Course Proposal Form

Please see Page 2 for instructions on which parts of this form to complete, whom to consult with to avoid unnecessary effort, and where to send the completed form.

Proposer(s): Cristina Adriana Alexandru, Conrad Hughes **Date: 05/12/2019**

Cover page: Basic permanent course information

Unless otherwise noted, items in this section are entered into EUCLID and **cannot** be changed without creating an entirely new course.

Course Name	Software Engineering and Professional Practice
Course Acronym *(used by the School only, e.g., for the Sortable Course List)*	SEPP
Course Level If the course is **only** available to MSc students, then it must be classed as Postgraduate. All other courses, regardless of level, are Undergraduate.	__X__ Undergraduate __ Postgraduate
Normal Year Taken	__ UG1 __X__ UG2 __ UG3 __ UG4 __ UG5 __ MSc
Also available in years *[This can be changed later if need be.]*	__ UG1 __ UG2 __ UG3 __ UG4 __ UG5 __ MSc
SCQF Credit Level Level 8 should normally be used for pre-honours courses. Level 10 should normally be used for optional UG3 courses (so UG4 students may also take them) and for courses aimed mainly at UG4 students. Level 11 should be used for courses aimed mainly at MSc students, whether or not UG4 students can also take them.	__ 7 __X__ 8 __ 9 __ 10 __ 11
SCQF Credit Points	__ 10 __X__ 20 __ 40 __ 60 __ 80 __ Other:
Delivery Location	__X__ Campus __ On-line Distance Learning
Course Type	__X__ Standard (default) __ Dissertation __ Online Distance Learning __ Other (specify: Placement, Student Led Individually Created Course, Year Abroad)
Marking Scheme By default, courses use a numerical marking scheme. If you wish to use a grade-only marking scheme, your course proposal below should justify this.	__X__ Standard (numerical) __ Letter grade only

Guidance for remaining sections:

For an initial course proposal, please complete the **cover page and Section 1 (Case for Support)**, which asks you to describe the need for this course and to provide an overview of the course design, including the learning outcomes. **Please discuss your plans as early as possible with the head of Curriculum Review to avoid unnecessary effort.**

Send the form with these sections completed to the BoS Academic Secretary and head of Curriculum Review (listed on the BoS page) to obtain their comments before filling out the remainder of the form.

If a full proposal is invited, please complete the remaining sections and send to **iss-bos@inf.ed.ac.uk**.

2. Student-facing course description and additional feedback and assessment information.
This section provides most of the information students see in the DRPS entry for this course, as well as related details for BoS consideration.

3. Further information for BoS consideration: sample materials.

4. Additional Course Details required for DRPS. *[Administrative information such as delivery timing and prerequisites.]*

5. Placement in degree programme tables. *[Required for all level 9-11 courses; used to determine where the course will be added to existing degree programme tables.]*

6. Comments from colleagues. *[All course proposal should be sent to relevant colleagues in the area as well as to the appropriate year organizer and BoS Academic Secretary for comment in good time before the BoS meeting. Use this section to indicate what feedback has been solicited and received.]*

Colour coding and item-by-item guidance:

Guidance is provided in italics for each item. Please also refer to the guidance for new course proposals at http://www.inf.ed.ac.uk/student-services/committees/board-of-studies/course-proposal-guidelines. Examples of previous course proposal submissions are available on the past meetings page http://web.inf.ed.ac.uk/infweb/admin/committees/bos/meetings-directory but note that the proposal form was updated in Jan 2019.

Sections in gold are for student view and are required before a course can be entered into DRPS. You must complete these sections even if your course has already been approved based on other documentation.
Sections in orange are for School use but are still required for all courses (even those that have already been approved based on other documentation).
Section in gray are for consideration by the Board of Studies. They are normally required for all new course proposals but may be omitted in some circumstances (e.g., for invited course proposals) if you obtain permission in advance.

1. Case for support

This section is for consideration by the Board of Studies. The final two boxes (Learning Outcomes, Graduate Attributes) will also go into the student-facing course description.

Overall contribution to teaching portfolio and relation to existing curriculum

Please explain (a) what motivates the course proposal (e.g. a previous course having become outdated/inappropriate, an emergent or maturing research area or new research activity in the School, offerings of our competitors) and (b) how it relates to existing courses and degree programmes (including any prerequisite courses). Every new course should make an important contribution to the delivery of our Degree Programmes.

Software Engineering and Professional Practice (SEPP) is a new 20-credit compulsory second year undergraduate course proposed for 2020-2021 semester 2 for Software Engineering, Data Science and some Computer Science degrees. It is intended to replace the current 10-credit Informatics 2C- Introduction to Software Engineering (Inf2C-SE) second year semester 1 course, one of the most important motivations being including an earlier focus (both theoretically and through practical engagement) on professional issues in the curriculum, as required by most recent curriculum updates. Building on the experience and feedback results for the current Inf2C-SE course, but also on the requirements of the Software Engineering job marker, other important motivations for SEPP include the need for:

1. More up-to-date considerations of contemporary iterative development and deployment lifecycles
2. Confronting students more with the context of developing large software systems (apart from professional issues mentioned above, also e.g. issues of cost, tight deadlines, respecting non-functional requirements, unpredictability) and its implications on software development decisions
3. More emphasis on developing team working skills, but also the management of team work through e.g. version control
4. More practical experience with all of the above through engagement with a larger, realistic, case study
5. A consideration of pedagogical approaches which can improve understanding and foster deep learning: formative assessment by instructors and through peer review, self-assessment, reflection on feedback, reflective writing.

