3815ICT-Software Engineering Workshop 3

Activity 1

The definition of Abstract Data Type in Wikipedia is

" In computer science, an abstract data type (ADT) is a mathematical model for data types, where a data type is defined by its behaviour (semantics) from the point of view of a user of the data, specifically in terms of possible values, possible operations on data of this type, and the behaviour of these operations. This contrasts with data structures, which are concrete representations of data, and are the point of view of an implementer, not a user."

(en.wikipedia.org/wiki/Abstract_data_type) But a somewhat related idea is presented by the video titled "Data Structures: Abstract Data Type (ADT)" www.youtube.com/watch?v=HcxqzYsiJ3k

1. How much does the definition in Wikipedia correspond to the explanation in the video?

2. Where do the concepts of encapsulation and information hiding come into the context of ADT?

Response 1

1. The YouTube video is an explanation of an abstract data type in extremely simple terms using the example of a car to illustrate the concept of an abstract data type in which the user has no need of the underlying structure which corresponds to the input given to the car (Input being steering and speed controls). The Wikipedia definition goes into great depth analysing this concept in mathematical and systematic terms. The two sources come to a similar conclusion being that the purpose of abstract data types relate to the class implementation system of many different programming languages and achieve the goal of providing an interface by which a user can manipulate the given objects.
2. An example of using an abstract data type would be the use of a library of functions and classes. The user will access the library and have access to all of the functionality within that library. They may even have access to the actual form of the library and perhaps even the ability to change things themselves. However, when simply using the library, they simple have access to the functions and the objects without the knowledge of the methods of implementation nor the operations within the functions or classes. This separates the user from the actual structure itself and they simply call on what they need. This is a method of encapsulation and information hiding in that the actual structure is hidden from the user and the utilization of the structure is stored in its own self-sufficient location.

Activity 2

1. Investigate what software engineers mean when they refer to the abstract data type CONTAINER. Hint: a container usually provides and application programming interface to store objects and then recuperate them. Generally, it stores objects of the same class (or type). Construction of a specific CONTAINER usually requires specifying the class of objects it will store. 2.

2. Investigate what software engineers mean when they refer to the abstract data type ITERATOR. Hint: An iterator provides methods to scan over a container or a collection and obtain the elements of a collection one at a time.

Study the two programs ExempleLlista.cpp and ExempleVectorIterador.cpp. These programs are written in C++ and the programmers also didn't write English. However, you should be able to follow the code and discuss the distinction between the standard container in C++ named vector and the one named list to contrasts both implementations of a CONTAINER.

Discuss implementation differences of implementing a Container as a list as opposed to implementing it as a vector.

Response 2

An iterator is utilized to point to specific locations within a container. They may be used to loop through a container or to alter the size and structure of a container.

The difference between implementing and working with a vector as opposed to a list comes down to their relationship with iterators and the ways in which the data can be accessed. Besides this, the methods of referencing various data locations within the array is different. A vector allows random access and manipulation of data which makes it simple to add and remove data from any location within. The trade-off however, is that the references within a vector must change with each successive manipulation. The list provides a durable reference location, where data may be manipulated around a certain index, the index itself is able to maintain its own reference within the given list. The difference however, is that access to the information is more difficult and requires iterators.

Activity 3

Review the slides at

http://www2.kenyon.edu/Depts/Math/Aydin/Teach/Spring03/218/HAND01

Make a summary of the following terms.

1. Precondition.

2. Post-condition

3. Programming by Contract.

Explain how doxygen could be used to show preconditions, and post-conditions, both

- in comments to the programming of methods and functions the define a class API

- in the generated documentation by doxygen for the class so programmers (consumers) of the class understand the class and its behaviour

Response 3

Precondition: A condition which must be met by the client before a supplier is obligated to manufacture a response. The precondition is a responsibility held by the client before sending a request to the supplier.

Post-condition: The post-condition is the obligation of the supplier to the client. If the client provides a request which fulfils the precondition, it is the supplier's responsibility to provide a response agreed upon in the post-condition.

Programming by Contract: A contract refers to an agreement between a client and a supplier. As the word insinuates, certain conditions must be met between both parties before a transaction may occur. This means that the responsibility is shared between the supplier and the client and mutual benefits are granted as a result.

Comments & generated documentation for the programming of methods and functions that define a class API:

Doxygen can provide documentation of preconditions and post-conditions for given methods and functions if it is able to determine these conditions based on the file structures provided. Doxygen is able to provide documentation and this could include diagrams which allude to various Contracts within the file structure. In relation to the documentation provided, Doxygen could provide an additional tab which would define all of the Contracts within an implementation for easy access to the programmer and comments could be added throughout the provided documentation when contracts have been designed by the developer of the implementation.

Activity 4

Use the previous activities and your research skills to provide a definition of Application Programming Interface. In particular, relate the concept not only to a class, but to what software engineers refer to as a framework. Your starting point can be the Wikipedia page en.wikipedia.org/wiki/Software_framework.

Response 4

An application Programming Interface is a means by which various software applications are able to communicate with one another. An API may be developed by large-scale corporations or single developers as a means to provide access to data that they may be storing. This allows outsiders to query that data and use it where necessary. These APIs can be used by developers to develop a framework by which they can leverage access to multiple APIs in order to provide a specific service which they are looking to offer.

