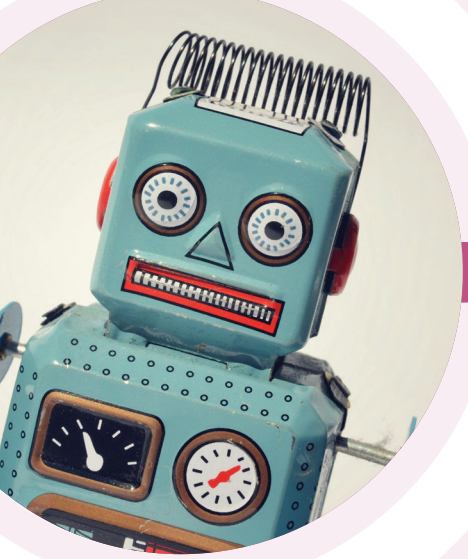


WHITEPAPER

How to test Chatbots

Author: Martin Hart

Introduction

One of the latest hot topics is the use of Chatbots to supplement ecommerce and to offer an alternative to conventional Apps. According to some predictions, Chatbots are the next big thing and could be the end of Apps and search as we know it.

What is a Chatbot?

A Chatbot is a specialist virtual-assistant type program that allows consumers to communicate with it via SMS, messenger applications and/or voice. Chatbots allow the consumer to fulfil specific tasks, such as: ordering pizza, booking tickets, (re-) ordering items, managing new subscriptions etc.

The advantage of using a Chatbot over a conventional App is that the consumer does not need to download the App to fulfil the desired task. Instead, the consumer will use SMS or a messaging application (e.g. WhatsApp, Facebook Messenger, Google Allo, etc.) to send messages to the Chatbot. The desired task will then be fulfilled via a dialogue between the Chatbot and the consumer.

A Chatbot does not necessarily require what is nowadays considered to be AI (artificial intelligence). For example, IBM's Watson system is using neural networks and self-learning to accumulate logic in order to resolve tasks. Watson is currently being trained by IBM to perform cancer diagnosis and learning by processing medical diagnostic data and existing prognosis. This kind of sophistication is not required to order pizza or cinema tickets.

Under the bonnet, Chatbots are workflow automation programmes which identify set workflows from inputs and then execute them.

An example conversation could be:

Consumer on messaging app:	**"What's on at cinema"**
Cinema chatbot (after consulting with the consumer's profile pulled up from the messenger app name).	**"Reading xyz cinema:** **1. [Action film]** **2. [Romantic comedy]** **3. [Foreign film]** **4. [Independant film]** **5. [Classic film]** **Please select a number or film name for screening times."**
Consumer on messaging app:	**1**
Cinema chatbot	**"[Action film] is screening today** **1. 15:00,** **2. 17:00 and** **3. 19:00** **To book, please select the number or another day"**
Consumer on messaging app:	**"Su" or "Sun" or "Sunday"....**
Cinema chatbot	**"[Action film] screening times on Sunday:** **1. 15:30,** **2. 20:00** **Please select the number or another day"**
Consumer on messaging app:	**"2" or "20:00" or "8PM"**
Cinema chatbot	**"Would you like to buy tickets for the [Action film] screening on Sunday at 20:00?"**
Consumer on messaging app:	**"2 adu 2 kids"**
Cinema chatbot	**The admission fee for 2 adults and 2 children is £28.95. Would you like to pay for them now using your** **1. credit card ending 1234** **2. debit card ending 2345?"**
Consumer on messaging app:	**"1" or 1234"**
Cinema chatbot	**"Thank you. Your payment of £28.95 has been successfully charged to your credit card ending 1234. We texted you an admission code and emailed you a bar code. Please present either at admission time and enjoy your film."**

Technical advantages

Technical advantages for Chatbot suppliers:

- No App is required
- No specific GUI required – saves development and testing cost
- The consumer is always using the latest version – no need to migrate an updated App version and wait for the consumer to upgrade.

Now, that we know Chatbots have a lot of potential, how about testing them?