The contents of the SEPP course need to follow on from the first year Inf1B-Object Oriented Programming course, and moreover they must lay the foundations for the later Software Design and Modelling (SDM), Software Testing (ST) and Professional Issues (PI) courses. For these reasons, in preparing this proposal, we consulted with former and current lecturers in our school's Software Engineering courses (Paul Jackson, Perdita Stevens, Nigel Goddard, Ajitha Rajan), the lecturers for the related Inf1B (Paul Anderson, Volker Seeker), SDM (Perdita Stevens), ST (Ajitha Rajan), Systems Design Project (SDP) (Barbara Webb) and PI (Stuart Anderson) courses, a specialist in pedagogy (Judy Robertson), as well as with the current Director of Teaching (Stuart Anderson) and Deputy Directors of Teaching (Sharon Goldwater, Paul Patras), administrative (Gillian Bell), learning technology (Alex Burford) and library staff (Angela Nicholson) members. This was achieved through both informal meetings and an Edinburgh Learning Design Roadmap (ELDeR) 2-day workshop. The plan outlined in this proposal is the accumulated result of these endeavours. In particular, through discussion with the current Inf1B course organiser, it ensures that SEPP will offer a natural progression from Inf1B by allowing students to practice their programming and test their understanding of good object oriented design. SEPP will introduce students more to testing and debugging which they will need for ST. It will motivate the need for UML models which they will learn more about in SDM, and touch on professional issues which will feed nicely into the PI course. Moreover, as discussed with the SDP course organiser, it will prepare students better at working in teams and managing a whole contemporary software development lifecycle and the challenges that it brings for the building of a larger scale software system as part of a more complex coursework.

Target audience and expected demand

Describe the type of student the course would appeal to in terms of background, level of ability, and interests, and the expected class size for the course based on anticipated demand. A good justification would include some evidence, e.g. by referring to projects in an area, class sizes in similar courses, employer demand for the skills taught in the course, etc

This course is aimed at second year students of Informatics programs who have passed the Inf1A- Introduction to Computation and the Inf1B-Object Oriented Programming courses. At the end of Inf1B, students will have familiarised themselves with object oriented concepts and Java programming. They will have been informally shown the motivation for an engineering approach to software development, and they will have a basic understanding of good engineering practice including version control, testing and documentation. SEPP will build on these initial foundations to shift focus on the engineering of larger software systems and the professional and other contextual issues involved. Implementation activities will also be carried out in Java, so no further programming background will be required from students. To help students revise their Java programming, the lecturers will provide links to resources and online tutorials (e.g. from LinkedIn Learning) on the Learn course page ahead of the first week of the semester. Moreover, the first 5 weeks of the course will involve their interaction with an existing code base, which will allow them to strengthen their programming skills.

Inf2C-SE currently has 236 students enrolled, however the demand for first year courses has considerably increased- to 400 students- this year. We therefore expect an increase in demand to over 300 students for SEPP in 2020-2021. For the rest of this proposal, we will consider this number to be at 320.

Anticipated Resource Requirements

Estimate how much lecturing, tutoring, exam preparation and marking effort will be needed in steady state, and any additional resources needed to set the course up initially. Provide estimates relative to class size where applicable and discuss how support staff will be recruited and supervised, if the class is likely to be very large. Please mention any scaling limits due to equipment or space. If equipment is required, say how it will be procured and maintained.]

We do not expect the resource requirements for the SEPP (20-credit) course to be more than double those for the current Inf2C-SE 10-credit course. We propose the following:

- 3 lectures a week instead of 2 which is currently the number for Inf2C-SE; this will facilitate the integration of professional issues, as well as inviting guest lectures from industry.
- No tutorials, so there will be no need for tutors (as opposed to Inf2C-SE which involved 21 tutor posts in 2019-2020)
- For each student, one drop-in 2-hour lab each week, held in a small tutorial-style room with support from one experienced mentor assigned to his/her assignment group (see 'Narrative description of the course aims and structure below'), starting in week 1. For 320 students, we could go for 20 2-hour lab groups of 16 students (4 groups of 4) a week.
- For each student, one 15-minute 'customer meeting' each week with a different mentor (to their own from the lab) acting as a customer who will provide/clarify requirements and evaluate his/her assignment group's work (see 'Narrative description of the course aims and structure below'). Each mentor will see 4 groups for 15 minutes each in a customer meeting, and so there is a need for 20 1-hour customer meeting groups every week..
  We propose 20 'mentor' demonstrator-type posts at 33 hours/post (for the 11 2-hour labs covering the 11 weeks of term, plus the 11 1-hour customer meetings). The mentors will be recruited from teaching support staff with experience in Software Engineering and Java, prioritising former tutors and demonstrators on previous iterations of Inf2C-SE who have had a positive impact on the course, and other applicants with a good amount of commercial experience as Software Engineers. They will be trained by the course organiser. The course lecturers will also drop into labs at different times in the semester to support the mentors.
- For each student, one recommended self-study 2-hour drop-in lab a week (provided to encourage meeting to work on assignment, no demonstrator support provided). We will aim to recommend typically less busy lab spaces and times which are after hours, and avoid booking these labs as much as possible.
- We propose for this course to be coursework only (no exam), so no exam preparation time is needed
- One or two TAs at minimum 120 hours of work (60 hours each) will be required for preparing the assignment instructions, marking criteria, tests or auto marker.
- The marking is estimated to require around 280 hours (1 hour coursework 1 and coursework 3, 1.5 hours for coursework 2), split into 7 marker posts at 40 hours each.

Quotas, special arrangements or unusual characteristics

Please specify if this course requires any special arrangements such as quotas or other registration arrangements; is a collaboration with another school or institution, or has other atypical characteristics that may affect finances or student registration. Further justification/information may be requested for such courses.

No.

Narrative description of the course aims and structure

Please describe the main goals of the course and how the course design will allow students to achieve those goals. This section should be consistent with the student-facing information provided below, but should provide additional information to help colleagues at BoS understand the vision and structure of the course. This description may refer to the learning outcomes and graduate attributes (next two boxes) and should explain how activities such as tutorials, labs, or in-lecture activities will support them, and how the proposed assessments will assess them.

