ORACLE

Implement a successful testing strategy

Program agenda

1. What is your testing strategy today?
2. Testing the NLP model
3. Testing the conversation
4. Testing custom components & eeh
5. Monitoring and improving your digital assistant

Program agenda

1. **What is your testing strategy today?**
2. Testing the NLP model
3. Testing the conversation
4. Testing custom components & eeh
5. Monitoring and improving your digital assistant

What is your testing strategy today?

Are your end users your testers?

- Developers “playing” with the bot?
- Getting representative users to “play with the bot”?
- Some batch tests?

What is your testing strategy today?

Are your end users your testers?

- Developers “playing” with the bot?
- Getting representative users to “play with the bot”?
- Some batch tests?

Benefits of a solid testing strategy

- NLP is like multi-dimension Jenga
  - Changing one single utterances COULD impact multiple intents
- Consistent and reproducible baseline to compare against
- Allows you to “what if” ideas
- Ability to measure improvements in the digital assistant
  - “Is the bot getting smarter?”

What can, and what should you test today?

Primary testing which can be automated

- The intent resolution
- Conversation flows
- Custom components
- Channel-specific customizations
  - E.g. Web SDK code

Image courtesy of pixabay.com

Program agenda

1. What is your testing strategy today?
2. **Testing the NLP model**
3. Testing the conversation
4. Testing custom components & eeh
5. Monitoring and improving your digital assistant

Testing the NLP model

Use the batch test feature for intents

- Developing testing utterances is part of writing utterances
  - 80/20 split training/testing
- Not aiming for 100% pass rate
- Consider the balance of your testing

Testing the NLP model

Use the batch test feature for intents

- Developing testing utterances is part of writing utterances
  - 80/20 split training/testing
- Not aiming for 100% pass rate
- Consider the balance of your testing

What type of NLP testing should you have?

- In-domain
- Out of domain
- Out of scope testing
- Random
- Neighbour testing
- “Long tail FAQ testing”

Testing the NLP model

- In-domain
  - Regular tests targeting the specific intent
  - E.g., *"Can I order a pizza?"* to test the Order Pizza intent

- Out-of-domain
  - *"The food order came at the wrong time"*
  - *"You threw out the pizza, that is totally out of order"*
    - Should not resolve to the Order Pizza intent, but the unresolved instead

