COMP2013

Systems Engineering I

Proof of Concept Design Brief

Group Work (Report and Video) deadline 11.55pm

Wednesday 15th January 2014

Individual Report deadline 11.55pm Monday 20th January 2014

Submission via Moodle and Version Control Repository

Overview

The goal of COMP2013 is to create a Proof of Concept (PoC) design for the system specified by your client. One or more prototypes of the PoC design should be constructed to demonstrate that it is viable and can be made to work. This document will specify what you need to aim for.

Aims

Your work on the PoC should achieve the following:

- Capture a feasible set of requirements for the system, taking care to specify the scope (what is included and what is not) and the expected behaviour and performance. Relevant issues such as security, robustness, reliability, cost, and other properties you identify, should be specified.
- Investigate and model the user interface (UI) for your system. The kind of UI will vary for each project but can be a graphical user interface (GUI), a touchscreen, a web-based interface, a video or speech based interface, or other hardware based interface (buttons, controls, or similar). You will each individually create a UI model during the first half of term but as a group must

decide which UI to support for your PoC.

- Carry out research into the ideas and concepts relevant to your system, using all available resources (e.g., online services, Science Library services, books, research papers, client expertise).

- Investigate and gain experience with the software you can use to build your system. This includes programming languages, development tools, libraries, operating systems, online services and so on. Much of this software will be open source but there is also access to a wide range of Microsoft software including the Azure cloud service.

- Investigate and gain experience with the hardware you need to use for your system. A range of hardware will be made available including Arduino/Engduino devices, Raspberry Pis, tablets (iOS, Android), smartphones (iOS, Android and Windows Phone), Kinect and PC components. During your research you should make requests for hardware as needed *but* while there a good choice available in the department there is no guarantee that specific (or expensive!) items will be available. Plan hardware requirements carefully.

- The construction of one or more prototypes to find out what works or not, and to provide demonstrations of aspects of the system you are developing. It is likely that you will need multiple prototypes, or small test examples, to experiment with software and hardware, and to learn how things work. Remember that a prototype is an experiment and should not be used as a product.

- To define your platform - that is the hardware, operating system, libraries and tools on which your PoC design will run. You should aim for a minimal, clean platform to limit dependencies, complexity and cost. An important part of this will be identifying your target operating system(s) and producing customised or stripped-down version that provides the features needed to support your PoS but omits all unnecessary features (i.e., remove the bloat).

- The development of a substantial source code base and other project artefacts. You must make use of a shared version control service such as Github to hold the all the source code and other files. The repository should be used properly and updated frequently, so that it contains a full trail of the work as it is done, and to provide all the advantages of source control.

- To identify the tool chain you need to implement your PoC design. Tools include IDEs, testing frameworks, libraries and so on.

- To start learning about testing and the use of testing tools. Your production design will need to be thoroughly tested, so start identifying how that can be achieved.

You also need to learn to work as a group member in a team. You already have some good experience from the first year, and you should build on that. Hence, as you work on your project you should:

- Function as a reliable and effective team member, meeting your commitments and making a full contribution.
- Ensure that the group is efficient and organised, that is capable of planning and allocating work effectively, and delivering results on time.

Background

Last year in COMP1010 you had to write programs that made use of a function call API to get robots to perform tasks such as wall following, mapping and navigating a course. That API enabled the C programs you wrote to make use of the hardware features on the robots, by reading sensor measurements, accessing the motor encoders and infrared emitters, and so on. You also made use of a tool chain for cross compiling code, allowing programs to be developed and tested on a PC using the simulator, and then deployed onto the robot hardware and operating system.

The combination of the robot hardware, operating system and the API provided a target *platform* that you could write code for. Using the simulator allowed you to experiment and write proof of concept programs, before having to deploy them on the robot hardware. Both the simulator and the real robot implemented the same API, so both could run the same programs. One thing you found out, though, was that the simulator did not behave in exactly the same way as a real robot, so you had to modify your code after observing what happened on the real robot. Bear this in mind as you develop a Proof of Concept design for your Systems Engineering project.

The robot target platform represented a layer of abstraction over the real robot hardware. Instead of controlling the hardware directly by programming the robot's Arduino board, you made use of an intermediate software layer that hid some of the details but made some things easier to achieve. Systems are typically composed of layers like this, with the hardware at the bottom and then successive layers of software that become progressively more abstract. Ideally the top layer has a level of abstraction that is a good match to the kind of programs you want to write, allowing you to more efficiently write code. Finding the right layers, abstractions and API functions is hard, and best done by building multiple versions to find out what is really needed.

Consumer products such as set top boxes, smart TVs, mp3 players, kiosks, tablets and phone devices are first constructed as proof of concept (PoC) demonstrators by creating a series of prototypes to explore a range of possible solutions. One of the fastest growing industry methods for developing prototype APIs on new hardware is to start with an OS, such as a version of Linux or Android, stripped down to the essential components only and running on minimum spec hardware. Then one or more layers of software are provided to implement the services needed and provide the API that application developers program

against. The aim is to identify a *baseline specification* to determine what features are required to support the device or application, along with the design, costs and trade-offs – good engineering practice!

