Inf2C-SE 2014/2015, Tutorial 3 for week 6

Friday 17th October, 2014

The purpose of this tutorial is to think about the wider picture in which some of the recent content of the course fits. Try to answer the questions posed on your own before the tutorial and discuss your answers, as well as those of the rest of the group, with your tutor. Of course you should additionally attempt to *explain* your answer. Without a justification for a particular answer you are just guessing.

1 Activity Diagrams

1. When and why would you use an activity diagram rather than a sequence diagram?
2. What is the main way in which an activity diagram is differentiated from a flow chart?

2 State Machines

1. States in a state machine diagram may refer to the states of a single object or a collection of objects. Unfortunately, when we have a collection of objects the number of *possible* states can increase dramatically. If you have n components each with m states, then the total number of *possible states* is m^n. More generally each component may have a different number of states, so if each component i has m_i states, and there are n components, then there are $\prod_{i=1}^{i=n} m_i$ total possible states. If we have just 8 components each with just 2 independent states we have 256 possible states. Why is this not normally a major problem for modelling?
2. Suppose then that you are taking part in some project and a co-developer comes to you with a state machine diagram with a number of states far higher than you were expecting for the size of system you are developing. What might this indicate?
3. Draw a state machine for the Cruise Control System as described in the first and second courseworks.

3 Design Patterns

1. Design patterns suggest that you copy an existing solution and modify it correspondingly for your specific application. One thing novice programmers are taught is that *"Copy-and-paste"* programming is to be avoided. Worst still, you may apply the same design pattern more than once in the same project. Isn't this a good argument to avoid the use of design patterns?

2. If you agree that the main advantage to the use of design patterns is as an aid to communication with other developers. Is there any value in using design patterns for an individual project? If you believe that you will only ever write programs that no one else will read (which is a sad thought), is there any benefit to *learning* design patterns?

3. Some programming languages have existing features that subsumes the need for particular design patterns. For example many dynamically typed languages (and some statically typed languages/variants) have a provision for *'MetaClasses'* and this largely obviates the need for the Factory pattern. Do you think that this means that the concept of a design pattern is not a useful one?