded conversation tester

Topic agenda

1. Conversation flow
2. Dialog flow
3. Variables
4. Components
5. Navigation
6. Embedded conversation tester



Conversation flows are similar to screen flows in web and mobile applications in that they **define the interaction between a bot and a user** for a specific task

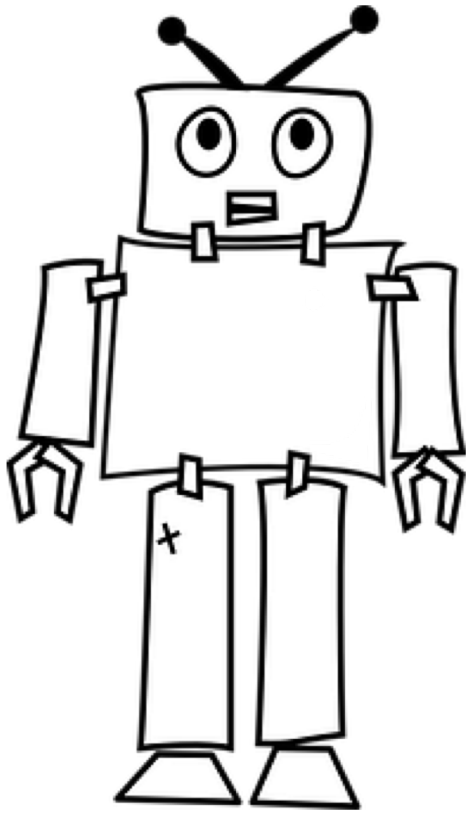


Image courtesy of pixabay.com

Oracle Digital Assistant dialog flow builder

- Conversation is design as a series of states
- Conversation flows from top to bottom by default
- Each state does one thing
  - Get user input, set variable, output, resolve entities, branch etc.
  - Each of which is implemented by a "component"
- Some components automatically take care of complex conversation logic
  - Composite bag entities
- Markup is called Oracle Bot ML (OBotML)

```
startOrderPizza:
  component: "System.Output"
  properties:
    text: "ok lets get that order sorted"
    keepTurn: true
  transitions: {}

resolveEntities:
  component: "System.ResolveEntities"
  properties:
    variable: "pizza"
    nlpResultVariable: "iResult"
    maxPrompts: 3
    cancelPolicy: "immediate"
  transitions:
    actions:
      cancel: "maxError"
      next: "setPizzaDough"

setPizzaDough:
  component: "System.SetVariable"
  properties:
    variable: "pizza.PizzaDough"
    value: "regular"
```

Topic agenda

1. Conversation flow
2. Dialog flow
3. Variables
4. Components
5. Navigation
6. Embedded conversation tester

This training session teaches **dialog flow fundamentals** using the dialog flow builder and OBotML.

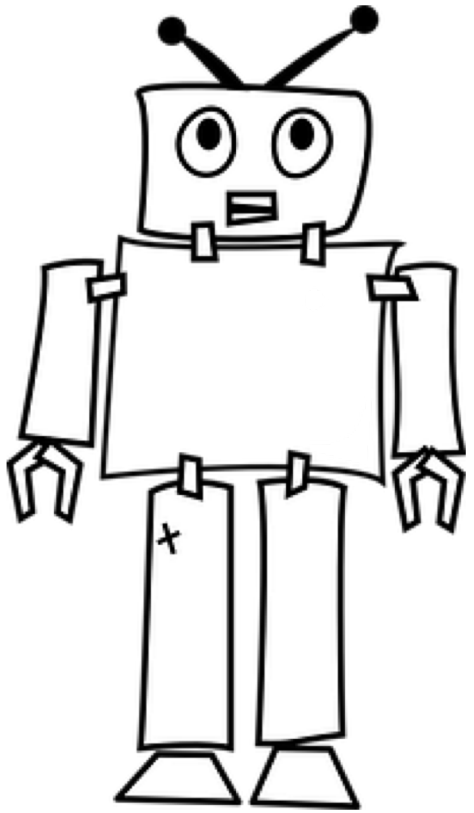
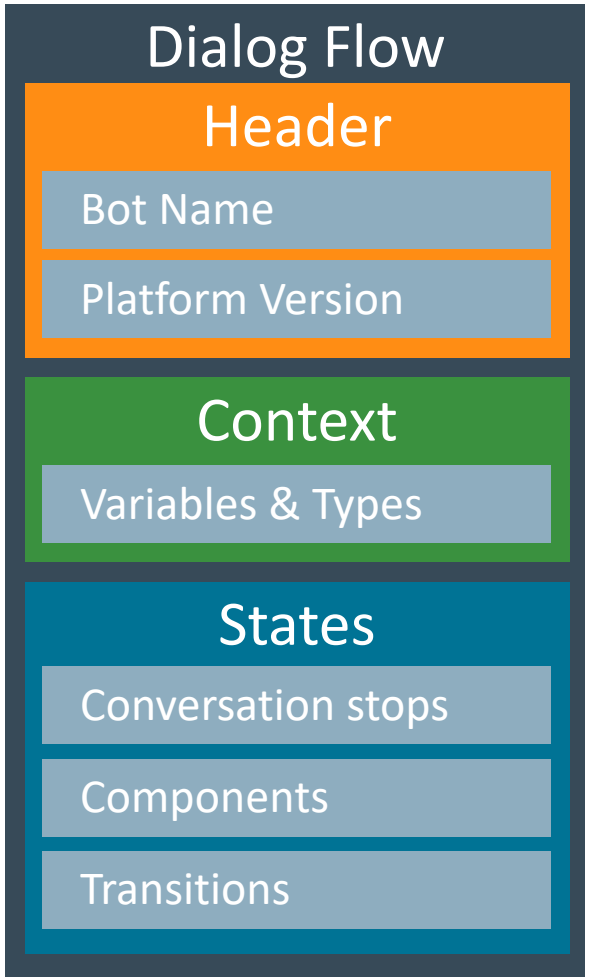