For courses that are important pre-requisites for other courses, this section may also provide content/syllabus information which is too detailed for the student-facing description, such as a lecture-by-lecture syllabus.

Summary statement (from ELDeR workshop): This course will introduce the foundations of contemporary iterative software development and deployment lifecycles, emphasising hands-on experience, real-world large-scale systems, and professional practice.

To achieve these aims, we plan for assessment to be 100% coursework, to allow for more hours to be spent by students on hands-on experience with a larger-scale system. We do not believe that understanding of the practical, professional and ethical challenges of Software Engineering can be meaningfully assessed through examinations. Lecture content will complement the coursework, as well as provide a foundation of the fields of both Software Engineering and professional practice in parallel, as shown in Fig. 1. It will be backed up by reading, which students will be expected to focus on to deepen their understanding of the concepts taught. Moreover, it will be supported by examples from the industry through guest lectures closely related to the 'live' topics at that moment in the coursework. In regards to Software Engineering, lectures and associated reading will both introduce the approaches, tools and techniques that students are required to use, as well as those that they could consider and choose to use (thus building up a 'toolkit'), as part of their coursework. Throughout their coursework, students will be required to reflect on what they have used (from the toolkit), why, how they worked for them, and how they would envisage them working if the software was increased in complexity. Lectures and associated reading will touch on Professional Issues throughout, and students will be required to reflect on the professional implications of their work throughout the coursework. We will make frequent reference to the ACM Code of Ethics and the BCS Code of Conduct (for example, the lectures on testing and debugging will be delivered in relation to ACM Code of Ethics 2.2- High standards, 2.5- Evaluation of computer systems, 2.9- Robustness). As the coursework progresses, the lectures will shift to bigger picture issues, sometimes in parallel with those being raised in the coursework's practice. The closing lectures will place the students' experience in context of Software Engineering in much larger or much more formal environments.

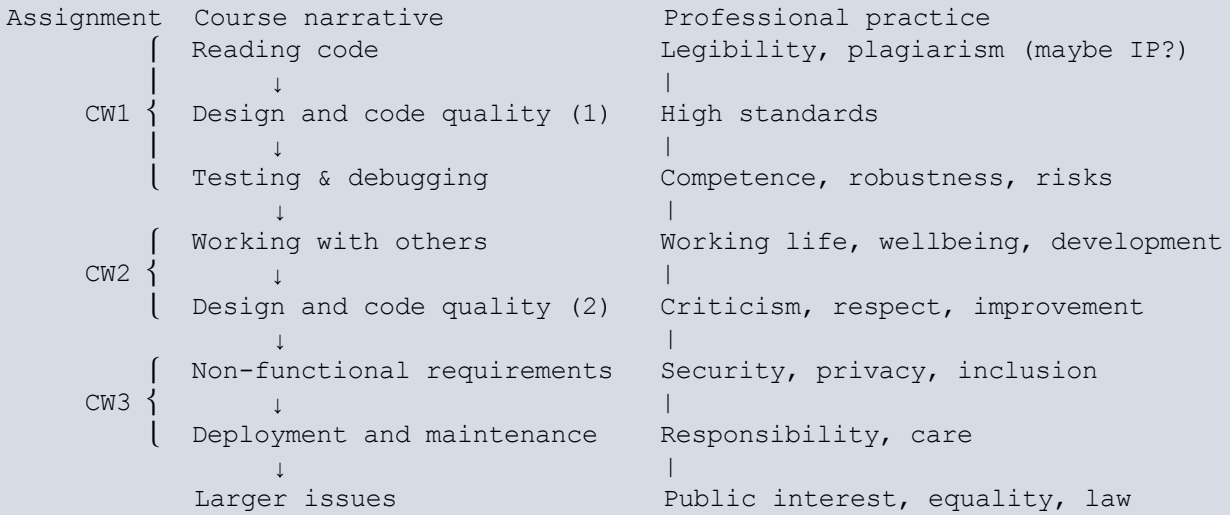

Figure 1- Overview of the course content

The coursework will consist of a realistic case study that students will be working on in groups of 4. We plan to seek industry support for the topic of this case study. **Its first part** will give students the opportunity to interact with an existing code base, interpret it, improve its design and code quality, debug and test it, as supported by the lectures and reading. In parallel, they will be asked to consider professional issues related to this task, such as thoroughly evaluating computer systems and their impact,s and ensuring high standards. The lectures and reading will back up this part by touching on change control, build tools, UML class diagrams (that they could consider in parallel with the code to help them improve its design), good design and good code, refactoring, testing and debugging. In terms of professional issues, topics around professional responsibilities will be touched on. This first part will be assessed formatively. **The second part of the coursework** will involve extending the code with additional/changed functionality by working alongside a technical standard that the code partially fulfills, while also considering professional issues surrounding working with others. Each student will be working on a different independent software module, so that problems can be minimised if he/she drops out of the course. It is at this point that agile software development processes will be introduced to the students in their lectures and reading, and they could decide to make use of agile practices in their work. Moreover, the lectures and associated material will return to design by discussing design patterns. **The third part of the coursework** will require students to see their system as 'in use' and provide maintenance to it by considering some changed non-functional requirements. Moreover, a larger scale ethical problem will be presented to them. In the lectures and associated material, we will be touching on non-functional requirements and their associated professional issues, as well as on deployment and maintenance. Coursework 3 will also be summative.

Throughout the coursework, students will be required to write reflectively- often on a weekly basis- on a variety of issues including: their choice of, and experience with, Software Engineering approaches, tools and techniques; professional issues; their group's interaction, thus developing their introspection. Other opportunities for introspection will be given by peer reviewing their classmates' solution on another module to their, comparing their code with that of other classmates for the same module, and through self-assessment for the first two parts of the coursework, the latter also allowing for the adjustment of expectations regarding assessment once marker comments are received.

