NOTES ON 2016 EXAM PAPER

Generally, I was quite pleased with how this exam, the first for the new course, was done. These notes are not model solutions, but they describe some common issues, which may be of interest to people who took the exam and to future students revising.

QUESTION 1

This question was compulsory and, as I explained in the final lecture, a major reason for this was to disincentivise missing out chunks of the material when revising. Most people clearly had revised the whole course. Most of it was straightforward. A few specifics focusing on the less bookworky parts:

(e) Even though I italicised *aim* some answers still explained what LSP is rather than what it aims to achieve. I wanted you to say something like "Clients written in terms of the superclass shouldn't break when they are given objects of a subclass instead.".

(f) Yes: there's no overriding, and a client written in terms of A will never invoke doSomethingElse, so it can't break. (This question is so easy it might almost qualify as a trick question... but it's important to understand what kind of coding patterns are and are not dangerous!)

(g) This was well done; of course, it's very similar to examples you had seen.

```
Winner:
  Individual | Syndicate;
Individual:
  'individual' name=ID;
Syndicate:
  'syndicate' name=ID '{'
    (members+=Individual)*
  '}';
```

(i) The "why" part: If your model transformation is bijective, then the models it relates capture essentially the same information, even if they show it differently. So if you can only express bijective bidirectional transformations, you can't relate different models that capture information about different aspects of the system design - which is the main point of MDD.

QUESTION 2

This was almost everyone's choice of long question. It was generally well done. (a) was bookwork. (b) was deliberately very open; for example, you could come down either for or against using a modelling tool and get the marks, provided your reasons were sensible. I did raise my eyebrows at the people who suggested this company go straight to hard-core

MDD, defining PIMs and PSMs and model transformations between them, given that you were told they had no UML experience; that would really be tough! Lots of people made that very point, though, that they could do that in principle but that it would probably be too big a step to take immediately. A easier angle on the "six different operating systems" issue to argue convincingly was that greater attention to design quality would make it easier for them to isolate and make explicit what was the same vs what was different between these versions, and hence to avoid needless duplication. Some people wasted time by regurgitating a bunch of material from the course but not relating it clearly to the question being asked. (c) was a bit harder than you might think at first sight. Most people correctly said that the Composite design pattern was relevant, but you have to think quite hard about how to apply it in this particular case (what is the parent class, what aggregates what?) I marked it quite generously.

QUESTION 3

This question was very much less popular than Question 2! I don't think it was particularly harder (when it was done, it was done well), but perhaps it was less attractive because it involved material taught quite late. (a)-(c) are bookwork. I think (d) was pretty straightforward for those who had done the labs; the hardest part was probably identifying pros and cons of the different methods. I expected you to say that the textual DSL might fit in better with the company's existing processes than the graphical one, since they were used to working with email, but that on the other hand it might be harder for non-programmers to use. For the non-DSL alternative, I thought the obvious suggestion was that they could just standardise their existing processes a bit more, e.g. introduce templates to use in the mails to make sure nothing got omitted. This would have the advantage of avoiding dependence on any complex tool, but the disadvantage that they wouldn't be able to generate input for the other systems from the templates. However, other answers were possible. The activity diagram at the end is straightforward.