Inf2C-SE tutorial: Requirements (mostly) NOTES

Q2

Work on students' own chosen systems: common mistakes will probably include:

- having no idea what any of these ideas are, because one missed the lectures...
- making use cases too small. A rule of thumb is that a system – any system! – should have between 10 and 100 use cases. This means, of course, that a larger system will have higher-level use cases. It does sometimes happen that one develops use cases at several different levels, but this is rare: beware turning use cases into top-down functional decomposition, which is not what they are about. In fact, wanting to avoid that danger is why we don't talk about "includes" or "extends" relationships between use cases in the course.
- forgetting that the oval on a diagram is not the whole use case, it's just the diagrammatic representation of a use case. (There is typically structured documentation behind the diagram explaining typical scenario, exceptions that can arise, error handling, etc.)
- making user stories too complicated (or, less often, not complicated enough). Remember they have to fit on an index card.

A good user story is something like (example from Wikipedia entry):
Starting Application. The application begins by bringing up the last document the user was working with.

Emphasise the idea that - in fitting with XP's "low ceremony" approach - a user story does not aim to include all detail that's needed to implement the story: rather it is a promise to have a conversation that fills in the details:

"A UserStory is a story, told by the user, specifying how the system is supposed to work, written on a card, and of a complexity permitting estimation of how long it will take to implement. The UserStory promises as much subsequent conversation as necessary to fill in the details of what is wanted. The cards themselves are used as tokens in the planning process after assessment of business value and [possibly] risk. The customer prioritizes the stories and schedules them for implementation. – RonJeffries"

A user story should be:

- Testable – You can write automatic tests to detect the presence of the story.
- Progress – The customers side of the team is willing to accept the story as a sign of progress toward their larger goal.

- Bite-sized – The story should be completable within the iteration.
- Estimable – The technical side of the team must be able to guess how much of the team's time the story will require to get working." (Kent Beck)

  (These quotes are from c2.com, to which students were directed for reading.)

Q3

1. "What a system should *be*" (responsive in 1 second, compliant with standard X, up 99.99% of the time, etc), or equivalently "any requirement that is not one about what the system should *do*", which is the definition of a functional requirement.
2. encapsulation. Some students may confuse it with abstraction, for which the definition is "The process of forgetting information so that things that are different can be treated as if they were the same" (see question below).
3. (a)
4. abstraction, see above.
5. Minimize coupling and maximize coherence, both for maintainability.
6. (e).

Q4

Marking scheme was as follows. Note the acceptability of different correct answers, but also the need for correct use of UML.

(a) Bookwork. 1 each for explanation, 1 for example.

(b) Application of knowledge. Stakeholders are patients, doctors, nurses, politicians, administrators (NB 5 of them, and Q only asks for 4 - some may coalesce doctors and nurses, which is OK given what the question tells them, but unlikely to be right in real life, so we have one group spare). 1 mark for each identification, 1 mark for each explanation.

(c) Bookwork. 4 for "explain", at most 2 if "manage conflicts between requirements of different stakeholders" or similar not included. 1 mark for drawback (only captures requirements that show up as requirements on specific interactions.)

(d) Problem solving. Any reasonable solution acceptable. Marks as follows: 2 marks for actors (doctorornurse, patient, administrator), 2 for use cases (enter consultation report, record changes, generate report), 2 for associations, 1 for details of notation e.g. capitalisation, 1 for comments.