THE UNIVERSITY of EDINBURGH
informatics

THE UNIVERSITY of EDINBURGH
informatics

Board of Studies

Course Proposal Template

PROPOSED COURSE TITLE: Software Design and Modelling

PROPOSER(S): Perdita Stevens

DATE: 22/10/15

SUMMARY

This template contains the following sections, which should be prepared roughly in the order in which they appear (to avoid spending too much time on preparation of proposals that are unlikely to be approved):

1. Case for Support
– To be supplied by the proposer and shown to the BoS Academic Secretary prior to preparation of an in-depth course description

1a. Overall contribution to teaching portfolio

1b. Target audience and expected demand

1c. Relation to existing curriculum

1d. Resources

2. Course descriptor
- This is the official course documentation that will be published if the course is approved, ITO and the BoS Academic Secretary can assist in its preparation

3. Course materials
- These should be prepared once the Board meeting at which the proposal will be discussed has been specified

3a. Sample exam question

3b. Sample coursework specification

3c. Sample tutorial/lab sheet question

3d. Any other relevant materials

4. Course management

- This information can be compiled in parallel to the elicitation of comments for section 5.

4a. Course information and publicity

4b. Feedback

4c. Management of teaching delivery

5. Comments

- To be collected by the proposer in good time before the actual BoS meeting and included as received

5a. Year Organiser Comments

5b. Degree Programme Co-Ordinators

5c. BoS Academic Secretary

[Guidance in square brackets below 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://www.inf.ed.ac.uk/admin/committees/bos/meetings/.]

SECTION 1 – CASE FOR SUPPORT

[This section should summarise why the new course is needed, how it fits with the existing course portfolio, the curricula of our Degree Programmes, and delivery of teaching for the different years it would affect.]

1a. Overall contribution to teaching portfolio

[Explain what motivates the course proposal, e.g. an emergent or maturing research area, a previous course having become outdated or inappropriate in other ways, novel research activity or newly acquired expertise in the School, offerings of our competitors.]

The current 10pt course Software Engineering with Objects and Components is a key element of the software engineering teaching in Informatics (effectively compulsory for SE students, and taken by many others, both UGs and MScs). Mostly in an object oriented context, it covers software design principles that help tell good design from bad, basic design patterns which record known-good design solutions, modelling using the Unified Modeling Language (UML) (including formal specification with the Object Constraint Language), and the relationship between these techniques and the software development process. Students are required to know Java, and code examples are given in Java; but the focus is on design and modelling, not programming, and there is no compulsory building of software in the course. The course is delivered in a partially ``flipped'' form, with reading and videos covering the basic factual material and ``lecture'' sessions devoted to exercises and discussion (though this format has proved difficult in recent years with increasing numbers (over 100 in 2015-16), and I have been using more standard lectures: the flipped format worked rather better with the 30-40 of previous years).

I believe the course is broadly successful, as judged both by its popularity and by the quite impressive level of achievement of students by the time of the final exam. However,

- students have rightly pointed out that the material in this course would benefit from being more explicitly connected with the programming that implements the designs. Some students find it difficult to grasp the abstract design concepts at all without connecting them explicitly to concrete implementations. Even those who do well in the current course often express that they would prefer a more joined-up view.

- Given the gradual emergence of tool-supported model-driven development from its niches into the mainstream, it would be highly beneficial to introduce the use of modelling tools, including tools for code generation, into the course.

I am attempting to give ``tastes'' of both of these changes within the current course, by giving pointers to work that interested students who have time can do in these areas. However, it is impractical to introduce these changes in a serious way within the 10pt course - getting to grips with code or with a new tool takes time, and all the material currently in SEOC is basic enough that it should not be evicted.

It was brought home to me in the course of writing [1] that we are indeed attempting to fit a quart into a pint pot with this course: other universities spend far more time on this material.

Therefore, this proposal is to replace SEOC by a new 20pt course, Software Design and Modelling, to contain a superset of SEOC's material, extending it by use of tools and connection to code as mentioned above. I polled the cohort of SEOC students in 2014-15 (at the end of the course) about whether they would have been likely to take such a course if it had been offered instead of SEOC, and what they thought of the idea. The reaction was overwhelmingly positive. While a few students did,

as one would expect, say that they would not have been able to spare 20pts and so would not have taken the course, it was only a few; most students who responded said they would have taken the 20pt version and many were strongly supportive of the suggested change.