Groups will be assigned mentors who will stick with them and offer them feedback throughout the course. Formative feedback will also be offered by peers during two rounds of peer review- once before each of deadlines 2 and 3. Finally, groups will be assigned to a different mentor to their own playing the role of 'customer', whom they could get feedback from on the interpretation of the requirements and the evaluation of their solution.

The assessment model proposed maps with the Learning Outcomes from the next section as follows:

1. For their coursework, students will be encouraged to choose between a range of modern techniques used in the design and development of large-scale software systems - introduced in lectures and covered in detail in associated reading - and then reflect on their experience with applying them for their small-scale case study. This will involve explaining these techniques, as well as answering 'what-if' questions about their applicability for more complex systems than the one they are building as part of their coursework. The quality of their reflection on the techniques used will constitute part of their mark for coursework parts 2 and 3.
2. All the parts of their coursework involve the application and evaluation of the modern techniques used in the design and development as part of a realistic case study. As part of their reflection, students will need to explain how the chosen techniques have worked for them (i.e. evaluation). How well they have applied the techniques will be apparent both from the produced software solution, as well as from their reflective accounts. Their evaluation will be assessed from their reflective accounts.
3. Students will need to learn how to work effectively as part of a team of 4 throughout their coursework. In this process, they will be required to use version control and build tools, and they will also be encouraged to use techniques for working in teams (e.g. pair programming). All of these will be covered

by lectures and associated reading. Students will be required to reflect weekly on team work and the progress of their team, for each part of the coursework. Students will be able to evidence this outcome through their reflective accounts, but also participation in terms of git commits and as supported by peers and their mentor.

4. Students will be asked to reflect on the professional and ethical implications of their work for each coursework submission, by being provided with questions that are targeted to each coursework part. The material on professional issues presented in the lecture, as well as associated reading, will offer them the background knowledge to be able to tackle this task. Moreover, in coursework 3, they will additionally be required to come up with solutions to a larger ethical problem related to the use of their developed software. The quality of their reflection on these topics will constitute an important percentage of their marks on this course.
5. Reading technical documents will be an important part of the overall reading for this course. Apart from those on different tools that students could use, students will need to interact with a technical standard as part of their second coursework. Moreover, they will be required to write documentation presenting their solution. Their appropriate use of any tools, interpretation of the standards document, and documentation produced will all be assessed as part of their coursework.

The marks for coursework parts 2 and 3 will be summed up from a group and individual mark for each student. Moreover, individual marks will be given for the individual's participation through peer reviews, commits to the team's code and according to the opinion of both students and mentors on the individual's levels of engagement and contribution (this may be used to moderate individuals' share of the group mark, as tried and tested in SDP).

To facilitate marking as well as self-assessment, a rubric-based marking scheme would be used by the markers. Moreover, we plan to provide this marking scheme for the students' self-assessment for the first two coursework deadlines.

A preliminary plan for the lecture topics, intertwined with guest lectures, coursework releases and deadlines, is provided below:

- Week 1
    - Getting started; Overview of the field and activities involved
    - Introduction to the coursework (potentially from industry representative); Change control, importing software, building it and getting it running.
    - Models; Producing a class diagram; Relationship code-diagram-
- Week 2
    - Principles of good design
    - Good code, incl. documentation/Javadoc; Code smells.
    - Refactoring
- Week 3
    - Testing; Unit testing, integration and system testing; Coverage
    - Test first and test driven development
    - Guest lecture 1: Testing
- Week 4
    - Bugs and bug reporting; GitLab support for same; Jira.
    - Debugging and bug fixing
    - Technical standards: APIs, interoperability.
- Week 5
    - Agile: principles, XP and Scrum; Comparison with plan-driven processes

- Working with others — professional relationships
- Guest lecture 2: Agile processes
- **Deadline coursework 1**
- **Coursework 2 launched**

- Week 6
  - Group feedback on coursework 1; Code review, peer review
  - Design Patterns (2 lectures)
- Week 7
  - Non functional requirements and metrics for them
  - Privacy & security.
  - Guest lecture 2: Non functional requirements
  - **Deadline peer review 1**
- Week 8
  - Usability
  - The digital divide.
  - Ethics, GDPR, discrimination.
  - **Deadline coursework 2**
  - **Coursework 3 launched**
- Week 9
  - Deployment and maintenance.
  - Guest lecture 3: Ethics
  - Cloud, services; Serverless
- Week 10- starting here optional content
  - Group feedback on coursework 2
  - Software engineering activities. Higher level concerns, risk management and improving process quality including planning. (2 lectures)
  - Lecture/guest lecture 4**:** Democracy & The Internet
  - **Deadline peer review 2**
- Week 11
  - IP and licensing
  - Potentially student demonstrations to representatives from the industry
  - **Deadline coursework 3**

Summary of Intended Learning Outcomes (MAXIMUM OF 5)

List the learning outcomes of the course. These must be assessable (i.e., observable), so must specify what the student should be able to do concretely, not simply what they should "understand". Use concrete verbs that indicate (a) what type of assessment would be appropriate, and (b) what level of knowledge/thinking is expected (from recall to analysis to novel creation). ***Example verbs:*** *define, explain, implement, compare, justify. Assessments (described later) should be tied to the learning outcomes.*

Outcomes should typically focus more on the types of thinking/skills developed than on the detailed course content, and the level of thinking should be appropriate to the level of the course: outcomes for a Level 11 course should include more higher-level thinking skills than for a Level 8 course. Further guidance on writing learning outcomes can be found at https://www.ncl.ac.uk/ltds/assets/documents/res-writinglearningoutcomes.pdf

On completion of this course, the student will be able to

1) Explain the modern techniques used in the design and development of large-scale software systems

2) Apply and evaluate these techniques in a small-scale, but real life, scenario

3) Work effectively as part of a team

4) Analyse the professional and ethical implications of software engineering decisions and propose solutions

5) Comfortably read and write technical documents, and interpret formal standards.