Image courtesy of pixabay.com

Dialog flow builder

- Three sections
  - Header, Context, States
- You can define variables
- Each state implemented by a component
  - Executes logic
  - Receives user input
  - Returns bot responses
  - Determines navigation




```
metadata:
  platformVersion: "1.0"
main: true
name: "FinancialBotMainFlow"
context:
  variables:
    accountType: "AccountType"
    txnType: "TransactionType"
    txnSelector: "TransactionSelector"
    toAccount: "ToAccount"
    spendingCategory: "TrackSpendingCategory"
    paymentAmount: "CURRENCY"
    iResult: "nlpresult"
    iResult2: "nlpresult"
    transaction: "string"
    dispute: "string"
    amount: "string"
    merchant: "string"
    date: "string"
    description: "string"
states:
  intent:
    component: "System.Intent"
    properties:
      variable: "iResult"
      confidenceThreshold: 0.4
    transitions:
      actions:
        Balances: "startBalances"
        Transactions: "startTxns"
        Send Money: "startPayments"
        Track Spending: "startTrackSpending"
        Dispute: "setDate"
        unresolvedIntent: "unresolved"
  startBalances:
    component: "System.SetVariable"
    properties:
      variable: "accountType"
      value: "${iResult.value.entityMatches['Ac
    transitions: {}
```

Topic agenda

1. Conversation flow
2. Dialog flow
3. Variables
4. Components
5. Navigation
6. Embedded conversation tester

Context variables

- Hold information for the duration of a conversation
  - Variables can explicitly be reset
  - Automatically reset when conversation starts over
- Valid types
  - string, boolean, int, float, double
  - resourcebundle
  - nlpresult, entity
  - Arrays are created from variables of type string
- Accessible through expression
  - ${variable_name.value}

```
context:
  variables:
    name: "string"
    age: "int"
    rb: "resourcebundle"
    departureDate: "DATE"
    mailAddress: "MAIL"
    destination: "AirportCodes"
    iResult: "nlpresult"


printBookingConfirmation:
  component: "System.Output"
  properties:
    text: "Hello, ${name.value}.
    Your flight to ${destination.value}
    on ${departureDate.value.date} is
    confirmed."
  transitions:
    return: "done"
```

User variables

- Defined at runtime using a name prefix of "user. "
- Saved in digital assistant database
  - Variable persisted beyond a single conversation

```
#creating/setting user scope variable
setUserScopeVariableForLanguage:
  component: "System.SetVariable"
  properties:
    variable: "user.preferredLanguage"
    value: "${profile.languageTag}"


#reading user scope variable
printPreferredLanguage:
  component: "System.Output"
  properties:
    text: "User preferred language code set to ${user.preferredLanguage}"
```

Profile variables

- Set by messenger client
  - Content depends on messenger client
  - E.g. firstName, lastName, locale, timezoneOffset

```
#read and print profile variable content
welcomeStatement:
  component: "System.Output"
  properties:
    text: |-
          Welcome ${profile.firstName}, ${profile.lastName}.

          I am guiseppe the pizza bot. I am ready to take your order.

  transitions:
    next: "getUserOrder"
```

System variables

- Set by the skill
  - You don’t have to specifically declare
  - Used to access useful information about the skill and how it is functioning
- system.message
  - Access to JSON conversation payload
    - ${system.message.channelConversation.type}
      - Access to information about the messenger channel: 'test', 'facebook' 'webhook' etc.
    - ${system.message.messagePayload.text}
      - Access to user typed input value
- system.errorState
  - Name of dialog flow state causing an error