- Out-of-scope testing
  - *"I want to order a new BMW"*

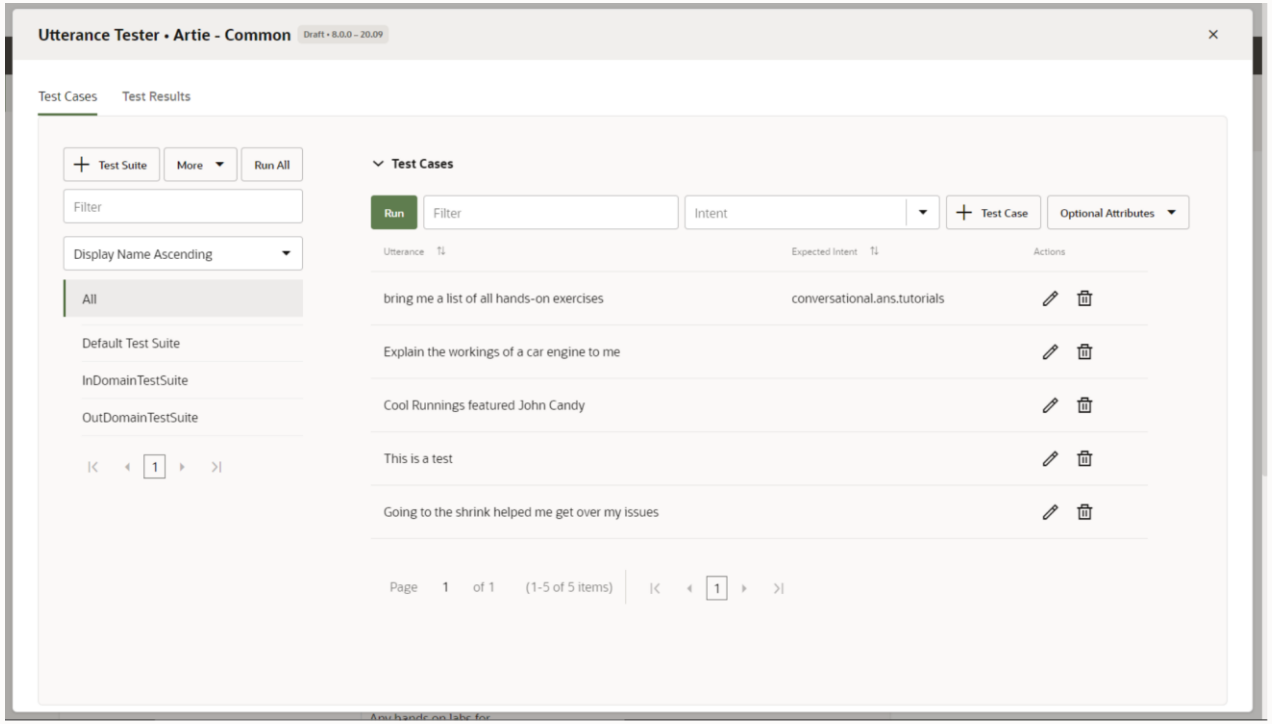
Testing the NLP model

- Neighbour testing
  - Using utterances from other skills, that should go unresolved for the skill being tested
  - This is to make sure there is no overlap

- “Long tail FAQ testing”
  - Make sure that the existing answer intents do not resolve for questions that should go to the Knowledge Base

Testing the NLP model

- Define test suites
  - View and compare runs over time
  - Multi-lingual testing
  - Analytics view
- Digital assistant-level intent tester
  - Test intent resolution at DA
  - You can define the context in which intent should be resolved
- Utterance test cases are part of the skill and are carried along with each version