Graduate Attributes, Personal & Professional Skills

List the personal attributes and generic transferrable skills this course will help develop. Examples include
Cognitive skills: *problem-solving, critical/analytical thinking, handling ambiguity*
Responsibility, autonomy, effectiveness: *independent learning, self-awareness and reflection, creativity, decision-making, leadership, organization and time management, flexibility and change management, ethical/social/professional awareness and responsibility, entrepreneurship*
Communication: *interpersonal/teamwork skills, verbal and/or written communication, cross-cultural or cross-disciplinary communication*

This course develops a wide range of graduate attributes and skills across several areas:

- Cognitive skills: problem-solving, critical/analytical thinking, handling ambiguity.
- Responsibility, autonomy, effectiveness: independent learning, self-awareness and reflection, creativity, decision-making, organization and time management, flexibility and change management, ethical/social/professional awareness and responsibility.
- Communication: interpersonal/teamwork skills, verbal and written communication.

1. Student-facing course description and additional feedback and assessment information

Except where noted, all fields are required and will go into the DRPS entry for the course (for use by students). Important: any text in DRPS is effectively a contract with students, so should not include details that are likely to change from year to year.

Summary Description *Provide a brief official description of the course, around 100 words. This should be worded in a student-friendly way, it is the part of the descriptor a student is most likely to read. If this course replaces another course, please say so in this summary.*	Software Engineering and Professional Practice teaches the practice of small team software development in modern society, equipping students to participate in an agile workplace (such as a startup or modern tech company) or a software-dependent research team. Students will gain experience working with a large codebase using many of the key tools of the trade: testing, interacting with customers, handling bug reports, designing and implementing new features. Professional aspects of Software Engineering — its legal, ethical and social environment, including issues of privacy, security, equality, democracy and intellectual property — will be approached through guest lectures and some practical work.
Keywords *Give a list of searchable keywords.*	software engineering, professional practice, ethics
Course Description *A more detailed student-facing description of the course, which should normally include (a) a more in-depth academic description of the learning aims, nature and context of the course, (b) a rough outline of the content or syllabus, often as bullet points, and (c) a description of how the course will be taught, how students are expected to engage with their learning and how they will be expected to evidence and demonstrate their achievement of the intended learning outcomes.]*	As students enter this course they team up in groups of four, and take over a pre-existing medium-sized software project — one with incomplete features, some tests, some outstanding bug reports and inevitable design flaws. Over the next eleven weeks they get to grips with the architecture of the system, test it, extend those tests, deal with the existing bugs, find new bugs, talk to their customers and design and implement new features. This is intended to replicate the experience of working in a small agile software team in a commercial or research-oriented environment. Included in the experience will be use of industry standard tools for software development (integrated development environments, version control, issue tracking and continuous integration), and key elements of modern agile development practice, such as Scrum, code review, peer review, and pair programming. This will be supported by a small amount of interaction with mentors playing the role of “customers”. As students engage in this practical work, the course will contextualise it against the broader themes, both of large-scale Software Engineering and its academic literature, and of today's urgent professional issues: the legal, ethical and social context in which software and its authors exist. Guest lecturers will speak on topics such as privacy, security, equality, democracy and intellectual property — some of which will have a direct impact on students' practical work.

	The course is assessed through a mixture of group and individual work, mainly on software development but also on written reflective practice. Group participation will be taken into account. The first of the three assignment submissions is entirely formative, with the second and third, as moderated by participation, combining to form the overall course mark.
Assessment Weightings: *These should correspond approximately to the proportion of learning outcomes that each component assesses. More than 30% coursework requires specific justification.* *The expectation for a 10pt course is 20% coursework with the equivalent of one 15-20hr assessed assignment (but possibly split into smaller pieces). See 'components of assessment' below.*	Written Exam __0___% Practical Exam __0___% (*for courses with programming exams)* Coursework __100___%
Further Assessment Information *Provide any further information that should go on DRPS for students. E.g., if the assessment includes required group work or if students must pass some individual component of assessment as well as the course overall.*	Assessment is based on programming and software development tool use, writing, and a small amount of online and in-person roleplay with "customers". Work is assessed both individually and collectively as part of a small team. Individual receipt of group marks will be moderated according to peer- and mentor-assessed contribution.
Components of assessment and time spent on assignments (for BoS only) *If not already included in the course narrative description, please describe the type of assessments (oral presentation, report, programming, etc) and* ***how each component of assessment will assess the intended learning outcomes****. Where coursework involves group work, it is important to remember that every student has to be assessed individually for their contribution to any jointly produced piece of work.* *Also estimate* ***how many hours*** *students will spend on assignments. Please see the School policy on Workload and Assessment, which states that students should not be expected to spend more than 6-7 hrs/wk per 10 credits, including contact hours.* *Note that it often desirable to include formative assignments which are not formally assessed but submitted for feedback, often in combination with peer assessment.*	We plan for coursework 1 to be formative. The student groups will each need to submit their reflective writing to date - incorporating a discussion/evaluation of the approaches, tools and techniques used, considerations of professional practice, their group's interaction - as well as their code (i.e. the improvement of the provided code base), tests and JavaDoc documentation. Formative feedback and information-only marks will be given for the group as a whole. Courseworks 2 and 3 will be summative. For coursework 2, the student groups will each need to submit all new writing, code, tests and JavaDoc, as well as (re-)submit their technical solution from coursework1. For coursework 3, we will also invite a comparison with others working on the same functionality, and responses to a larger ethical issue. The individual's participation through git commits, peer reviews and peer and mentor opinion will be used both to provide some marks for participation, and to moderate distribution of group work marks. Two opportunities for peer review of classmates developing different functionality to the student's - one before coursework 2 and one before coursework 3 - will offer good sources of additional feedback. To facilitate marking as well as self-assessment, a rubric-based marking scheme will be used by the markers. Moreover, we plan to provide this marking scheme for the students' self-assessment for the first two coursework deadlines.