System variables

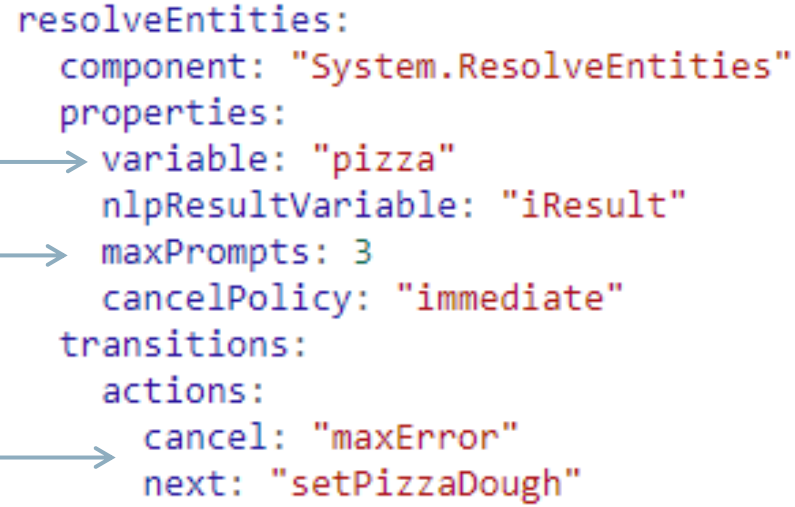
- system.invalidUserInput
  - Boolean flag set to true when user failed providing a valid input value
- system.conf.<name>
  - Access to custom skill bot configuration parameters
- system.entityToResolve
  - Used with composite bag entity to track the current entity to provide input for
- system.actualState, system.expectedState
  - Used in out-of-order message handling

Topic agenda

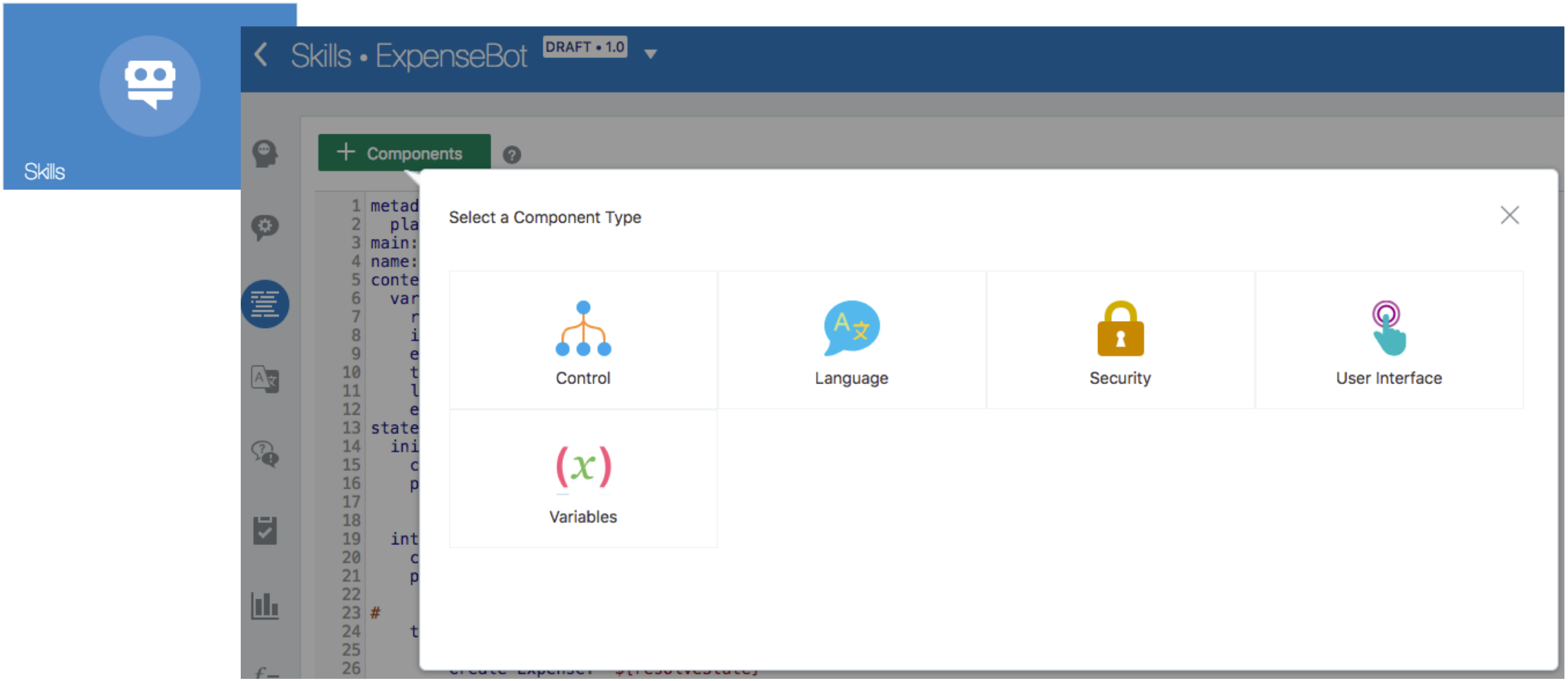
1. Conversation flow
2. Dialog flow
3. Variables
4. Components
5. Navigation
6. Embedded conversation tester