Activity 5

An alternative way of describing software behaviour is with axioms that specify functionality. For example, the abstract data type STACK may be defined with a series of axioms like this.

$$\text{is\_empty}(\text{push}(P, x)) = \text{false}$$

This means that pushing a stack with something always results in a non-empty stack. List a series of other properties that define how a stack behaves without given details of a particular implementation. You do not have to use mathematical notation, you can use English. For example, “The constructor returns an empty stack.”

Response 5

- Pushing a stack with something always results in a non-empty stack
- Popping a stack with only one item always results in an empty stack
- Pushing an item onto a stack always adds that item to the top of the stack
- Popping an item always removes an item from the top of a stack.
- An empty stack cannot be popped
- A stack may be of any size as long as the system is able to support it
- Items may only be removed from or added to the top

Activity 6

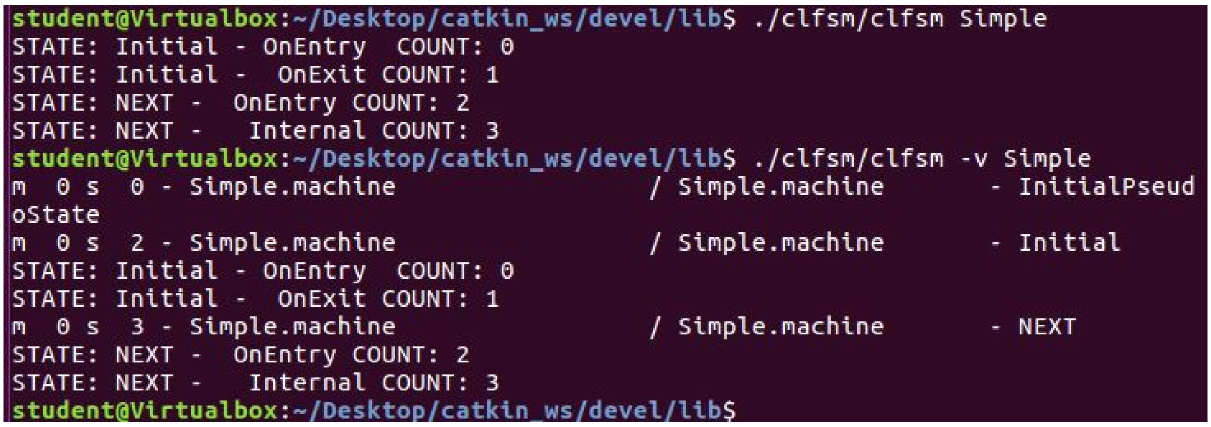
UML’s state diagrams (which is a derivation of Harel’s state-charts [2]) are event-driven. Study the notation in the following web sites:

- en.wikipedia.org/wiki/UML_state_machine
- www.uml-diagrams.org/state-machine-diagrams.html
- https://www.lucidchart.com/pages/uml-state-machine-diagram

However, logic-labelled finite-state machines (LLFSMs) have transitions only labelled by Boolean expression (the only have guards). Drusinsky considers LLFSMS under the name of procedural state machines [1, Page 51]. Drusinsky’s presentation acknowledges that in that case, the model is not at the mercy of the arrival of events: “because [the automaton] can access the input symbols at any time, it can visit states as fast as we wish” [1, Page 15].

Look at the LLFSM presented in Figure 1 (or alternatively, Figure 2 as they should model the same behaviour) and predict what its its output. Do not execute it yet. The exercise consist on you interpreting the model. Once you have written down what you expect the output to be, execute the LLFSM in a ROS (Robotic Operating System) catkin workspace provided for this lab. Copy the actual output of the execution into your report for this workshop and discuss the differences.

Response 6



```
student@Virtualbox:~/Desktop/catkin_ws/devel/lib$ ./clfsm/clfsm Simple
STATE: Initial - OnEntry  COUNT: 0
STATE: Initial -  OnExit COUNT: 1
STATE: NEXT -  OnEntry COUNT: 2
STATE: NEXT -   Internal COUNT: 3
student@Virtualbox:~/Desktop/catkin_ws/devel/lib$ ./clfsm/clfsm -v Simple
m  0 s  0 - Simple.machine                  / Simple.machine       - InitialPseud
oState
m  0 s  2 - Simple.machine                  / Simple.machine       - Initial
STATE: Initial - OnEntry  COUNT: 0
STATE: Initial -  OnExit COUNT: 1
m  0 s  3 - Simple.machine                  / Simple.machine       - NEXT
STATE: NEXT -  OnEntry COUNT: 2
STATE: NEXT -   Internal COUNT: 3
student@Virtualbox:~/Desktop/catkin_ws/devel/lib$
```

My Initial thought would be that the console would print out

STATE: Initial – OnEntry Count: 0

STATE: Initial – Internal Count: 1

STATE: Initial – OnExit Count: 2

STATE: Next – OnEntry Count: 3

STATE: Next – Internal Count: 4

STATE: Next – OnExit Count: 5

>> Back to initial State

At this point I was unsure where the 'Internal' command would come into play. My assumption was that the state would be entered, then run the internal command, then exit. I was unsure if the state would revert back to initial or if there was a means by which the process was restarted. This has been clarified simply by running the machine.

Activity 7

Explain one of the following, and the describe what kind of tools are they.

1. *make*

2. *bmake*

3. *cmake*

4. *catkin*

Response 7

Catkin is a build system or compiler. The role of a compiler is to take raw code which has been written and transforming it into an executable program which can be run by a computer in an environment which corresponds to the language used. Catkin is the official build system of the ROS environment.