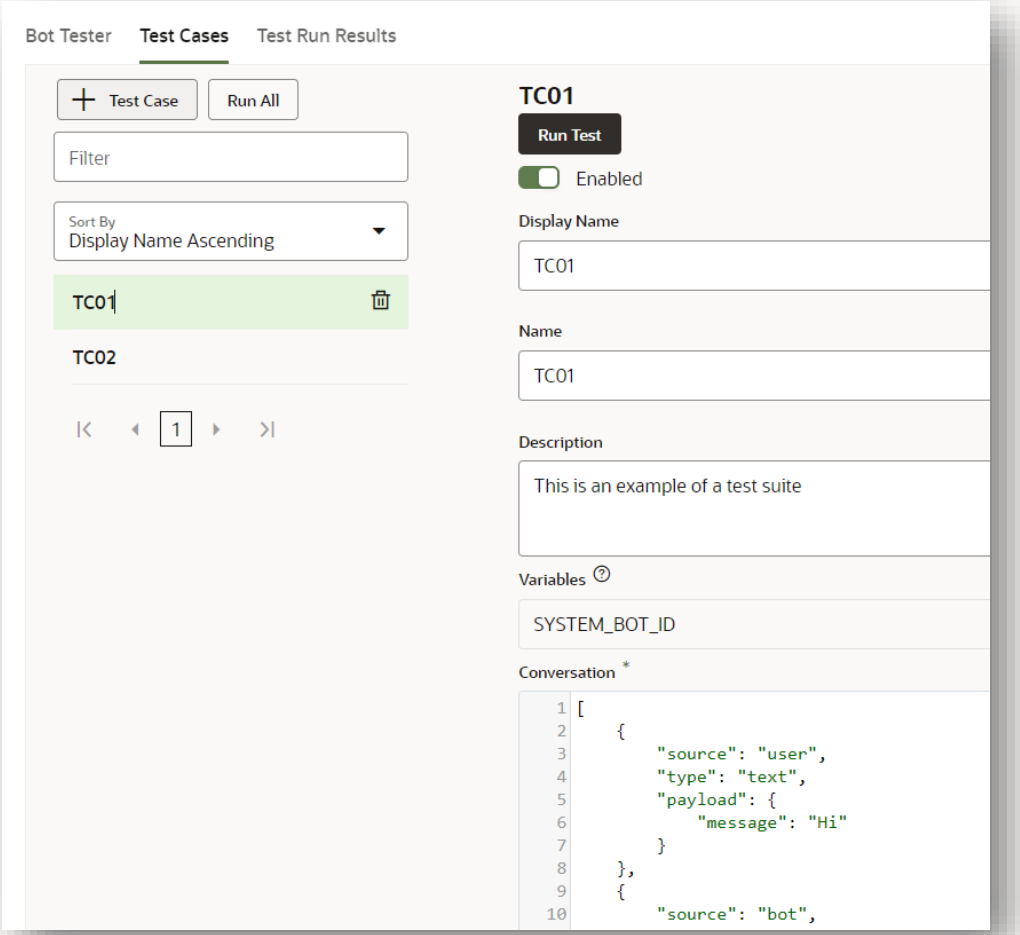
Program agenda

1. What is your testing strategy today?
2. Testing the NLP model
3. **Testing the conversation**
4. Testing custom components & eeh
5. Monitoring and improving your digital assistant

Testing the conversation

Intent conversation is only one element, does the conversation correctly flow?

- Does the bot still give the “right answer”
- Does the entity resolution flow still work the same
- Are the prompts and messages still correct?
- Are you still getting the same disambiguation dialogs?
- Are you correctly handling errors?