Components

- Do the real work
  - Every state references a component
  - Input components update variables
  - May have properties
  - Defines navigation
- System components
  - Provided by the Bots platform
  - Perform standard functionality and UI rendering
- Custom Components
  - Implemented by you the developer (typically for backend integration)



Component templates



Control

- System.ConditionEquals
- System.ConditionExists
- System.Switch

- System.DetectLanguage
- System.Intent
- System.MatchEntity
- System.QnA
- System.TranslateInput

Security

- System.OAuthAccountLink
- System.OAuth2AccountLink
- System.OAuth2ResetTokens

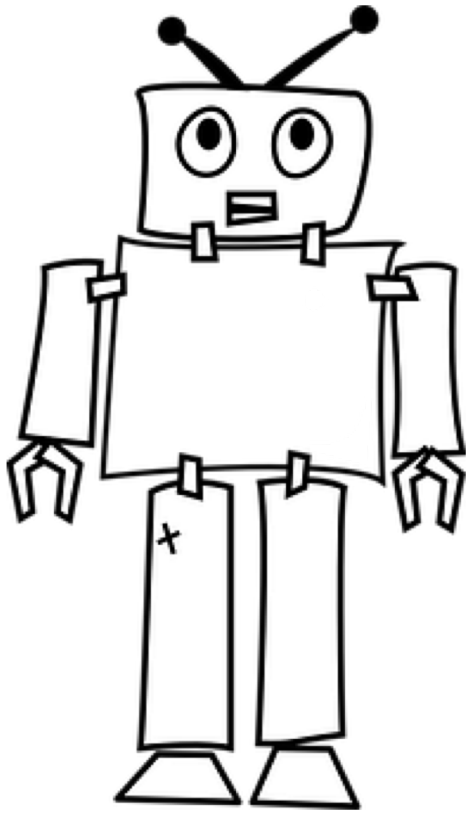
Variables

- System.SetVariable
- System.CopyVariables
- System.ResetVariables

User Interface

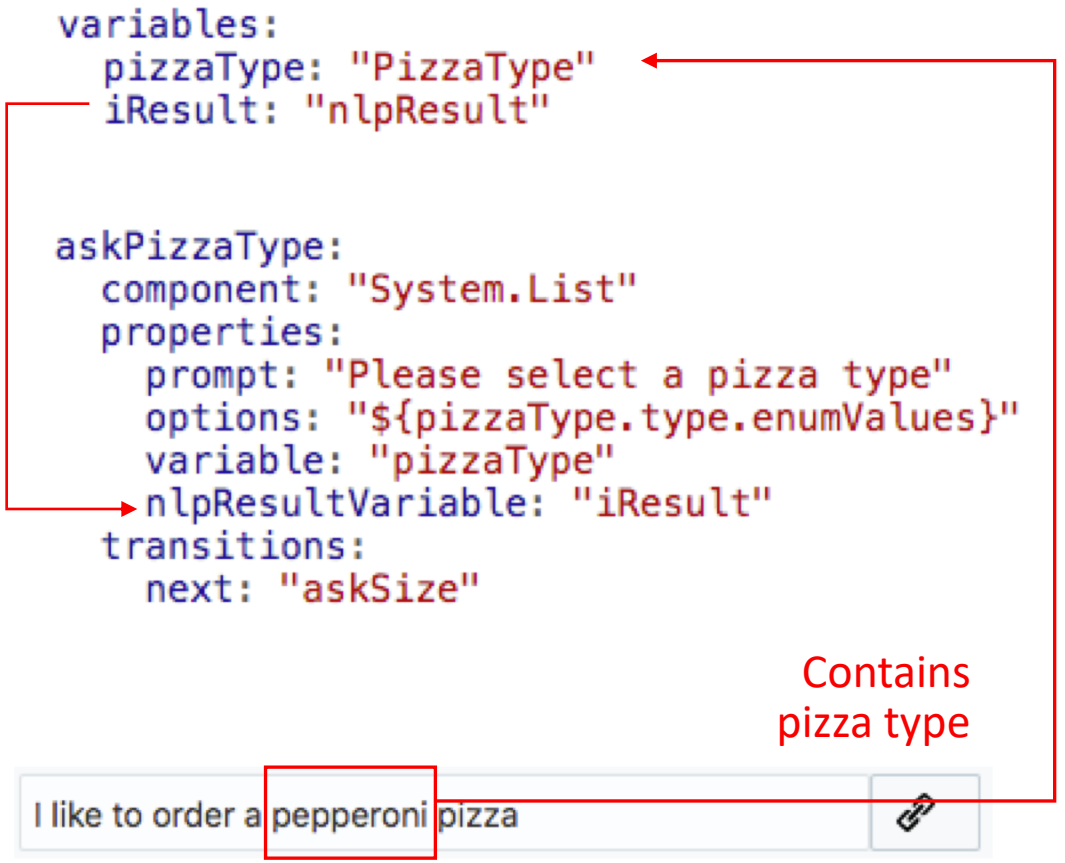
- System.AgentConversation
- System.AgentInitiation
- System.CommonResponse
- System.Interactive
- System.List
- System.ResolveEntities
- System.Text
- System.WebView

User input **components save user input** or user selection **in variables**. The result from NLP processing is saved in a variable of type nlpresult.