With a core specification in place, the final combination of required services and capabilities can be selected for the desired product. Also additional hardware choices can be explored based on costs, technology innovation and timeframe for development.

Modern systems engineering processes will prototype and proof the production of a new platform through stages including:

- Careful identification and specification of requirements, with a particular focus on usability. Also market research.
- Thorough research into the problem area to identify potential solutions, relevant new technologies, likely problem areas, feasibility, complexity and a range of other factors.
- Selection and integration of existing components and APIs such as network stacks, graphics devices, human interface device drivers, codecs and so on.
- The development of a series of abstraction layers for higher level software to make use of the hardware, avoiding the need for application developers to access the low level layers or hardware directly.
- Creating a tool chain and processes for compiling and linking components and libraries for a target platform (ARM/X86/MIPS etc.).
- Developing test suites and performance benchmarking.

COMP2013 Timeline

Week(s)	Activities	Individual Courseworks
1	Form group, meet client and start gathering requirements. Also start research.	CW1: Virtual machine configuration and core tool installation.
2-5	Continue working on requirements and research. Do individual courseworks including the prototyping of user interface concepts.	CW2: Setting up a web server and HTML5 tools on a VM. CW3: create UI

		demonstrator.
Reading week	Read! And complete coursework.	
7-11	Develop the PoC design, build one or more prototypes.	
10-11	Continue PoC and start working on the PoC group report and video.	
Christmas Break		
Term 2, Week 1	Deliver PoC design report and video.	
Term 2, Week 2	Deliver individual report.	

COMP2014 will follow directly on from COMP2013, starting from the first week of Term 2 and continuing to the first week of term 3. During that time you will turn your PoC into a product design, give demonstrations and a live presentation, and complete the project.

Group Blog

Each group will maintain a group blog on WordPress (http://wordpress.com). This should record progress, minutes of meetings, decisions made, results achieved and so on. You are should include photos, images and video as well, so you have a comprehensive record of your progress.

Sign up for a free WordPress account, and use your project name as your username (or as close as you can get). The blog URL should be submitted on Moodle. All group members should be able to add entries to the blog.

Team Management

One member of each group will be assigned as the group manager, and be responsible for organising the allocation of work and keeping track of the progress of the project. The group manager will normally remain the same until the end of the project in COMP2014 but unsatisfactory performance will lead to replacement.

All other members of the group are expected to fully cooperate and contribute. They will also take on various roles, including:

- Deputy group manager, to assist the group manager as necessary and ensure the group blog is kept up to date.
- Client Liaison, responsible for interfacing with the client, arranging meetings, sending reports.
- Technical Lead, responsible for leading the PoC design.
- Chief Researcher, responsible for leading the research and recording the results.
- Chief Editor, responsible for leading the creation of the group documentation and video.

Further, each member of the group should have both a primary and a secondary role to ensure that responsibilities are shared fairly (for example, the Technical Lead might have a secondary role as sub-editor).

Groups must attend their assigned lab sessions and can also use the labs at any other time they are available. Minutes should be taken for group meetings, and added to the group blog. It is very important to record decisions, deadlines and who is meant to be doing which tasks, so that there is no confusion (or slacking!) over what is going on.

You must divide the work to maximise the effectiveness of the group. An individual cannot take part in every task or learn everything, so you must work together and communicate effectively so that everyone in the group knows what is going on.

All group members are expected to contribute an equal amount of time and effort. There is no room for passengers.

Individual Blog

As an individual, this module requires you to show clear evidence of your individual contribution to the project, and assessment will take account of this. Use a private, individual, WordPress blog for this. Record what you have done and when, along with your own thoughts and comments. You will need to invite the module lecturers to view your private blog but not your other group members.

Project Deliverables

Key issues to address in the the PoC deliverables include:

- A clear and well-defined set of requirements.
- Results of the research done. What were your sources, what did you find out, what conclusions or decisions did you reach?

- Identification and specification of the platforms or operating systems that are able to support your application. For example, if you use Linux, which distributions provide relevant features such lightweight capabilities, ability to be customised for your project and driver compatibility.

- Exploration of the ways to configure an OS or platform in a virtual machine, in the cloud, or on a real device, to determine how the OS can be customised and streamlined to suit the project you are working on.

- The project version control repository, which all team members must use and *must* use it properly. This means updating very regularly, keeping the repository organised and not checking in broken code. The repository should contain all the source code artefacts and documentation.

  NOTE - you must use a version control repository, this is not optional.

- How the system might be packaged for delivery to end users. For example, potential launch and shutdown scripts. There should be no visibility of the underlying OS to end users.

- Removal of unwanted packages and dependencies from the platform, OS or other software used – the aim is to minimise the software footprint and to provide the optimal selection of components. You should seek out the most economical settings, exploring alternatives and trade-offs. There should be evidence of benchmarks and testing for minimum RAM/Processor/Graphics/Network capabilities to which can deliver appropriate quality of service.