Testing of Chatbots

Basically, Chatbots constitute a "client/server" type model where the consumer is in the role of the client and the Chatbot is in the role of the server.

The Chatbot offers a distinct number of services/methods to communicate with it. Key differences are that:

- Chatbots must be able to interpret unstructured inputs (e.g. "Sunday", "Su", "Sun", "tomorrow", "in 2 days", etc.) all meaning the same thing and
- Chatbots must be able to interpret (to a degree) non-linear behaviour.

Even taking these differences into account, existing testing methodologies and techniques can be applied.

For those readers who have undergone formal ISEB and/or ISTQB training and certification, a Chatbot would qualify as a "input poor but feature rich SUT (system under test)" … for which State Transition Testing is (according to the text books) the best match.

Functional testing

Using common (EP, BVA, etc.) techniques

Given that the Chatbot offers a distinct number of services/methods, these can undergo traditional test analysis using EP (Equivalence Partitioning), BVA (Boundary Value Analysis) or any other appropriate test techniques.

EP would be a good starting point for the above Cinema Chatbot example.

Input partitions would be something along the lines of:

- Consumer driven classes:
  - Service/Method [e.g. film enquiry, book ticket, etc.]
  - Location [default, "Oxford", "West London", etc.]
  - Date and time
  - Film selection
  - Screen selection
  - Payments selection.

Output partitions would be along the lines of:

- List of locations
- List of available dates and times
- List of available films
- List of available screening
- Payment information.

In the background, the Cinema Chatbot would also query and consume information from: the consumer database (WhatsApp name, payment information, default location, etc.), location database, films and screening database, etc.

Depending on the complexity and capabilities of the Chatbots the positive and negative test cases can be created using these partitions.

Free form inputs

The free form input could be seen as a major concern, but in practise "edit packages" would analyse and categorise them.

For example, in my beloved to-do-list applications (e.g. Todoist or Remember the Milk), similar free form inputs occur. For example: if today is Friday, 04 November 2016, tomorrow could (amongst other things) be entered as:

"tomorrow, tom, Saturday, Sat, 5/11, 5/11/16, 5/11/2016, 05/11/16, 05/11/2016, 5 Nov, 5 November, 05 Nov, 05 November 2016, ..."

Even if there are a number of entry options, an edit package would look at the patterns and combinations and translate it into a distinct internal format, such as "2016-11-05" which would then be used for further processing.

Testing such an edit package using standard EP, BVA or classification tree analysis would be pretty straight-forward. Afterwards, the edit package would be a standard off the shelf product and be integrated with minimal additional testing.

State transition testing

The cyclic nature of the Chatbot could be addressed by state transition testing.

As a reminder (I am sure that many readers keenly remember the ISEB practitioner in software testing and ISTQB advanced test analyst courses) about the lingo in state transition testing:

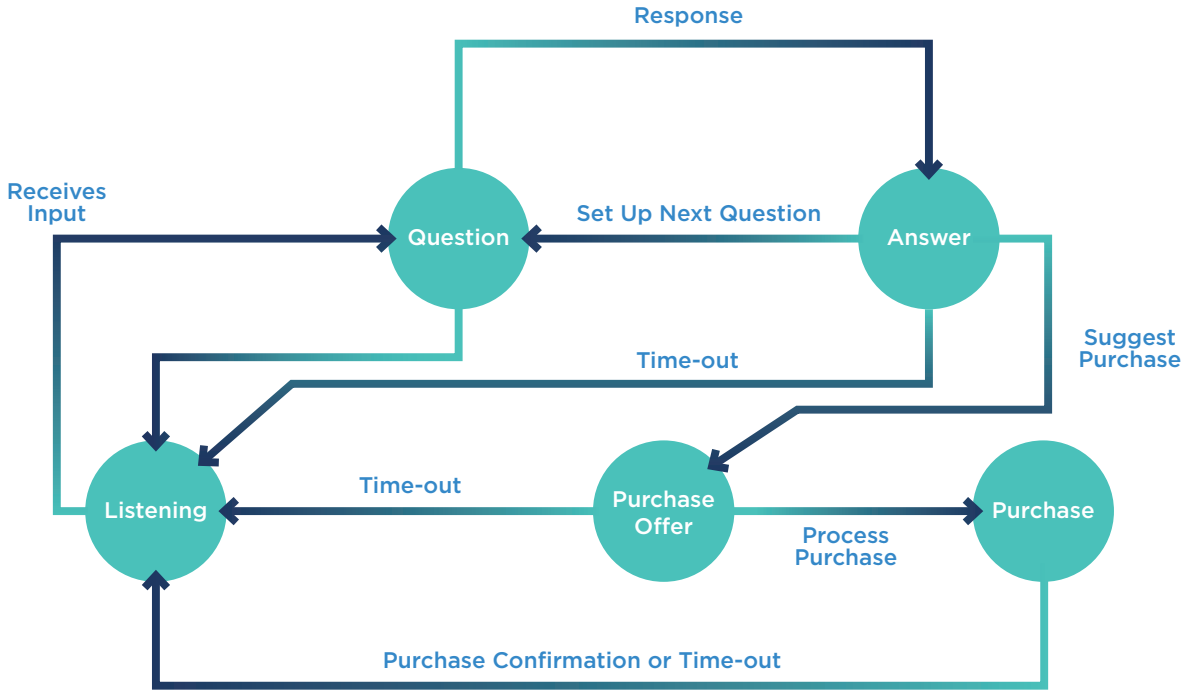
Coverage:

- Traversing: exercising a function for the first time
- Iterating: exercising a function a second time and
- Re-iterating: exercising a function a third or more times.

Switch coverage:

- 0-switch coverage: traverses functionality
- 1-switch coverage: iterates functionality and
- N-switch coverages re-iterates functionality.

A state transition chart would look like this:



We are Testing Solutions Group.
For more information visit: **www.tsgconsulting.co.uk**

The matching state transition table would be:

Input												
State	Question		Answer		Purchase offer		Purchase		Time-out	
	from	to	from	to	from	to	from	to	from	to
Listening	Listening	Question								
Question	Question	Answer								
Answer			Question	Answer	Answer	Purshase offer			Answer	Listening
Purchase offer					Purchase offer	Purchase			Purchase offer	Listening
Purchase							Purchase	Listening	Purchase	Listening

This would result in the following test scenarios:

Path 1					
Switch	Testcase	Start State	Input	Output	End state
0	1	Listening	Question	Response	Answer
0	2	Answer	Time-out	Listening	Listening

Path 2					
Switch	Testcase	Start State	Input	Output	End state
0	1	Listening	Question	Response	Answer
0	2	Answer	Response	Purchase offer	Purchase offer
0	3	Purchase offer	Time-out	Listening	Listening

Path 3					
Switch	Testcase	Start State	Input	Output	End state
0	1	Listening	Question	Response	Answer
0	2	Answer	Response	Purchase offer	Purchase offer
0	3	Purchase offer	Purchase	Purchase details	Purchase
0	3	Purchase	Confirmation	Purchase confirmation	Listening

Path 4					
Switch	Testcase	Start State	Input	Output	End state
0	1	Listening	Question	Response	Answer
0	2	Answer	Question	Response	Answer
1	2	Answer	Time-out	Listening	Listening

Path 5					
Switch	Testcase	Start State	Input	Output	End state
0	1	Listening	Question	Response	Answer
1	2	Answer	Question	Response	Purchase offer
1	3	Purchase offer	Time-out	Listening	Listening

Path 6					
Switch	Testcase	Start State	Input	Output	End state
0	1	Listening	Question	Response	Answer
0	2	Answer	Question	Response	Answer
1	3	Answer	Question	Purchase offer	Purchase offer
1	4	Purchase offer	Purchase	Purchase details	Purchase
1	5	Purchase	Confirmation	Purchase confirmation	Listening

Conclusion

Testing is a universal discipline that uses proven logical methodologies to prove the correct functional behaviour of software.

Even "state of art" ideas like Chatbots can be tested using proven methodologies. Testers shouldn't be concerned about new technology, but should look for ways of applying what they already know to new situations.