

COMP204P

Systems Engineering I

Document version 1.0

Deadlines

Project Web Site for clients and Experiment log from Scenario Week 1:

11.55pm Monday 12th December 2016

Individual Report and Project Video:

11.55pm Monday 9th January 2017 (or earlier)

Bi-Weekly team reports and minutes on Moodle:

11.55pm Monday 10th October 2016

11.55pm Monday 24th October 2016

11.55pm Monday 7th November 2016

11.55pm Monday 21st November 2016

11.55pm Monday 5th December 2016

11.55pm Monday 19th December 2016

Overview

COMP204P in term 1 and COMP205P in term 2 are the second year team project modules, where you undertake a substantial project worth 30 credits in total (including your other modules you take 120 credits overall this year). You work on the same project for both modules, starting in October, continuing through term 2, and finishing in late April during the first week of term 3.

You will be allocated to a team including 2~3 students and assigned a project client to work with. The teams are formed taking into account your interests and academic performance in the first year. There is a diverse range of project clients, a number

external to UCL including large and small companies, a number in collaboration with medical doctors and hospitals, charities, and other departments and faculties within UCL. Every client has a real world set of requirements for a system or application that they want you to develop for their use. All projects involve developing software, and some also include hardware.

The projects are organised via the UCL Computer Science Industry Exchange Network (IXN), which provides a complete framework for running projects with a wide range of external collaborators during term time. During the summer considerable effort has been put into contacting and liaising with this year's clients to specify interesting and challenging projects. You will need to work with your client in a professional way to identify their requirements and create the software or system that they need. All projects have the potential to make an impact and deliver real results to the client -- remember that UCL Engineers Change the World!

In COMP103P last year your app development project took you through your first full software development cycle with a client. Your second year project will take you through a second full development cycle, by starting with Envisioning with experimental phases, through to building a Proof of Concept (PoC). Working on a much more demanding project over a considerably longer time you will encounter a large number of software engineering challenges. Throughout this process you will not only develop your technical and development skills, but should also focus on your team working, evidence-based documentation and transferrable skills as you learn to be a professional developer. Take time to monitor your personal development and evaluate how your skills are improving, drawing on your experience in the first year and the skills learnt from the IEP content.

Project Goals

The principle goal of the complete project (COMP204P and COMP205P) is to create fully working Proof of Concept (PoC) design and implementation for the system specified by your client. The PoC should be at least good enough to fully demonstrate how your application or system looks and works, giving the client a clear understanding of what can be achieved. If you are up to the challenge, then you may

be able to turn the PoC into software/hardware of deliverable quality that can be put into real use.

As well as the actual PoC system, you will deliver the design and specification documents, the source code, user documentation, and the results of the research done during the project. These will be gathered together on your project website, which will serve as the principle showcase of your project and its results.

During term 1 on COMP204P you have two major sub-goals. The first is to meet and establish a good relationship with your client, in order to fully identify the project requirements, goals and deliverables. The second is to carry out and document the experimental research into potential solutions, gather the materials needed, and carry out experiments to find what you can actually achieve.

Having a good relationship with your client will be very important. The relationship will continue throughout the project and you will be reporting to them and meeting with them on a regular basis. You will also need to be clear on what you can and cannot achieve, and keep expectations realistic. Don't just say yes to everything, make sure decisions are agreed in writing over email and do clearly explain what you believe is possible. Consider change management and value proposition factors for your client.

During term 2, as the project continues with COMP205P Systems Engineering II, you will take the results of COMP204P to design and build a robust and fully functional PoC. That PoC is intended to provide at least a reference design for your system that is capable of being developed into a final product.

IEP Context

Along with COMP203P Software Engineering and HCI, the two project modules are part of the Integrated Engineering Programme (IEP). In your first year you gained a good experience of team working, a wide range of project skills, and took part in a number of projects. This project will draw heavily on that experience and you should put what you have learnt into full practice.

