2. Solutions to exercises

2.1 Explain why it is important that software products are developed and delivered quickly. Why is it sometimes sensible to deliver an unfinished product and then issue new versions of that product after delivery?

It is important to develop and deliver software products quickly because of the dynamic and competitive market and because of the need for a company to generate revenue as soon as possible. If you don't deliver quickly, then a competitive product may be introduced first and capture the market. Customers are reluctant to change after they have made a decision. Furthermore, product development is expensive and there is a need to try and cover costs of people, premises etc. as soon as possible otherwise there may be insufficient funding to run the company.

Delivering an unfinished product may make sense because it introduces your product to customers and gives them a chance to embed it in their way of working. They are therefore more likely to commit to a finished version when it has been completed.

2.2 Explain why the fundamental objectives of agile software engineering are consistent with the accelerated development and delivery of software products.

The fundamental objectives of agile software engineering are incremental development and delivery of software and minimising activities such as documentation that do not contribute directly to the delivery of the software. These lead to accelerated development because incremental development means that changes can be accommodated during the development process with only minimal product rework. Rework is a major factor in slowing down delivery. Minimising ancillary activities means that the team can spend its time on delivering software and so can complete it more quickly.

2.3 Give three reasons why Extreme Programming (XP) is envisaged by its developers is not widely used.

1. XP fails to provide guidance on how projects should be managed and organized - it simply suggests that teams should be self-organizing. The reality of software development in businesses is that there needs to be a clear interface between the development team and the wider organization and someone who takes responsibility for everyday management tasks.
2. Pair programming is a fundamental part of XP but it is contrary to the way that many organizations work and is sometimes disliked by developers. There is no definitive evidence that pair programming is more effective than individual developers.
3. While the idea of embedding a customer in the development team is an attractive one, it is impractical in many circumstances to find a representative customer who has time for such involvement.

2.4 You are developing a software product to help manage student admissions in a university. Your agile development team suggests that they should develop this as a number of small

releases that potential customers can try and then provide feedback. Comment on this idea and suggest why it may not be acceptable to the system's users.

This issue exemplifies the problem with incremental delivery. There are some processes where all of the functionality of a system is needed and users prefer to stick with their existing system rather than experiment with an untried new system. They are too busy to 'play around' with unfinished software as they are working on tasks that have a strict deadline. They are therefore unlikely to want to experiment or become involved in the agile development process.

2.5 Explain why the product owner is an essential role in a Scrum development team. How might a development team that is working in an environment where there are no external customers (e.g. a student project team) reproduce this product owner role?

The product owner is an essential role because they are the interface with the potential customers for the software and they have to represent the needs and wishes of these customers to the team. Failure to do this properly will lead to software that does not do what customers want and so is unlikely to be successful.

In a situation where there are no external customers, the problem is that the whole team have ideas about what the software is required to do. While it may seem that appointing a team member as a product owner is an alternative, the reality is that individuals will always question their authority as they are no more qualified than anyone else to say what should and shouldn't be in the product.

Rather than a 'product owner' a possible alternative is a 'product discussion' where the team meet at the start of each iteration and discuss what they will do during that iteration. This should be a short, time-bounded meeting where everyone has the opportunity to make proposals for the work of that iteration. At the end of the meeting, if there is no consensus on what proposals should be accepted, the team should vote and the proposals that get the highest number of votes should be implemented.

2.6 Why is it is important that each sprint should normally produce a 'potentially shippable' product increment? When might the team relax this rule and produce something that is not 'ready to ship'?

It is important to product a potentially shippable product increment for three reasons:

1. This means that there is always a high quality version of the software available as a demonstrator for potential customers and funders.
2. The software should be reliable so can be used as a basis for future development, reused, etc.
3. If people leave the team, new people are not faced with the problem of trying to finish incomplete software.

The circumstances where this rule might be relaxed is when the aim of the sprint is to gather information rather than to create working software. So, if the sprint is developing a throw-away prototype, this does not need to be shippable; alternatively if some new

technology such as a new type of database is being tested, there should be no expectation of producing finished software.

2.7 Explain why estimating the work required to complete a product backlog item using effort in 'person-hours' or 'person-days' can lead to significant variations between the estimated effort and the actual effort to implement that item.

Team members have differing ability and experience. What one team member might complete in 10 hours, might take another 20 hours. Teams will always tend to defer to more senior members when making estimates and, because they are experienced, they may underestimate the time needed by more junior members to finish the work.

2.8 Explain why daily scrums are likely to reduce the time that is normally required for new team members to become productive.

Daily scrums are a mechanism for information exchange where the whole team can discuss issues. New team members attending a scrum are not only provided with updates on progress and problems faced by the team, the meeting also means they can ask questions without feeling that they are interrupting the work of other team members.

2.9 One of the problems with self-organising teams is that more experienced team members tend to dominate discussions and therefore to influence the team's way of working. Suggest ways to reduce this problem.

Two ways in which this problem might be reduced:

1. All team-members propose solutions and issues anonymously on post-it notes and these are all discussed by the team.
2. More junior members of the team are asked to lead discussions on issues rather than always relying on senior team members.

2.10 Scrum has been designed for use by a team of 5-8 people working together to develop a software product. What problems might arise if you try to use Scrum for student team projects where a group work together to develop a program. What parts of Scrum could be used in this situation?

Problems and issues

1. No explicit product manager
   1.2. ScrumMaster is unlikely to be experienced in using Scrum so can't really act as a coach
   1.3. Difficulties in arranging daily Scrums because team members are not full-time but have class commitments.
   1.4. The project goal is not usually to produce shippable software but a working perhaps incomplete demonstrator.
   1.5. Inexperienced team unlikely to be able to make accurate estimates

   Aspects of Scrum that can be used:

   1.6. Product backlog
   ii. Time-boxed sprints producing demonstrable software in each sprint

iii. Regular all-team meetings to discuss problems and agree on backlog items