[1] Seiko Akayama, Birgit Demuth, Timothy C. Lethbridge, Marion Scholz, Perdita Stevens, Dave R. Stikkolorum. Tool Use in Software Modelling Education. In Post-proceedings of the Educators' Symposium of MODELS 2013. Available from: http://ceur-ws.org/Vol-1134/paper6.pdf

1b. 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.]

Students in UG3-5 or MSc years who have an interest in software engineering, e.g., plan to go into employment in that area, and who have background in basic software engineering such as is presented in Inf2C-SE and in object oriented programming such as is presented in Inf1OP. Student comments indicate that they see the course as highly relevant to employment.

As the proposed course replaces SEOC we look first at that to estimate expected demand. The demand for SEOC rose in 2014-15 to over 80 students, having been in the 40s for the previous few years. (It is unclear what caused the large rise: I initially thought it was because it was opened up to UG4s for the first time, that is, that it might have been a blip, but this year's SEOC has well over 100 students.) My polling of last year's SEOC cohort suggested that, even though the new course would require students to commit 20pts rather than 10pts, it would not be significantly less popular.

So I suggest one should plan for at least 40 students, and not be too surprised to get a lot more. **This means the course must have a cap at the capacity of the largest computing lab we can allocate to the taught lab session**, and students on degrees for which it is (effectively) compulsory must be given priority. (Labs in this course will be comparable to lectures in others: taught by the course lecturer with content that must be available to all, even if it veers off the planned trajectory. These are not self-guided labs, which are more scalable. In this case “just put on another lab” should not be regarded as any less of a step than “just put on another lecture” - something we've never yet done.)

1c. Relation to existing curriculum

[This section should describe how the proposed course relates to existing courses, programmes, years of study, and specialisms. Every new course should make an important contribution to the delivery of our Degree Programmes, which are described at http://www.drps.ed.ac.uk/15-16/dpt/drps_inf.htm.

Please name the Programmes the course will contribute to, and justify its contribution in relation to courses already available within those programmes. For courses available to MSc students, describe which specialism(s) the course should be listed under (see http://web.inf.ed.ac.uk/infweb/student-services/ito/students/taught-msc-2015/programme-guide/specialist-areas), and what its significance for the specialism would be. Comment on the fit of the proposed course with the structure of academic years for which it should be offered. This is described in the Year Guides linked from http://web.inf.ed.ac.uk/infweb/student-services/ito/students.]

Replaces SEOC, strengthening the Software Engineering programme, as described above. Should be listed for MSc under "Computer Systems, Software Engineering & High-Performance Computing". Significance is that this is a core area of SE.

Best offered as a first-semester course, as then UG3 students can take advantage of what they have learned when they do SDP in second semester, and UG4+ students in their projects. This is not an absolute constraint.

1d. Resources

[While course approvals do not anticipate the School's decision that a course will actually be taught in any given year, it is important to describe what resources would be required if it were run. Please describe how much lecturing, tutoring, exam preparation and marking effort will be required in steady state, and any additional resources that will be required to set the course up for the first time. Please make sure that you provide estimates relative to class size if there are natural limits to its scalability (e.g. due to equipment or space requirements). Describe the profile of the course team, including lecturer, tutors, markers, and their required background. Where possible, identify a set of specific lecturers who have confirmed that they would either like to teach this course apart from the proposer, or who could teach the course in principle. It is useful to include ideas and suggestions for potential teaching duty re-allocation (e.g. through course sharing, discontinuation of an existing course, voluntary teaching over and above normal teaching duties) to be taken into account when resourcing decisions are made.]

Lecturer: me, Perdita Stevens.

Teaching assistant, to help present the lab sessions and, the first time of presentation, to help design the lab exercises. (Essential at first presentation; in steady state, nice to have if the class size is in the 30s, essential if it's in the 60s or above, to ensure students can get help with tool problems etc.)

Marking assistant(s), to help provide prompt detailed feedback on the summative practical exercise and on formative work students do during the rest of the course.