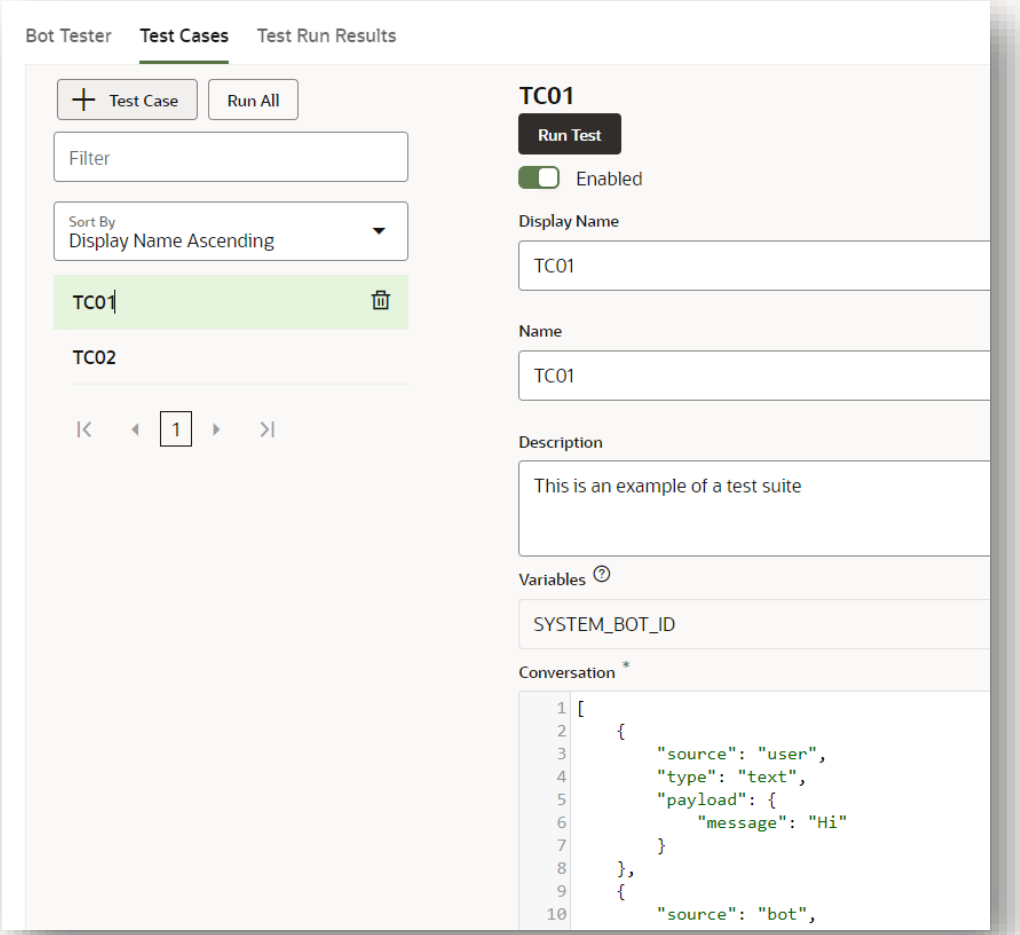
Testing the conversation

Intent conversation is only one element, does the conversation correctly flow?

- Does the bot still give the “right answer”
- Does the entity resolution flow still work the same
- Are the prompts and messages still correct?
- Are you still getting the same disambiguation dialogs?
- Are you correctly handling errors?

Skill conversation tester allows you to record and play back conversations

- Will fail tests if difference in conversation flows
  - Answers, entity gathering, prompts, disambiguation, error handling



> Can anyone see the limitation with this approach?

Testing the conversation

Still useful but

- Keep relatively short
  - There is only pass or fail the whole test
  - Currently limitations on how many tests you can have
- Randomizing prompts or different answers can fail a test
  - You can indicate some strings are ignored

Image courtesy of pixabay.com

Program agenda

1. What is your testing strategy today?
2. Testing the NLP model
3. Testing the conversation
4. **Testing custom components & eeh**
5. Monitoring and improving your digital assistant

Testing custom components

Custom Components & Event Handlers

- Test Driven Development:

  is a software development practice that focuses on creating unit test cases before developing the actual code. It is an iterative approach that combines programming, the creation of unit tests, and refactoring.

- You can use your preferred tool chain of choice (ex. Jasmine, Mocha, Chai…etc.)

- Bots Node SDK provides utility classes to mock:
  - Conversation request
  - Event Handler request

Testing custom components

Create Mock Request

Import Testing Library & Custom Component

Create Mock Conversation from:

- Mock properties
- Mock Variables
- Mock Message Payload
- Mock Channel Type

```
const Tester = require("@oracle/bots-node-sdk/testing");
const HelloWorldComponent = require('../components/hello.world');

it('should respond to a request with params', done => {
    // create a conversation payload iwth properties and variables
    const properties = { name: 'Unit Tester' };
    const variables = { hello: 'Howdy' };
    const request = Tester.MockRequest(null, properties, variables);
    const conv = Tester.MockConversation.fromRequest(request);

    // stub/watch the variable method
    const varSpy = spyOn(conv, 'variable').and.callThrough();
```

Testing custom components

Assert Custom Component Output

Assert Custom Component output:

- Replies
- Transition
- Variables
- Errors
- …

```
  // invoke the component
  HelloWorldComponent.invoke(conv, (err) => {
    expect(err).toBeUndefined();
    expect(conv.getReplies()).toBeDefined();
    // check that the spy was called (once as getter, once as setter)
    expect(varSpy).toHaveBeenCalledTimes(2);

    // make assertions on the responses
    const reply = conv.getReplies()[0];
    expect(Reflect.has(reply.messagePayload, 'text')).toBe(true);
    expect(reply.messagePayload.text).toEqual('Howdy Unit Tester.');
    expect(conv.variable('greeted')).toBe(true);

    done();
  });
});
```

Testing custom component services

Be able to check all custom component services are “up and running” and performing as expected