Components are not rendered when …

- The input component's *variable* property points to a dialog flow variable that has a value set
- A value is set through natural language processing (entity slotting)
  - Input component *nlpResultVariable* property points to a dialog variable of type 'nlpresult'
  - Component *variable* property references dialog flow variable of an entity type
  - Entity value got extracted using NLP



Entity slotting example

I like to order 12 red roses

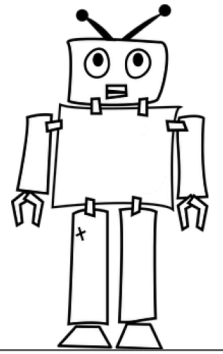
```
bouquetName: "Bouquets"
flowersName: "Flowers"
iResult: "nlpresult"
```

```
getUserIntent:
  component: "System.Intent"
  properties:
    variable: "iResult"
  transitions:
    next: "showMenu"
    actions:
      OrderFlowers: "startOrderFlowers"
      RequestAgentSupport: "startHumanAgent"
      unresolvedIntent: "resetiResult"
```

iResult.value.matchEntities['Flowers'][0]

Roses

```
showFlowersMenu:
  component: "System.CommonResponse"
  properties:
    variable: "flowersName"
    nlpResultVariable: "iResult"
    processUserMessage: true
    translate: "${useTranslationService.value}"
    metadata:
      responseItems:
      - type: "text"
```



Component **does not** render.

Entity slotting example

I like to order flowers

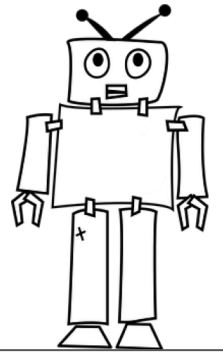
```
bouquetName: "Bouquets"
flowersName: "Flowers"
iResult: "nlpresult"
```

```
getUserIntent:
  component: "System.Intent"
  properties:
    variable: "iResult"
  transitions:
    next: "showMenu"
    actions:
      OrderFlowers: "startOrderFlowers"
      RequestAgentSupport: "startHumanAgent"
      unresolvedIntent: "resetiResult"
```

iResult.value.matchEntities['Flowers'][0]

NULL

```
showFlowersMenu:
  component: "System.CommonResponse"
  properties:
    variable: "flowersName"
    nlpResultVariable: "iResult"
    processUserMessage: true
    translate: "${useTranslationService.value}"
    metadata:
      responseItems:
      - type: "text"
```



Component **does** render.

Topic agenda

1. Conversation flow
2. Dialog flow
3. Variables
4. Components
5. Navigation
6. Embedded conversation tester

Navigation in a dialog flow is through state transitions.

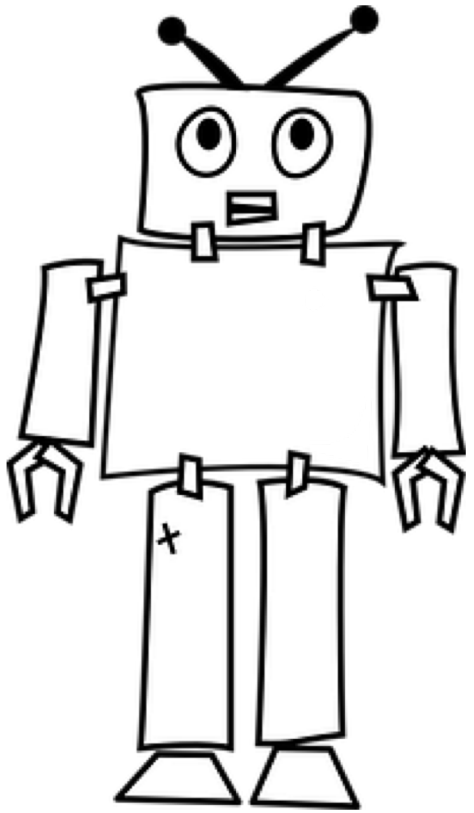
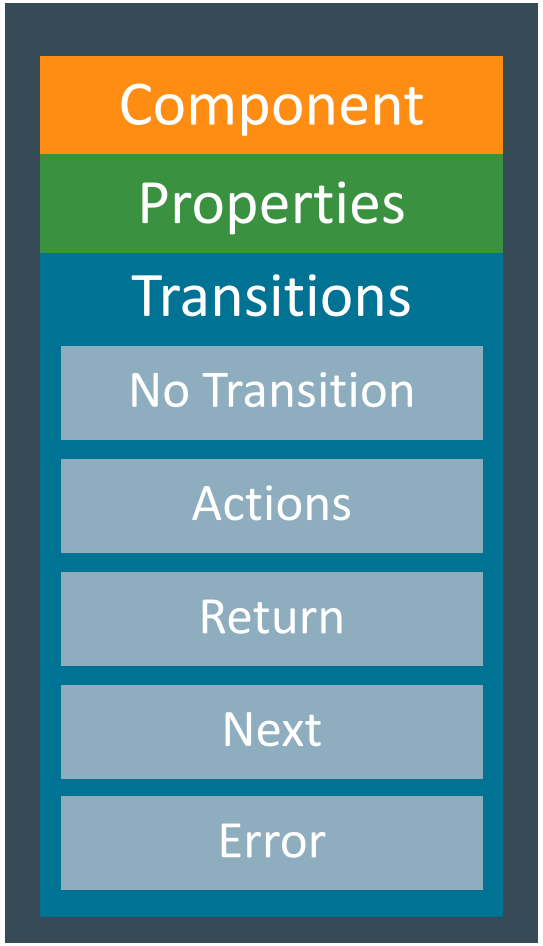