Scalability: the absolute limiting factor is the size of a computing lab (I don’t know what this will be in the reopened Appleton Tower: around 100-150?). It would be possible to use a central, IS-provided lab if that could be timetabled and would get us more capacity, since the software I plan to use is likely to be available on both Linux and Windows: but from earlier discussion of programming exams, I don’t think that helps much.

Since the course will be available to both UG3 and UG4, we could consider offering this course less than every year. However, I think in practice that would lead to undesirably huge class size in the years it was offered. (This course would first be offered in 2016/17, and I am due for sabbatical in 2017/18: in the absence of an obvious alternative person to teach it, I would suggest that the School consider not offering this course in that year, and advertise that plan from the beginning (i.e. so that 2016/17's UG3 students do not assume they can always take it in their UG4 year). We would then see from class size in 2016/17 and 2018/19 whether it's feasible to continue in an alternate year pattern or not!)

SECTION 2 – COURSE DESCRIPTOR

[This is the official course descriptor that will be published by the University and serves as the authoritative source of information about the course for student via DRPS and PATH. Current course descriptions in the EUCLID Course Catalogue are available at www.euclid.ed.ac.uk under 'DPTs and Courses', searching for courses beginning 'INFR'*]*

2a. Course Title *[Name of the course.]* : Software Design and Modelling

2b. SCQF Credit Points: 20

[The Scottish Credit and Qualifications Framework specifies where each training component provided by educational institutions fits into the national education system. Credit points per course are normally 10 or 20, and a student normally enrols for 60 credits per semester. For those familiar with the ECTS system, one ECTS credit is equivalent to 2 SCQF credits. See also http://www.scqf.org.uk/The%20Framework/Credit%20Points.]

SCQF Credit Level: 10

[These levels correspond to different levels of skills and outcomes, see http://www.sqa.org.uk/files_ccc/SCQF-LevelDescriptors.pdf. At University level, Year 1/2 courses are normally level 8, Year 3 can be level 9 or 10, Year 4 10 or 11, and Year 5/MSc have to be level 11. MSc programmes may permit a small number (up to 30 credits overall) of level 9 or 10 courses.]

Normal Year Taken: 3

[While a course may be available for more than one year, this should specify when it is normally taken by a student. "5" here indicates the fifth year of undergraduate Masters programmes such as MInf.]

Also available in years: 4(5)MSc

Different options are possible depending on the choice of SCQF Credit Level above: for level 9, you should specify if the course is for 3rd year undergraduates only, or also open to MSc students (default); for level 10, you should specify if the course is available to 3rd year and 4th year undergraduates (default), 4th year undergraduates only, and whether it should be open to MSc students; for level 11, a course can be available to 4th and 5th year undergraduates and MSc students (default), to 5th year undergraduates and MSc students, or to MSc students only]

2c. Subject Area and Specialism Classification: SE

[Any combination of Computer Science, Artificial Intelligence, Software Engineering and/or Cognitive Science as appropriate. For courses available to MSc students, please also specify the relevant MSc specialist area (to be found in the online MSc Year Guide at http://web.inf.ed.ac.uk/infweb/student-services/ito/students/taught-msc-2015/programme-guide/specialist-areas), distinguishing between whether the course should be considered as "core" or "optional" for the respective specialist area.]

Appropriate/Important for the Following Degree Programmes:

[Please check against programmes from http://www.drps.ed.ac.uk/15-16/dpt/drps_inf.htm to determine any specific programmes for which the course would be relevant (in many cases, information about the Subject Area classification above will be sufficient and specific programmes do not have to be specified). Some courses may be specifically designed for non-Informatics students or with students with a specific profile as a potential audience, please describe this here if appropriate.]

Important for: all degree programmes whose titles include Software Engineering

Appropriate for: any degree programme in this School. (Unlikely to be suitable for significant numbers of students from elsewhere: SEOC typically gets a few enquiries but hardly any enrollments once the prerequisites are understood.)

Timetabling Information:

[Provide details on the semester the course should be offered in, specifying any timetabling constraints to be considered (e.g. overlap of popular combinations, other specialism courses, external courses etc).]

Semester 1, ideally. Not to overlap with any other course in subject area SE.