- Use Postman collections
- Define body/payload
- Assertions

https://learning.postman.com/docs/writing-scripts/test-scripts/

ODADigitalAssistant Development, 5 mins ago

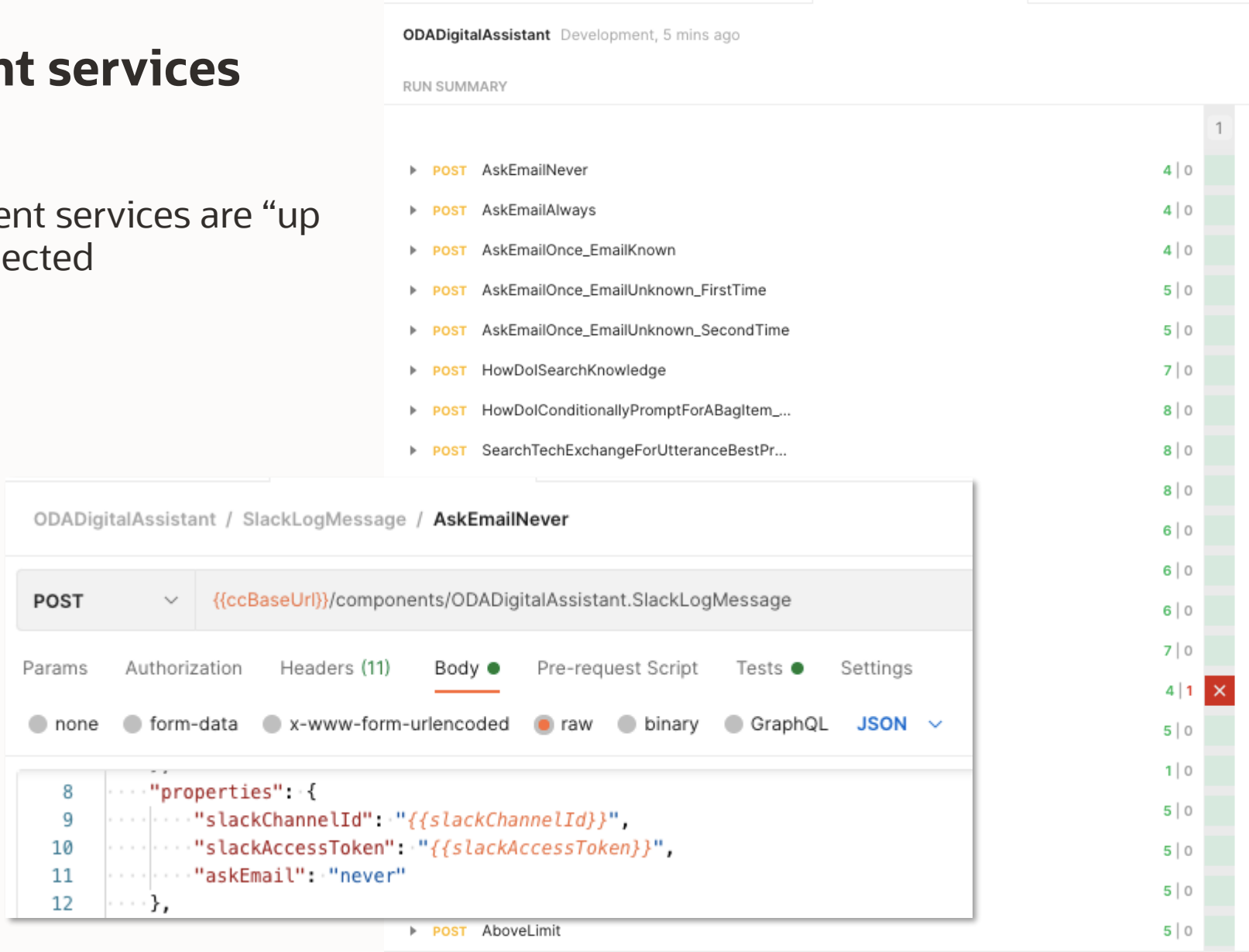
RUN SUMMARY

			1
POST	AskEmailNever	4 | 0	
POST	AskEmailAlways	4 | 0	
POST	AskEmailOnce_EmailKnown	4 | 0	
POST	AskEmailOnce_EmailUnknown_FirstTime	5 | 0	
POST	AskEmailOnce_EmailUnknown_SecondTime	5 | 0	
POST	HowDoISearchKnowledge	7 | 0	
POST	HowDoIConditionallyPromptForABagItem_...	8 | 0	
POST	SearchTechExchangeForUtteranceBestPr...	8 | 0	
POST	HowDoISearchKnowledge_OnlyDocs	8 | 0	
POST	HowDoICreateASkill_SmallLimit	6 | 0	
POST	HowDoICreateASkill_ToBigLimit	6 | 0	
POST	HowDoICreateASkill_DefaultLimit	6 | 0	
POST	Headphones	7 | 0	
POST	100	4 | 1	✕
POST	5000	5 | 0	
POST	ErrorInput	1 | 0	
POST	First	5 | 0	
POST	Second	5 | 0	
POST	Limit	5 | 0	
POST	AboveLimit	5 | 0	