	Students will be expected to spend around 4.5 hours a week, plus the 2 hours lab time with the mentor and the 15 minutes with the client (around 7 hours total) working on the assignments (not considering study time in preparing for them).
Feedback Information *Provide a high-level description of how and what type of feedback will be provided to students, for inclusion in DRPS.*	Students will be provided with formative feedback for their first coursework, as well as through interaction with mentors and peers. Mentor meetings will offer feedback on the quality of the produced solution to date, the use of good Software Engineering approaches, tools and techniques, the discussion of teamwork and other professional issues, and conformance to coursework instructions. A small number of roleplayed ”customer” meetings will focus on interpretation of requirements and evaluation of the solution. Students will be provided with summative feedback for their second and third coursework. This feedback will consist of both group feedback and individual feedback personalised to each student.
Additional Feedback Information (for BoS use only) *If not already included in the course narrative, provide further details on planned feedback arrangements. This includes how course feedback is solicited from the class and responded to, as well as what feedback students will get (either on work that contributes to their final mark, or not).* *The University is committed to a baseline of principles regarding feedback that we have to implement at every level, and the School encourages submission of at least one piece of written work for formative feedback.* *In general, formative feedback:* • *Should say how students can improve.* • *Need not be on individual work (e.g., consider a lecture or document summarizing common issues.)* • *Can include oral feedback during labs/tutorials* • *Can include feedback from peers* • *Clickers/TopHat/equivalents can provide in-class feedback for both students and lecturer.* • *Is returned in time for other forms of assessment to which it relates, to allow feedforward.*	Course feedback will be solicited both from students and the course team. From students, we aim to conduct mid-term feedback using a combination of a short TopHat quiz to be used in a lecture and a separate online questionnaire to be used outside the lecture. Moreover, student progress in coursework and as reported by the mentors will be a good source of information.End of term feedback from students will be collected through the official Course Enhancement Questionnaire. We also aim to gather mid-term and end of term feedback from the mentors, through online chats and a final face-to-face discussion. Future course components and sources of support will be adapted in line with the feedback results. Students will be provided with formative feedback for their first coursework (in writing and as in-lecture feedback), as well as through interaction with mentors, customers and peers (during peer review). The formative feedback from the first coursework and mentor meetings will touch on the quality of the produced solution to date, the use of good Software Engineering approaches, tools and techniques, the discussion of professional issues. Constructive advice on how to improve and hints to reading will be provided. We will aim to deliver feedback on the first coursework to the students within at most 10 days of the deadline (as opposed to the typical 14 days for summative assessment) so that they have time to consider it for their next deadline. During mentor meetings, groups will also be advised on improving their team work and software engineering process. During customer meetings, feedback on the interpretation of the requirements and the evaluation of the solution will be provided. Students will be provided with summative feedback for their second and third coursework, both in writing and as in-lecture

<table>
<tr><td></td><td>feedback. This feedback will be of a similar standard to the formative feedback. Students will be given both their group feedback, as well as feedback on their individual work. For coursework 2, we will aim to deliver this feedback to the students within at most 10 days of the deadline (as opposed to the typical 14 days for summative assessment), as it may allow students to improve their solution for the final deadline.</td></tr>
<tr><td>Breakdown of Learning and Teaching Activities
State how many hours students spend on each part of the course. The total should be 10 x course credits, but please also see the School policy on Workload and Assessment.which states that students should not be expected to spend more than 6-7 hrs/wk per 10 credits, including contact hours.

Assume 10 weeks of lectures slots and 10 weeks of tutorials, but these need not all be used. As a guideline, a 10-pt course typically has 18-20 lecture hours, but should have only around 15 lectures of examinable material; the rest should be used for guest lectures, revision sessions, introductions to assignments, etc.</td><td>Contact hours
<table><tr><th>Hours</th><th>Type</th></tr><tr><td>32</td><td>Lecture Hours</td></tr><tr><td>0</td><td>Seminar/Tutorial Hours</td></tr><tr><td>0</td><td>Dissertation Project Supervision Hours</td></tr><tr><td>22 (12 of which also considered for summative assessment hours)</td><td>Supervised practical/Workshop/Studio hours</td></tr><tr><td>4</td><td>Feedback/Feedforward hours</td></tr><tr><td>27 outside contact hours, 12 during workshops</td><td>Summative assessment hours</td></tr><tr><td>0</td><td>Revision Session Hours</td></tr></table>
Non-contact hours
<table><tr><th>Hours</th><th>Type</th></tr><tr><td>115</td><td>Directed Learning & Independent Learning hours</td></tr></table>
Total hours: 200</td></tr>
<tr><td>Reading List/Learning Resources
You are encouraged to create resource lists using LEGANTO</td><td>Sommerville “Engineering Software Products”
ACM code of ethics: https://www.acm.org/code-of-ethics
BCS code of conduct: https://www.bcs.org/membership/become-a-member/bcs-code-of-conduct/</td></tr>
</table>

Further information for BoS consideration: sample materials

A full proposal for a new course must include examples of exercises and assessment. Please provide these below, along with publicity information if the course is to be advertised outwith the School.