2d. Summary Course 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.]

This course introduces the design and modelling of software systems using object-oriented techniques. We start by exploring the use of modelling in software development. Students learn to document designs in the Unified Modeling Language, UML, with emphasis on class, sequence and state diagrams and the Object Constraint Language, OCL. We use modern model-driven development tools and discuss their strengths and weaknesses. We study criteria that make one design better than another in context and introduce design principles and patterns that capture good practice.

Course Description:

[Provide an academic description, an outline of the content covered by the course and a description of the learning experience students can expect to get. See guidance notes at: http://www.studentsystems.ed.ac.uk/Staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html#AcademicDescription].

Note: the above link is broken. I think it should be to

http://www.studentsystems.is.ed.ac.uk/staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html#AcademicDescription

The course begins by placing design and modelling in the context of the various software engineering processes in widespread use today. Via labs, lectures and self-study using readings, videos and formative exercises, it teaches students to produce (initially straightforward) designs and to document them using UML models, both on paper and with an appropriate tool. We discuss the different affordances of these modes and how models may be used (i) purely informally, (ii) as reviewed documentation for designs, and/or (iii) in model-driven development in which models are formal artefacts and code may be generated from them.

The second part of the course focuses on identifying and producing good designs. What principles should a good object-oriented design follow? We learn some common design patterns and their role in development and learning.

Finally we turn to practical model-driven development: how can the cost-benefit ratio of modelling be improved, now and potentially in the future? Students will learn about model transformations, both model-to-model and model-to-text (e.g., code generation) and be introduced to current tools supporting these. We discuss the role of domain specific languages and the integration of model driven development with agile processes. Throughout the course, we identify the deficiencies as well as the benefits of the fast-changing state of the art, aiming to equip students to critically evaluate tools and techniques that become available to them in future.

Pre-Requisite Courses:

[Specify any courses that a student must have taken to be permitted to take this course. Pre-requisites listed in this section can only be waived by special permission from the School's Curriculum Approval Officer, so they should be treated as "must-have". By default, you may assume that any student who will register for the course has taken those courses compulsory for the degree for which the course is listed in previous years. Please include the FULL course name and course code].

Informatics 1 - Object Oriented Programming (INFR08014)

Informatics 2C - Software Engineering (INFR08019)

- if this doesn't give us a problem with allowing appropriately experienced MSc students to take it? NB although Inf1OP is compulsory for our undergraduate students, it may be helpful to be explicit that the course depends on it, for the sake of these MSc students.

Co-Requisite Courses:

[Specify any courses that should be taken in parallel with the existing course. Note that this leads to a timetabling constraint that should be mentioned elsewhere in the proposal. Please include the FULL course name and course code].

None

Prohibited Combinations:

[Specify any courses that should not be taken in combination with the proposed course. Please include the FULL course name and course code].

Software Engineering with Objects and Components (INFR10056)

Other Requirements:

[Please list any further background students should have, including, for example, mathematical skills, programming ability, experimentation/lab experience, etc. It is important to consider that unless there are formal prerequisites for participation in a course, other Schools can register their students onto our courses, so it is important to be clear in this section. If you want to only permit this by special permission, a statement like "Successful completion of Year X of an Informatics Single or Combined Honours Degree, or equivalent by permission of the School." can be included.]

Available to Visiting Students: Yes

[Provide a justification if the answer is No.]

2e. Summary of Intended Learning Outcomes (MAXIMUM OF 5):

[List the learning outcomes of the course, emphasising what the impact of the course will be on an individual who successfully completes it, rather than the activity that will lead to this outcome. Further guidance is available from https://canvas.instructure.com/courses/801386/files/24062695]

To assist BoS only: C/E indicates principally assessed in the coursework vs in the exam. Omit these markers from the course descriptor.