Testing custom component services

Be able to check all custom component services are “up and running” and performing as expected

- Use Postman collections
- Define body/payload
- Assertions



Testing custom component services

Be able to check all custom component services are “up and running” and performing as expected

- Use Postman collections
- Define body/payload
- Assertions

ODADigitalAssistant Development, 5 mins ago

RUN SUMMARY

POST	AskEmailNever	4 | 0
POST	AskEmailAlways	4 | 0
POST	AskEmailOnce_EmailKnown	4 | 0
POST	AskEmailOnce_EmailUnknown_FirstTime	5 | 0
		5 | 0
		7 | 0
		8 | 0
		8 | 0
		8 | 0
		6 | 0
		6 | 0
		6 | 0
		7 | 0
		4 | 1
		5 | 0
		1 | 0
		5 | 0
		5 | 0
		5 | 0
		5 | 0

ODADigitalAssistant / SlackLogMessage / **AskEmailNever**

POST {{ccBaseUrl}}/components/ODADigitalAssistant.SlackLogMessage

Params Authorization Headers (11) Body Pre-request Script Tests Settings

```
pm.test("transition", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.transition).to.equal(true);
});

pm.test("error", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.error).to.equal(false);
});

pm.test("keepTurn", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.keepTurn).to.equal(true);
});

pm.test("action", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.action).to.equal(undefined);
});
```

Program agenda

1. What is your testing strategy today?
2. Testing the NLP model
3. Conversation flow
4. Custom component
5. **Monitoring and improving your digital assistant**

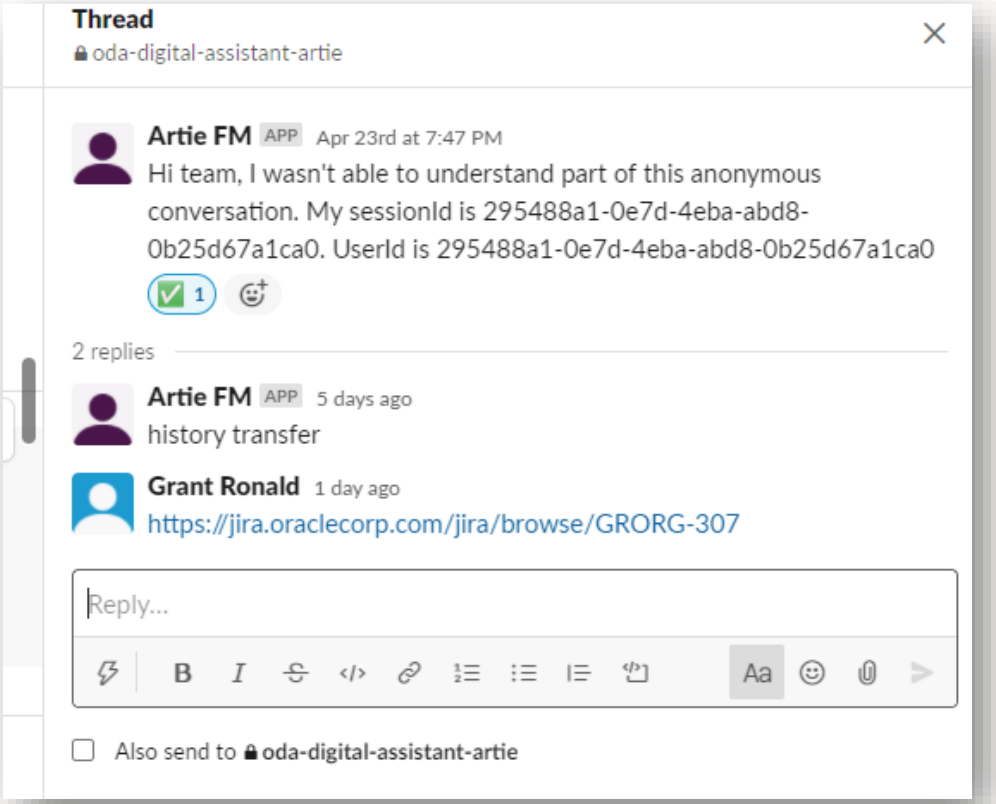
Monitoring and improving your digital assistant

