Practical 1: Eliciting Requirements

Developing the requirements for a software system is a three stage process of elicitation, analysis, and verification. When you elicit requirements it is essential that you understand something of the clients’ domain so that you can understand what they mean by particular statements they might make. When you specify requirements it is essential that you understand something of the software design process so that you can reflect the requirements in a form that is suitable for passing on to other members of the team. This practical is designed to allow you to experience the requirements elicitation process from both sides so that you might gain a better understanding of the need for structured ways of managing requirements.

This workshop will work in four teams:
1. Team 1 will act as the clients
2. Team 2 will act as the software designers
3. Team 3 will act as the requirements elicitation team
4. Team 4 will act as the requirements validation and verification team

In lectures in the following weeks we will cover the details of elicitation, analysis and specification, and validation and verification of requirements, so consider this as a chance for you to explore these areas for yourselves. The clients should explain what they want and the elicitation team should ask them questions to clear up any inconsistencies and to fill in gaps. As the requirements are written down (on the whiteboard or on slides) consider the following questions:
• Clients – do you understand the requirements and are they correct?
• Requirements elicitation team – do you understand the requirements?
• Requirements managers – do you understand the requirements, are they clear, concise and unambiguous?
• Designers – do you understand the requirements and can you start to design the system based on them? Do you need more detail?
• Everyone- how might you structure the requirements (arrange them in groups or under headings) to make the easier to understand?

Client Brief: Our research indicates that the market for home security systems is growing at a rate of 40 percent per year. We would like to enter this market by building a microprocessor-based home security system that would protect against and/or recognize a variety of undesirable “situations” such as illegal entry, fire, flooding and others. The product, tentatively called SafeHome, will use appropriate sensors to detect each situation, can be programmed by the homeowner, and will automatically telephone a monitoring agency when a situation is detected.

References: This class is adapted from 433-341 Software Engineering Process and Practice, University of Melbourne, 2003. The SafeHome example is taken from Pressman, Software Engineering, Chapter