1 Design simple object-oriented systems. (C)

2 Create, read and modify UML diagrams documenting designs, both on paper and in an appropriate tool. (C,E)

3 Determine whether a UML model and a body of Java code are consistent; if they are inconsistent, identify the inconsistencies precisely and propose remedies. (C)

4 Evaluate and evolve object-oriented software designs, making use of common design patterns if appropriate. (E)

5 Discuss the use of modelling and model-driven development tools in software development, e.g. why and how models of software can have varying degrees of formality, capabilities and limitations of the tools. (E)

Assessment Information

[Provide a description of all types of assessment that will be used in the course (e.g. written exam, oral presentation, essay, programming practical, etc) and how each of them will assess the intended learning outcomes listed above. 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. Please include any minimum requirements for assessment components e.g. student must pass all individual pieces of assessment as well as course overall].

"Coursework" element is an individual exercise done in the scheduled lab session, provisionally in week 6. This is technically similar to our programming exams: done open book, on locked down DICE machines, and involving students submitting files electronically for marking. It concentrates on assessing the first 3 learning objectives. Because this exercise is to be done during semester, in a normal lab slot, not during the exam diet after teaching has completed, I have classified it as “coursework” rather than “practical examination”. There are several motivations for doing it this way: one is that it is important students are not tempted to postpone reaching competence in the early learning objectives, because the later weeks of the course will require that competence; another is to

help make apparent to students that a prolonged period of revision is not required, as the coursework will assess the skills practised in the early lab sessions. I hope this will reduce stress. As this is a single element of assessment accounting for 50% of the final mark, it should be sent to the External Examiner for comment, however: this could most conveniently be done by enclosing it with the resit exam papers in the summer.

The written examination concentrates on assessing the last two learning objectives, plus the "on paper" part of the second.

I have considered imposing hurdles on the coursework and exam components, but on the whole, do not consider it necessary.

Assessment Weightings:

Written Examination: 50%

Practical Examination: 0%

Coursework: 50%

Time spend on assignments: 0

[Weightings up to a 70/30 split between exam and coursework are considered standard, any higher coursework percentage requires a specific justification. The general expectation is that a 10-point course will have an 80/20 split and include the equivalent of one 20-hour coursework assignment (although this can be split into several smaller pieces of coursework. The Practical Examination category should be used for courses with programming exams. You should not expect that during term time a student will have more than 2-4 hours to spend on a single assignment for a course per week. Please note that it is possible, and in many cases desirable, to include formative assignments which are not formally assessed but submitted for feedback, often in combination with peer assessment.]

(Why is the heading about time, but the explanatory text about assessment weighting? That gives an unfortunate impression. In this case, while the individual study will include spending some time on formative exercises, splitting the time for those out of the general individual study time does not seem sensible.)

Academic description:

[A more technical summary of the course aims and contents. May include terminology and technical content that might be more relevant to colleagues and administrators than to students.]

(This seems to be a repeated section. See above.)

Syllabus:

[Provide a more detailed description of the contents of the course, e.g. a list of bullet points roughly corresponding to the topics covered in each individual lecture/tutorial/coursework. The description should not exceed 500 words but should be detailed enough to allow a student to have a good idea of what material will be covered in the course. Please keep in mind that this needs to be flexible enough to allow for minor changes from year to year without requiring new course approval each time.]

(This doesn't correspond to anything in the current DRPS, though an old version of that used to have a section with this heading! See 2d and 2e above, which between them include all the information in a typical current DRPS entry, and all I think is appropriate.)

Relevant QAA Computing Curriculum Sections:

[Please see http://www.qaa.ac.uk/en/Publications/Documents/Subject-benchmark-statement-Computing.aspx.pdf to check which section the course fits into.]

Software engineering

Graduate Attributes, Personal and Professional skills:

[This field should be used to describe the contribution made to the development of a student's personal and professional attributes and skills as a result of studying this course – i.e. the generic and transferable skills beyond the subject of study itself. Reference in particular should be made to SCQF learning characteristics at the correct level http://www.sqa.org.uk/files_ccc/SCQF-LevelDescriptors.pdf].

(That's a broken link and I don't know what it should point to.)

Reading List:

*[Provide a list of relevant readings. See also remarks under **3d**.]*

See under 3d.

Breakdown of Learning and Teaching Activities:

[Total number of lecture hours and tutorial hours, with hours for coursework assignments.]

[The breakdown of learning and teaching activities should only include contact hours with the students; everything else should be accounted for in the Directed Learning and Independent Learning hours.