- Determining strategies for how to test your platform, OS and application, to demonstrate that is functional and meets the requirements. Look at unit, functional and acceptance testing tools. Testing should be automated as much as possible (i.e., done by a computer). Manual testing is not acceptable.

- A robust and straightforward user interface, that has good usability and matches the needs of the end user.

Group Main Report (40% of overall module mark)

Submit by 11.55pm 15th January 2014 via upload on Moodle, as a single pdf document, one copy per group. The report should **not** exceed 30 pages (sides of A4) in length. Note this is a maximum length, you don't need to create exactly 30 pages.

Value will be placed on conciseness and clarity. Writing a good report will be a challenge, so allocate enough time for it. You will need to carefully choose what you include and what you omit. Do not dump everything in regardless.

For managing documents like the report you might find cloud services like Dropbox or

Google Drive useful.

The report content should address the issues listed earlier in this document, making sure to include the following:

- Research carried out for your PoC Design.
    - Potential platforms and operating systems. Remember, the platform is the hardware or environment needed to run the OS and the application being developed.
        - What did you look at? Compare the alternatives.
        - What did you choose and why?
        - What are the trade-offs?
    - Do the same for potential programming languages, libraries and other software components suited for the target platform.
        - What choices were there and what are their strengths and weaknesses?
    - Overview of the tool chain needed to build, setup and configure the platform, OS and system.
        - Again, identify and compare the choices.
            - For example, configuration scripts, build process, version control and deployment process.
        - The goal is to automate as much as possible to avoid having to build or configure manually every time something changes. Also to manage all the source code and other artefacts efficiently.
    - Critical discussion on what you have identified as the required performance and operational capabilities for the system.
        - How can they be achieved?
    - Why is your PoC fit for purpose? Provide the evidence.
- The chosen user interface
    - Give an overview of the UI features.
    - Explain why it is suitable and meets the usability requirements.

- Progress made in constructing your PoC Design prototype(s)
    - Describe what you have created so far, showing the design and other features.
        - Use appropriate language, notation (e.g., UML) and terminology to describe the design.
        - Justification of choices and trade-offs.
    - You might cover one or more of the following:
        - How you are configuring your OS, what packages are included or removed.
        - Discussion on the build and integration processes.
        - Scripts or other processes for configuring and launching the OS and application.
        - The layers identified (e.g., hardware, OS, library/framework, application) and how they communicate with each other.
        - The public APIs that your platform and/or OS make available to allow the application to be written.
        - Identification of what has been adapted from the open source community and what is to be newly developed or configured.
- Testing Strategy
    - Investigation of relevant available testing tools and methods.
    - Comparison of alternatives.
    - How automation can be achieved.
    - Results of experiments on the PoC prototype(s).
- Plans for developing the product design during term 2.

Group Video Presentation (15% of overall module mark)

Submit by 11.55pm 15th January 2014 via upload to YouTube, one video per group. The YouTube URL should be submitted via Moodle. The video does not have to be made public on YouTube, as long as it is accessible to the project markers. The maximum length is 15 minutes (default YouTube limit) but aim for around 10 minutes. If possible submit in

720p format, so that details are visible.

The video should:

- Start with an overview of the system being developed and of the key requirements.
- Present the PoC design, showing the key features.
    - Include demonstrations of the system in action.
        - Use screen capture or video of the system in use.
    - Make sure you give a good depiction of what the system might look like and how it should behave.
- Give a summary of achievements to date and plans for developing the product in term 2.

Individual Report (15% of overall module mark)

Submit by 11.55pm 20th January 2014 via upload on Moodle, as a single pdf document. The report should not exceed 8 pages (sides of A4) in length.

The purpose of the individual report is to evaluate the group and the work done so far. The evaluations should be as objective as possible, not just opinions - justify and quantify what you claim.

The report should provide:

- A summary of personal achievements made on the project, and your contributions and decisions made that affected the development of the PoC.
    - Don't just copy and paste your blog content. Treat this part as show case of your contribution to the project. But make sure it is accurate.
- A critical assessment of each group member's contribution, including your own, in their primary and secondary roles.
    - What are each person's strengths and weaknesses (including your own).
    - What roles are they best suited to.
    - Find a way of assigning numbers to a range of attributes (e.g., reliability, technical skills, communication skills, level of contribution), so you can weigh up each person.

    - Don't just give opinions.
- Your personal assessment of the system architecture, quality of the decisions made, whether the system is fit for purpose and how the further development of the system should proceed.
    - Again, fit quantitate ways of assessing these.

Individual Contribution (15% of the overall module mark)

This will be determined from your progress and input during the project, and from your individual blog. A good mark depends on:

- Being a good team player.
    - You need to contribute and give your opinions and input.
    - But also accept the group decisions - the group might not always do what you think is right so don't disrupt progress by being confrontational.
- Delivering your work on time and to good quality.
- Taking part in group meetings and activities.
    - This is a group project, going off and working by yourself is not acceptable.
    - Don't just do the things you like and refuse to do other tasks.
    - If you don't like programming, too bad! Take your share.
    - Mentor each other. If you understand something others don't, then teach them about it.