From COMP103P Object-Oriented Programming (App Development) last year you already have some experience of working with a project client. You are now moving to a much larger and more complex project, with more demanding and high profile clients. In particular, your new project will require:

- Working with your project client over an extended period and keeping the relationship working well.
- Tackling a complex problem, with many potential solutions explored as phases, and identifying the compromises needed to achieve a good working result. Good design knowledge and practice will be important.
- Carrying out detailed research of published work, tools, software and the other elements needed for your project.
- Applying proper implementation and testing strategies, drawing on the computer science knowledge you are building up.
- Properly documenting all your work and results.
- Developing your writing, presentation, management, role effectiveness and communication skills.

It should be emphasised that this is not simply another project. As well as bringing together all the skills listed above, you need to demonstrate that you are a good Computer Scientist. Depth of understanding through the current state of the art and proper application of the Computer Science material will matter and be an important part of the assessment.

Platforms

Last year in COMP105P Robotics Programming 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 written and compiled on a PC, 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 design and write code for. Knowing the API allowed you to reason about the behaviour of programs, while you experimented and wrote proof of concept programs. One thing you found out, though, was that your code didn't always do what you expected when running on 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 microcontroller 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, music players, kiosks, tablets and smartphones are first constructed as proof of concept demonstrators by creating a series of prototypes to explore a range of possible solutions. A very common approach for developing prototype APIs for new hardware or software systems 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, or within a basic virtual machine. 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 and software 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 and especially value propositions. Also market research as part of the investigation and literature review.
- Thorough research into the problem area to identify potential solutions, relevant new technologies, feasibility, complexity and risk, 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 underlying software or 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 (e.g., ARM/X86/MIPS, abstract containers for Unity, plugin environments for Visual Studio/BlueMix etc.).
- Developing test suites, generating effective sample data cases and performance benchmarking for reporting insights on the value propositions.

The COMP204P Envisioning Cycle

By the end of term 1 you should have one, or possibly several, early version experiments or functionality needed for PoCs in place based on the results of your requirements gathering and experiments. You should aim to:

- Capture a feasible set of requirements for the system or application, 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) or user experience (UX) 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, other hardware based interface (buttons, controls, or similar), or a set of APIs.

- Carry out research and document 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 might 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.
- Where applicable, 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 experimental 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. For example, an important part of this might be identifying your target operating system(s) and producing customised or stripped-down version that provides the features needed to support your PoC but omits all unnecessary features (i.e., remove the bloat).
- The management of your source code base and other project artefacts. You must make use of a shared version control service such as the departmental GitLab server 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(s) you need to implement your PoC experiments. Tools include IDEs, testing frameworks, libraries and so on.
- To start learning about testing and the use of testing tools. Your final Proof of Concept design created during COMP205P will need to be thoroughly tested, so start identifying how that can be achieved.

COMP204P Timeline

Week(s)	Activities
1 (3rd Oct)	The initial briefing lecture is on Tuesday 4th October, in the Christopher Ingold XLG2 Auditorium. You need to form your team, meet your client and start gathering requirements. Also start your research.
2-4 (10th Oct)	Build your client relationship, continue working on requirements and research. Existing / example code investigations should begin. Create project website for your client, TAs and supervisor. Present Initial analysis of literature findings to client. Requirements must be agreed and finalised by the end of week 4, with the research identifying strategies, ideas, tools, etc. Compare Software Engineering Methods – choose to explore (many answers are correct!)
5 (31st Oct)	Scenario Week 3 (assessed as part of this module), complete documentation on website. Develop an R&D experiments logging tool as a web app, present to your client how you will conduct and log experiments
Reading week	Review experimental design and propose architecture ideas for how the functionality will integrate (systems integration).
7-10 (14th Oct)	Develop the phases of experiments that will lead you to your PoC design(s). Write up experiments, build one or more prototypes.
11 (12th Dec)	Scenario Week 4 (assessed as part of COMP203P)

As part of the IEP content term 1 includes two scenario weeks, where you will spend the entire week working on one challenge, with no other teaching. Hence, there are 8 weeks in which to focus on the COMP204P project. Note that IEP Scenarios 1 and 2 were the coding curriculum in the first year, the second year has scenarios 3, 4, 5 and 6.