Course information and publicity *The course web page (typically the Learn landing page) will be linked from the Sortable Course List, and information such as timetables and assignment deadlines must be made available prior to the start of the academic year. Please specify here if any additional info/publicity is needed for your course: typically only if it is aimed largely at non-SoI students.*	No additional information/publicity is needed for this course.
Sample tutorial/lab sheet questions *Provide a list of tutorial questions and answers and/or samples of lab sheets. These need not be fully fleshed out but should indicate what sort of exercises will be provided to help students learn the material.*	There will be no lab sheet questions. The mentors will be provided with a more detailed system description than that provided to the students, so that they can act more as customers while offering consistent advice.
Sample assessment materials *If the course is primarily assessed by **exam**, provide a sample exam question with model answers. Any non-standard exam format must be justified. The online list of past exam papers gives an idea of typical and alternative exam formats: http://www.inf.ed.ac.uk/teaching/exam_papers/.* *If the course is largely or primarily assessed by **coursework**, provide a sketch of a possible assignment with an estimate of effort against each sub-task and a description of marking criteria.*	One idea we are considering for the coursework is a trading simulator: something which executes various trading strategies in simulated financial markets, and presents the results. Students work on this in teams of 4 — or as close to this as possible, to support an introduction of pair programming later on. Students write together on a weekly basis throughout the course, on both course content and team experience. An incomplete but somewhat functional implementation will be provided at the start of course, with various flaws. A class diagram will be provided, but the implementation diagram will be slightly off: the first exercise will be to discover this and do some basic reconciliation between the two. This offers students a chance to familiarise themselves with the codebase while introducing some of the ideas of formal graphical representations and justifying their use in larger projects. Some bug reports are already seeded in the system, and students are introduced to the process of debugging and asked to investigate and fix these bugs if possible. Some bugs may be problematic — not reproducible, or not bugs. The students are also introduced to the existing tests, some of which fail, and a portion of the code which has no tests. They fix the existing failures, and write new tests and fix failures which those detect. Some sharing of tests will take place at this point, and students will get to run each others' tests against their own code. A writing topic while working on tests could ask students to discuss at what point they would consider a piece of

	software to be “safe”, “reliable”, etc. This might mix terms with technical and non-technical meaning, opening up a space to talk about public communication of risk. The above is assessed formatively, considering the writing, tests and JavaDoc, and paralleled by lectures introducing the formal ideas behind the practice, interleaved with some bigger picture material. The second, assessed assignment requires the students to read a standards document which the implementation partially fulfils (an API of some sort, possibly for new trading strategies) and complete the missing elements. This will break down into easily divisible parts so that this code is completed and assessed individually. Beyond the two hour labs each week, drop-in sessions with mentors acting as customers will provide students with 15-minute slots to resolve ambiguities and contradictions between code, standard, and assignment spec. Peer reviews of another student’s module from another group are also organised. Reflective writing at this point might ask about the students’ experience of writing to a standard: whether that constrained them or benefited them. After receiving copies of each others’ code they could be asked to write about openness, plagiarism, and the “public code” movement. The second assignment is assessed summatively, considering again the writing, tests and JavaDoc, as well as a (re-)submission of the code, tests and JavaDoc from the first assignment. The third assignment comprises two parts. It presents students with an ethical critique of the software, demonstrating that its behaviour so far disfavours a certain class of investor or company. This will be derived from known data, such as disparities in gender balance of corporate CEOs, risk preference by age, or similar. A considered response to this will be required, either justifying the outcome, suggesting remedies, or both. The second part will invite the students to do some relatively light work on the software user interface, either extending or adjusting it, with usability or other goals in mind. Depending on scale and divisibility this may be assessed as group work or individually.

	Peer reviews of another student's module are organised again. Students can also see everybody else's solution to their assignment 2 module shortly before the third assignment deadline. The third assignment is assessed summatively, considering written reports (comparison with others doing the same module, response to ethical issues, usability discussion), participation (git commits, peer reviews, opinions of students and mentor), code, tests and potentially improved GUI (minor GUI work, maybe some sketches in report). With three deadlines spread evenly across the course, we aim for a steady balance of student effort, with some relief in weeks 10 and 11 since the closing "big picture" materials are not examinable through coursework. The first deadline is formative — with information-only marks — as we onboard students into the team development experience, so final marks will be split evenly between the second and third submissions. Within these, we intend programming and tool use (handling of version control, bug reporting system, testing) to account for about 70% of the mark, and writing/reflective practice 30%. The group/individual split will reflect that in the assignments, which are anticipated to emphasise individual work in the second assignment and group work in the first and third. Individual marks for group work will be moderated by the student's participation, and some marks will be explicitly given for participation.
Any other relevant materials *Include anything else that is relevant, possibly in the form of links. If you do not want to specify a set of concrete readings for the official course descriptor, please list examples here.*	One standard that might be interesting for students to work with would be OAuth, allowing them to authenticate system users with an open identity service: this would facilitate a discussion of cloud services and provide them with a potentially useful bit of experience with third party software. Its use may require some degree of scaffolding so as not to be overwhelming though. https://oauth.net/

1. Additional Course Details for DRPS

Except where otherwise noted, these fields are required for entry into EUCLID and will be visible to students in the DRPS entry.

Planned Academic Year of Delivery *(The first year you anticipate the course running, e.g. AY 2019-20)*	2020-2021
Course Organiser *(By default, the course proposer)*	Cristina Adriana Alexandru
Intended Delivery Period	__Semester 1 _X_Semester 2 __Full Year __Summer __Other (please specify):
Timetable considerations/conflicts *For School use. Please specify any constraints to be considered (e.g. overlap of popular combinations, other specialism courses, external courses etc). Include whether the semester delivery is constrained or could be flexible.*	
Is this course available to visiting students?	__Yes (default) _X_No **If no, please provide a justification here:**
Required pre-requisite courses *Use sparingly: these are enforced in PATH and can only be waived by approval from the School's Curriculum Approval Officer. Note that cross-year required pre-requisites may prevent MSc students from registering; consider using recommended pre-requisites or "other requirements" instead.*	__No _X_Yes (please specify full course name(s) and code(s)): Inf1- Introduction to Computation Inf1B
Recommended pre-requisite courses	_X_No __Yes (please specify full course name(s) and code(s)):
Required co-requisite courses *Specify any courses that must be taken in parallel with the existing course. Note that this leads to a timetabling constraint that should be mentioned elsewhere in the proposal.*	_X_No __Yes (please specify full course name(s) and code(s)):