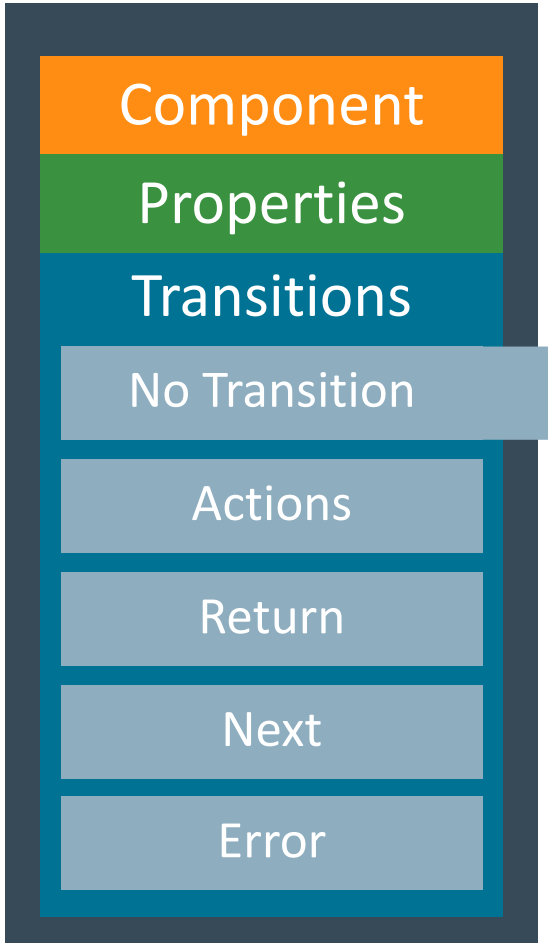
Image courtesy of pixabay.com

Transitions



- Transitions are directives that conditionally direct a conversation to a logical next dialog flow state
  - Transitions can be defined locally on a state or global as a default transition
- You can define one or more transition for a state
  - At any time, only a single transition is followed
  - Local transition definitions precede global transitions
  - Action transition preceded next, error and 'no' transition
- Allow branching in the conversation flow.

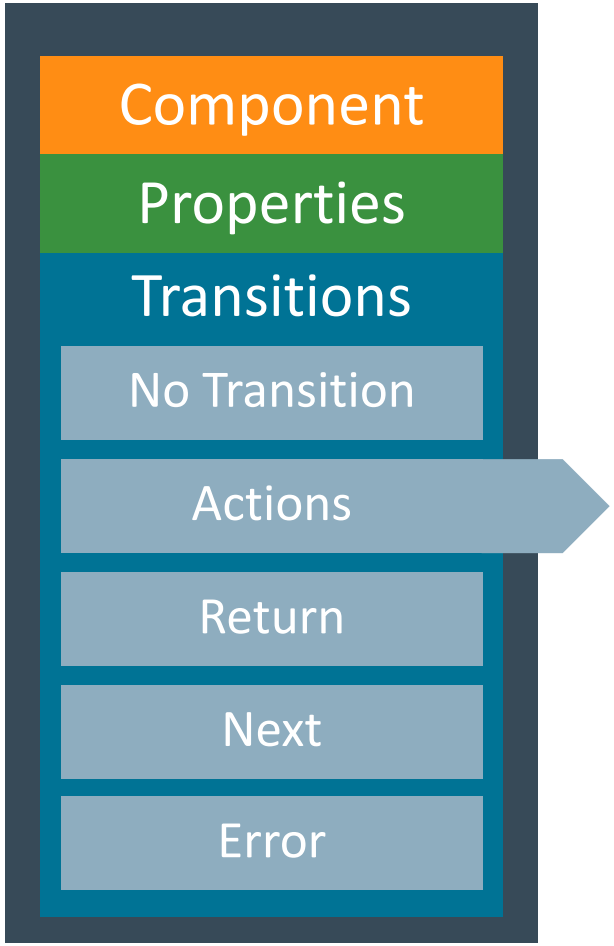
Transitions




```
sayHello:
  component: "System.Output"
    properties:
      text: " ... "
    transitions: {}
```