Labs

Your team will be allocated a weekly lab time, which all team members must attend in order to meet with their team TA or supervisor. At the lab session, the team should report on their progress, highlight any issues, and get feedback. The progress of

teams and individuals will be monitored via the labs sessions with the new UCL Greenlights progress system. You will also be audited for your 100 marks distribution.

Team Management

One member of each team is assigned as the team leader, and is responsible for organising the allocation of work and keeping track of the progress of the project. The team leader will normally remain the same until the end of the project in COMP205P but unsatisfactory performance will lead to replacement. A good team leader will always aim to gain consensus within the team and ensure that work is allocated fairly.

The other members of the team are expected to fully cooperate and contribute. During the project team members will also take on various roles, including:

- Deputy team leader, to assist the team leader as necessary.
- Client Liaison, responsible for interfacing with the client, arranging meetings, sending reports.
- Technical Lead, responsible for leading the PoC design work.
- Chief Researcher, responsible for leading the research and recording the results.
- Chief Editor, responsible for leading the creation of the team documentation and video.

Further, each member of the team 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). These roles can change and evolve during the project, so are not fixed.

Teams must attend their assigned lab sessions and can also use the labs at any other time they are available. Minutes should be kept for team meetings. 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 team. An individual cannot take part in every task or learn everything, so you must work together and communicate effectively so that everyone in the team knows what is going on. For

individuals this will require some give and take, you cannot expect to always do the things you like, and should take the opportunity to learn new things or improve in areas where you are weak.

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

Project Deliverables

Content for the Bi-Weekly Reports, Experiment Log and Project Website

Team work, 35% of overall module mark.

Bi-Weekly reports must be submitted on time (see Moodle for the upload links and dates).

The deadline for the website is 11.55pm Monday 12th December 2016.

Bi-Weekly Reports

These reports are to track the progress of the project, what has been done and what each team member has done. See the Bi-Weekly Report tab in Moodle for more information.

Web Site

Each team is allocated a website on a departmental web server. All project documentation, as listed below, must be maintained on that website. Your website is initially empty, so you will need to create it from scratch. It should be properly structured and presentable -- you should be able to show it to your client without any embarrassment.

Aim for conciseness, clarity and readability, making best use of the web medium. The website is not intended to be a traditional written project report, so use of graphics, video and interactive components is encouraged. But don't add material for the sake of it; keep the content relevant and accessible.

The website should include the following:

- An attractive and informative home page, to introduce the project, the team, and highlight specific features.

- A logical structure, with each of the sections listed below contained in their own set of pages, accessed via a menu from the home page and other pages.
- Background and context. What is the system to be designed is meant to do and in what context. Include the initial problem statement, information about the client and the specific challenges to be met. State what is needed for the project to be a success.
- The detailed project requirements and scope that have been identified and agreed with the client. A clear and well-defined set of requirements should be presented, including use cases as appropriate. Include user interface requirements (as relevant to your project).
- Results of the research done. What were your sources, what did you find out, what conclusions or decisions did you reach?
- A description of each relevant prototype built or experiment done. What was discovered, how did it contribute? The use of photos, diagrams and videos is encouraged here.
- The development of the user interface as applicable to your project. Describe the proposed UI features using your HCI knowledge. Justify why it is suitable and why it meets your usability requirements.
- A link to 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. You can use services like VSTS, GitLab or Bitbucket, and your repository must be accessible for marking.

- Initial strategies for how to test your system, to demonstrate that it 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 alone is not acceptable.

- A record of the progress of your project, including meeting minutes and copies of the bi-weekly reports.
- Plans for developing the final PoC design during term 2.

