CITS5501 week 6 exercise model solutions

Question 1

Currently, if the user makes an error when articulating a command to the device (or the device picks up surrounding noise and interprets it incorrectly as a command), an exception trace is printed to the display and the device must be reset.A colleague of yours working on the voice command interpreter argues that this is acceptable, since it is a precondition of the methods in the interpreter that commands provided fall into a specified list (e.g. "start video", "pause video", "search for video"), potentially with correct "arguments" after the command.Is your colleague correct? Briefly explain why or why not, justifying your answer. (400 words or less)

Sample solution:

My colleague's reasoning is incorrect. From the fact that a method's preconditions are false, it does not necessarily follow that the entire *system* should fail or need to be re-set; and from the fact that an exception trace has been produced, it does not follow that it should be shown to an end-user.

In fact, the contrary is true. For most systems with non-technical end users, it's important that the system be robust and usable. "Robustness" means that the system continues to function even when faults occur in its components. And in the context of usability – non-technical users are unlikely to know what to do with a stack trace, if shown one. Assuming some error *does* need to be shown to the user, it is best to display it in terms the user can understand.

Question 2

You are given a specification for the voice command interpreter component. Testing it thoroughly is important, since users may become annoyed if they give a command and it isn't carried out.You would therefore like to come up with a list of tests, and some measure of how thoroughly they "cover" the functionality of the voice command interpreter.For this purpose, briefly explain how you might model the component (for instance, as a function, a graph, a set of logic expressions, etc.) If there are multiple possible ways, give

what you believe to be the best or most applicable way. Additionally, what measure of coverage would you use? Ensure you justify your answer.(400 words or less)

Mark and feedback

Sample solution:

Multiple answers are possible.

Here, we give an answer that uses syntax-based testing. Good arguments can be made for Input Space Partitioning – the interpreter could be modelled as a "black box" which takes in commands, and produces some behaviour – or for a graph-based model.

Logic- or graph-based models seem less plausible, given the information currently available to us, but there's no reason they couldn't be used as well, in practice; it's merely that on the facts of the scenario, ISP and syntax-based testing seem to fit most naturally.

Given that the component interprets sets of commands which, from the problem description, seem to be largely describeable using a grammar, rather than being in "natural language" like commands to Alexa or similar systems, a syntax-based model seems like a natural choice.

From what we're told, a regular grammar couldn't *completely* define the set of input commands – presumably, there's an infinite (or perhaps just very large) set of names or keywords that users might use to describe videos – but we could still describe and test much of the component's intended behaviour using a grammar.

We're told the interpreter has about a dozen commands that may take several "arguments". Should we perform exhaustive testing ("Derivation Coverage")? We'd need more information to know. For example, suppose each command takes two arguments, each of which could take two possible values – that's $12 \times 2 \times 2 = 48$ tests, which doesn't seem infeasibly large. But the possible values the arguments can take could well be much larger. If they are, then Production Coverage seems a reasonable alternative. The number of tests would be equal to the number of productions, so in the above case, more like $12 + 2 + 2 = 16$ tests.