- Performs default navigation
  - Top-to-bottom
  - Useful for testing and mockups

Transitions




```
validateUserEntry:
  component: "System.MatchEntity"
    properties:
      …
    transitions:
      actions:
        match: "handleValidEntry"
        nomatch: "handleInvalidEntry"
```

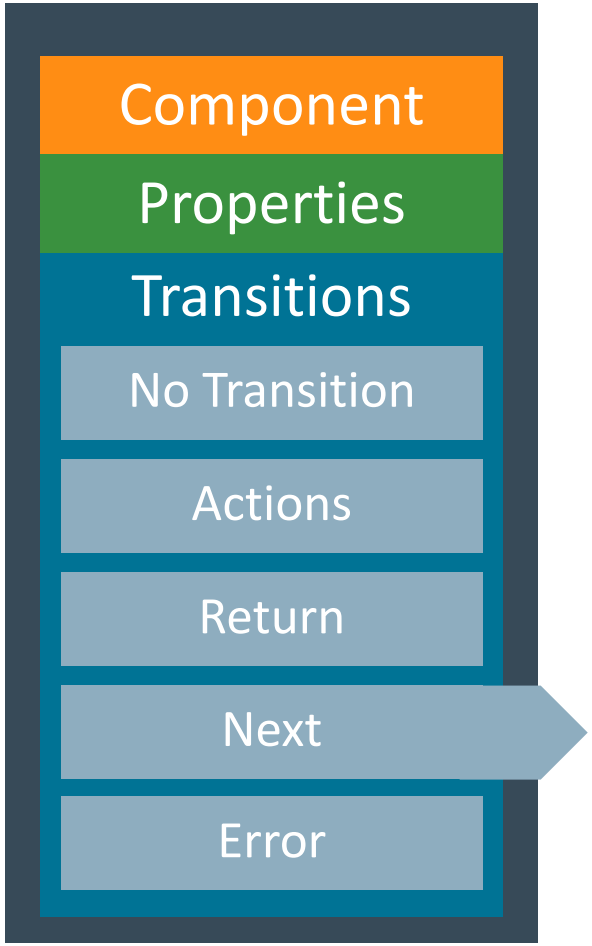
- Actions are
  - Component outcome strings
  - Mapped to a state in the dialog flow by skill designer

Transitions


```
printBalance:
  component: "System.Output"
  properties:
    …
  transitions:
    return: "done"
```

- Exits the flow
- Reset all flow context variables
- Entry back into flow begins back at the top state

Transitions




```
setCode:
  component: "System.SetVariable"
  properties:
    …
  transitions:
    next: "verifyCode"
```

- Proceeds to named state
- Recommended to use instead of "no transition"

Transitions


```
type:
  component: "System.Intent"
  properties:
   …
transitions:
    error: "handleError"
```

- If an unhandled exception occurs
  - e.g. a component exception
  - Local error handling

Default transitions ("global" transitions)

- Defined in dialog flow context
  - Optional (but useful)
- Allows you to define default navigation for :
  - actions
  - error transition
    - No more “Oops …" messages
  - next transition
- Used as a fallback when local state does not handle a transition type

```
variables:
  …

defaultTransitions:
  error: "globalErrorHandler"
  next: "globalNextState"
  actions:
    textReceived: "intentState"
    ...

states:
  ...
```

Topic agenda

1. Conversation flow
2. Dialog flow
3. Variables
4. Components
5. Navigation
6. Embedded conversation tester

Oracle Digital Assistant provides a **powerful embedded conversation tester** that can be used for testing and debugging conversation flows.