Within your website contents these are the kind of issues you want to address (subject to applicability for your project):

- What are the potential platforms, operating systems, operational environments needed. The platform is the hardware or environment needed to run support the PoC.
 - What did you look at? Compare the alternatives.
 - What did you choose and why? What examples did you find?
 - What are the trade-offs?
- What are the potential programming languages, libraries and other software components identified.
 - What choices were there and what are their strengths and weaknesses?
 - What did you choose and why?
- A description of the tool chain needed to build, setup and configure the platform, OS, settings and application.
 - 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.
- Progress made in constructing your PoC Design prototype(s) by discussing the experiment phases

- Using the experiment log data you build weekly, describe what you have created so far, showing the design and other features. Especially note when things change.
 - Use appropriate language, notation (e.g., UML) and terminology to describe the design.
 - Justification of choices and trade-offs, especially for use of design patterns to refactor, optimise and integrate your solution phases.
- 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.
- Your research, what you investigated, why it is relevant, what did you discover.
- How is the work good Computer Science?
- Testing Strategy
 - Investigation of relevant available testing tools and methods.
 - Comparison of alternatives.
 - How automation can be achieved.
 - Results of experiments on the experimental phases and PoC prototype(s).

The content should be on the website by the 12th December deadline.

Team Video Presentation

Team work, 15% of overall module mark. Deadline 11.55pm Monday 9th January 2017.

Upload to both UCL MediaCentral (<https://www.ucl.ac.uk/mediacentral>) and if given with client permission, to YouTube for their own visibility purposes (make sure you ask them to check the video before uploading and if it should remain public or private). Ensure that the video is linked from your website.

The maximum length is 6 minutes and should be precise in discussion. Plan it well with an introduction to the domain. If possible submit in 720p or 1080p format, so that details are visible.

The video should:

- Start with an introduction slide of your project title, the students involved, the client sponsor and UCL's logo.
- Then an overview of the application or system being developed and of the key requirements.
- Cover the research and experiments performed. Any key citations or prior works should be shown.
- Present the initial experimental phases that are leading the PoC design that has been selected, 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 final PoC in term 2.

Individual Report

Individual work, 15% of overall module mark. Deadline 11.55pm Monday 9th January 2017.

Upload on Moodle, as a single pdf document. You can upload your report at any time from the 12th December if you want to get it done and out the way. The report should be a maximum of 4 pages (sides of A4) in length containing this content:

- A one page summary of your personal achievements made on the project, including your specific contributions and the decisions you made that affected the development of the PoC.
- A one page personal assessment of the experimental phases worked on and the initial PoC design, quality of the decisions made, whether the system is fit for purpose, demonstrates clear value propositions and how the further development of the system should proceed.
- Up to two pages providing a critical assessment of each team member's contribution, including your own, in their primary and secondary roles.
 - What are each person's strengths and weaknesses (including your own). Remember Strengths Finder.
 - What roles are they best suited to.
 - Find a systematic way of reviewing 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.

In addition for each team member, including yourself, give two grades:

* For the quality of their contribution.

* For the quantity of their contribution.

On this scale:

- F: Fail. The work done is unsatisfactory, ranging from no contribution at all to too little to be considered a pass.
- D: Just good enough to pass but weak and inconsistent.
- C: Satisfactory or average. Some contribution has been made but plenty of room to improve.
- B: Good. A reasonable contribution with some weaknesses.
- A: Very good. Good all round and commitments met. Showing leadership and initiative.
- A+: Excellent. Well above expectations, excellent results produced, very good leadership and initiative.

- A++: Exceptional all round.

Most people are probably operating at B level. If you award an A then it really has to be justified, A+ should be considered rare and that the person is really excellent with work that should be highlighted by the client in their communications. A++ should be very rare, and it is unlikely to be used but would typically have achieved work that would lead to a short paper to be published.

Think very carefully about the grades you give. Simply giving everyone as (including yourself) is going to need justification, otherwise your judgement will be called into question. A+ and A++ absolutely need to be justified. Your individual report is seen by the examiners only, so the grades you give are not disclosed to anyone else.

Individual Contribution

Individual work, 20% of the overall module mark

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

- Behaving professionally and making good decisions. * Delivering your work on time and to good quality.
- Being a good team player.
- You need to contribute and give your opinions and input.
- But also accept the team decisions - the team might not always do what you think is right so don't disrupt progress by being confrontational.
- Taking part in team meetings and activities.
- Helping others especially with common findings that would be useful for others to cite.
- This is a team 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.

The remaining 15% of the overall mark is from Scenario Week 3.