Techniques we have built-in to Artie

- Development team get immediate feedback on Slack for
  - Unresolved requests
  - Where the user has indicated answer is not helpful
- Tracked in development JIRAs

Using ODA Insights

- Most obviously checking unresolved intents for each skill
  - For Artie we have a common handler
- Sample check of conversations correctly resolving



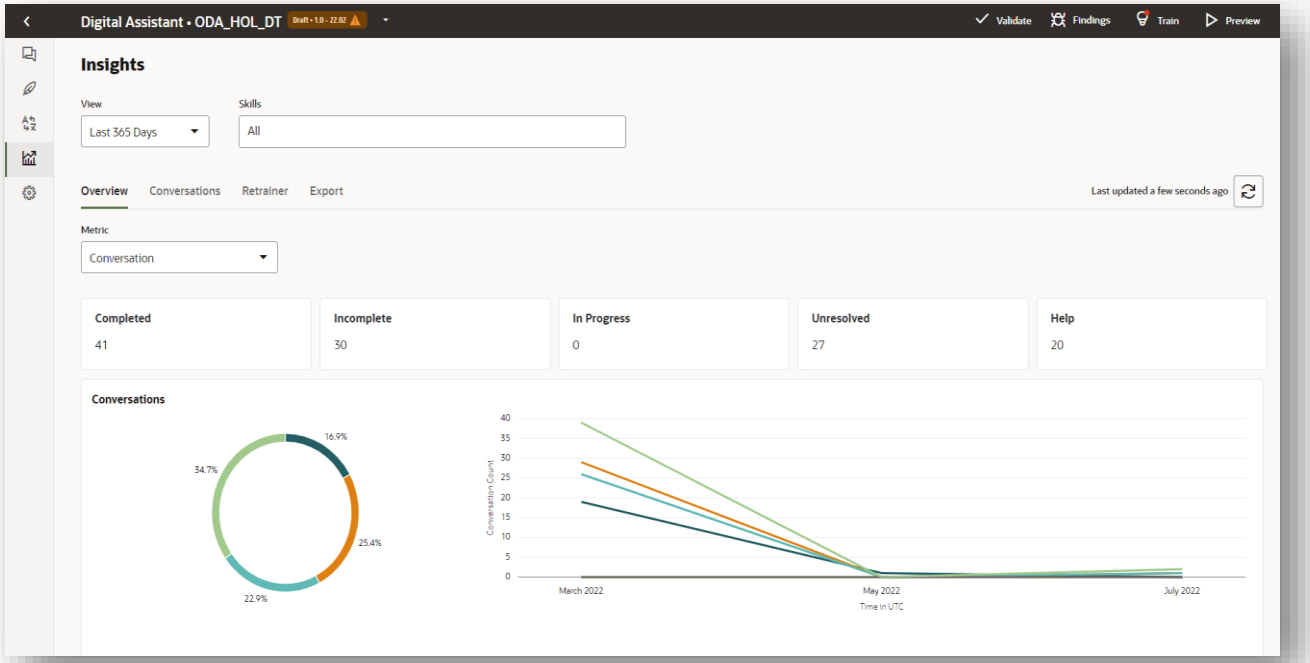
Monitoring and improving your digital assistant

For retraining

- Don’t add just the failing phrase on its own
- Consider variations
- Ensure a balance
- Test to measure the impact

Things to consider for the future

- What measures do we actually want to see from insights?
- Consider use of insights markers



ORACLE