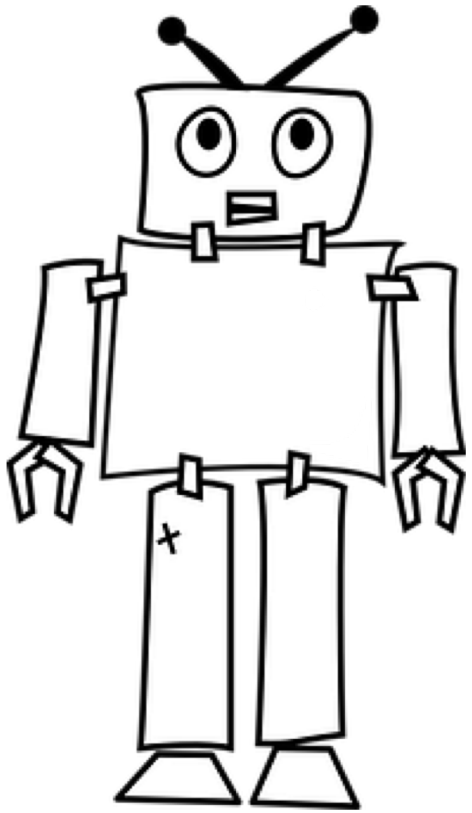
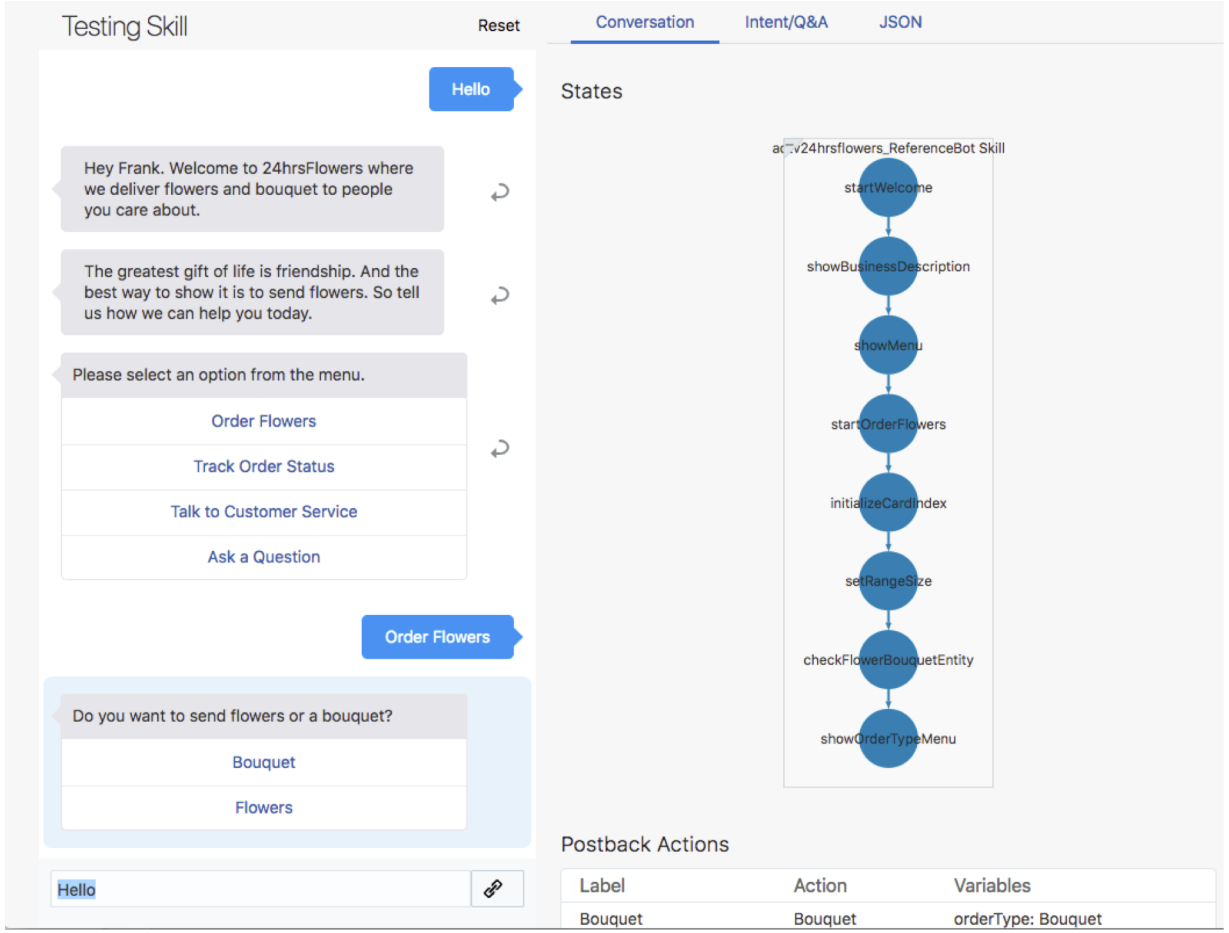


Image courtesy of pixabay.com

Embedded tester

- Allows you to test conversation flows without deploying the bot
  - Shows current state of system and context variables
  - Supports rich bot responses
    - Images, Cards, Lists, Attachments
  - "Circles" show visited states
- Shows Intent / Q&A resolution
- JSON payload
  - Shows raw message payload exchanged between user and bot



Integrated Cloud

Applications & Platform Services


ORACLE®