Prohibited Combinations *Specify any courses that may not be taken in combination with the proposed course].*	_X_ No ___ Yes (please specify full course name(s) and code(s)):
Other Requirements/Additional Information *This information is often used by MSc students and students from other Schools to see if they have appropriate background without having done our School's courses. So please avoid course titles, instead list specific knowledge and skills (such as mathematical concepts, programming ability or specific languages, etc).* *Also list any other constraints on registration, for example: "Only available to 4th Year Informatics students including those on joint degrees." or "This course is open to all Informatics students including those on joint degrees, and to students in the School of Mathematics. Other external students whose DPT does not list this course should seek permission from the course organiser."*	__ No _x_ Yes (please specify): Only open to 2nd year Informatics students, including those on joint degrees. Prerequisite knowledge of object oriented programming required.
Visiting Student Pre-requisites	___ Same as "other requirements" ___ Different than "other requirements" (please specify):

2. Placement in degree programme tables: for level 9-11 courses only

This section is for consideration by the Board of Studies and will be used later by ITO to determine where the course will be added to existing degree programme tables.

Is this course restricted to students on a specific degree? *E.g., some courses are only available to students on a specific CDT or MSc.*	___No ___Yes (please specify and provide justification):
Is this course compulsory for students on any degree(s)?	__No __Yes (please specify and provide justification):
Any issues for part-time students? *Normally, part-time students have access to the same courses as full-time students on the equivalent degree. If you anticipate any problems with this, please specify here.*	

For optional courses:

If this course is available but non-compulsory for students on various degrees (most courses), please fill in this section. The choices here determine where the course appears in degree programme tables (DPTs) and the 2-3 character tags are displayed in the Informatics sortable course list.

Should this course be tagged as 'ML' (machine learning foundations and methods)? *Courses with the ML tag are typically very high-demand and most degrees limit the number of ML credits. If your course might appeal to a similar audience but draw off students from these large courses, please select 'no' and choose one of the tags below.*	__No __Yes
If you chose 'no', please choose at least one of the following tags… *Ideally, select exactly one, unless there is a good argument for more than one. These three are used in various combinations for many of our degrees.*	__**FSS** (CS foundations, systems, and software) __**AIA** (artificial intelligence applications and paradigms) __**COG** (cognitive science: including HCI and NLP courses, but not most other AI courses. Please restrict to courses most relevant to natural cognition.)
…and also tick if any of the following tags or categories apply. *Do not tick any of these if you selected 'ML' already.*	__**NS** (natural systems: e.g., computation by or about biological or social systems. Many COG courses are also NS. This tag is mainly relevant for MSc in Informatics.) __**SE** (software engineering: including courses that are highly relevant to SE degrees. All SE courses should also be FSS. This tag is mainly relevant for UG SE degrees.) __Databases and data management systems (used for Data Science MSc and MSc(R)) __Unstructured data and applications (used for Data Science MSc and MSc(R))

	__Level 11 Security courses (used for Security MSc)
If you are not sure which tags are most appropriate or have other questions about this section, please note any comments/issues here.	

3. Comments from colleagues

All course proposal should be sent to relevant colleagues in the area as well as to the appropriate year organizer and BoS Academic Secretary for comment in good time before the BoS meeting. Please indicate here what feedback has been solicited and received.

Additional Comments *Summarise any comments received from relevant individuals prior to proposing the course. If you have not discussed this proposal with others please note this.*	In writing this document, we have consulted with: ● Lecturers who have taught Inf2C-SE: Paul Jackson, Perdita Stevens, Nigel Goddard, Ajitha Rajan ● Lecturers who are teaching related courses: ○ Inf2B: Paul Anderson, Volker Seeker ○ SDM: Perdita Stevens ○ ST: Ajitha Rajan ○ Professional Issues: Stuart Anderson ○ SDP: Barbara Webb ● Judy Robertson, who is specialised in Pedagogy ● The Director of Teaching (Stuart Anderson) and Deputee Directors of Teaching (Sharon Goldwater, Paul Patras) ● Administrative (Gillian Bell), learning technology (Alex Burford) and library staff (Angela Nicholson) members ● Current tutors and demonstrators on Inf2C-SE, most of whom have also been students on a previous iteration of Inf2C-SE We have also considered past and current student feedback on Inf2C-SE from last year's Course Enhancement Questionnaire and this year's mid term feedback collected through TopHat and a Jisc survey Considering initial input from Stuart Anderson, Sharon Goldwater, Paul Jackson, Volker Seeker, Paul Anderson, Perdita Stevens and Judy Robertson, we first decided to organise an ELDeR workshop, to have the opportunity to brainstorm ideas with colleagues. This workshop took place on the 17th and 18th of September. Its participants were: Paul Jackson, Paul Anderson, Paul Patras, Gillian Bell, Alex Burford and Angela Nicholson. Vidminas Mikucionis (who is a tutor on the current iteration of Inf2C-SE and a former student on this course) joined us as a 'critical friend' at the end of the workshop to provide feedback on our plan. The decisions reached as part of ELDeR were materialised in the first draft of this proposal. Going further, we organised discussions with more lecturers as recommended at the first BoS: Ajitha Rajan, Nigel Goddard, Barbara Webb. Moreover, we have considered past and current student feedback on Inf2C-SE, and held a one-hour feedback discussion with the current tutors and demonstrators from Inf2C-SE, most of whom have also been students on a previous iteration of Inf2C-SE.

Year Organiser Comments *Year Organisers are responsible for maintaining the official Year Guides for every year of study, which, among other things, provide guidance on available course choices and specialist areas. The Year Organisers of all years for which the course will be offered should be consulted on the appropriateness and relevance on the course. Issues to consider here include balance of course offerings across semesters, subject areas, and credit levels, timetabling implications, fit into the administrative structures used in delivering that year.]*	
BoS Academic Secretary Comments *Proposals must be checked by the Secretary of the Board of Studies prior to discussion at the actual Board meeting. This is a placeholder for their comments, mainly on the formal quality of the content provided above.*	