The total being 10 x course credits. Assume 10 weeks of lectures slots and 10 weeks of tutorials, though not all of these need to be filled with actual contact hours. As a guideline, if a 10-pt course has 20 lecture slots in principle, around 15 of these should be filled with examinable material; the rest should be used for guest lectures, revision sessions, introductions to assignments, etc. Additional categories of learning and teaching activities are available, a full list can be found at:

http://www.euclid.ed.ac.uk/Staff/Support/User_Guides/CCAM/Teaching_Learning.htm]

Lecture Hours: 15 hours

Seminar/Tutorial Hours: 0 hours

Supervise practical/Workshop/Studio hours: 14 hours (NB for BoS: not including the summative assessment in week 6, timetabled in the same slot, to avoid double-counting)

Summative assessment hours: 4 hours

Feedback/Feedforward hours: 1 hour

Scheduled revision session: 1 hour

Directed Learning and Independent Learning hours: 165 hours

Total hours: 200 hours

You may also find the guidance on 'Total Contact Teaching Hours' and 'Examination & Assessment Information' at:
http://www.studentsystems.ed.ac.uk/Staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html

Keywords:

[A list of searchable keywords.]

software engineering

software design

software modelling

model-driven development

SECTION 3 - COURSE MATERIALS

3a. Sample exam question(s)

[Sample exam questions with model answers to the individual questions are required for new courses. A justification of the exam format should be provided where the suggested format non-standard. The online list of past exam papers gives an idea of what exam formats are most commonly used and which alternative formats have been http://www.inf.ed.ac.uk/teaching/exam_papers/.]

Exam questions to be set for the new course will be similar in style to those set for SEOC. For example, 2014 q3 would still be suitable for the new course.

I would like to modify the format of the exam from SEOC's "choose 2 questions out of 3" format to the (also widely used) "do q1 and then either q2 or q3" format. This would make it easier to ensure that all areas of the course are covered.

3b. Sample coursework specification

[Provide a description of a possible assignment with an estimate of effort against each sub-task and a description of marking criteria.]

The form of the Week 6 summative coursework, to be done in the standard two-hour lab slot, individually, open book, on locked-down machines provided with standard documentation but not internet access, would be something like:

Q1. A software team is designing a system to manage the production of volumes of papers. On discussion with the customer, they find that a paper may be a conference paper or a journal paper. Any paper consists of sections. Each section has a single author. A paper may have many authors, who will include at least the authors of all the sections. Papers are collected into volumes. The same paper may occur in several volumes, but any section occurs in exactly one paper.

Draw a class diagram to illustrate the situation, as known to the designers so far, according to the above text: do not change design decisions or invent new concepts, but comment (in a UML note on your diagram) on any aspect where you have to make an assumption. Your diagram should include at least classes Volume, Paper and Section, and at least three of: interface, aggregation, composition, generalization. Show multiplicities and navigabilities, where you can deduce them from the text. Do not show attributes or operations.

Q2. Study the provided body of Java code for the BORG Calendar. In the tool:

Draw a single-scenario sequence diagram to represent what happens when an object of class Memo receives message decrypt(``password''). Assume that in this scenario isEncrypted() returns true.

Draw a protocol state diagram for class Memo.

Draw a protocol state diagram for class ReminderPopup. Here you'll have a trickier job to decide what the states are, and you'll need to look at the superclasses as well as this class's own code.

Q3. Study the provided protocol state diagram and Java class. They are currently inconsistent: the Java code does not correctly implement the diagram. Leaving the diagram unchanged, modify the Java code so that they are consistent. Use comments to explain how you have resolved any issues you come across.

Effort: this coursework is done within the timetabled 2-hour slot.

Marking criteria: correctness of the diagrams and code produced, both syntactic and semantic. To be checked automatically to the extent possible, to enable prompt feedback.

3c. Sample tutorial/lab sheet questions

[Provide a list of tutorial questions and answers and/or samples of lab sheets.]

Early lab questions will be like those given above for coursework: e.g., the first lab will work on UML class diagrams and their relation to Java code, the second on sequence diagrams.

Later lab sessions might include questions like:

Study the given metamodel, models and texts (in this case, documentation stubs). Develop a model transformation that takes as input a model conforming to the metamodel, and

produces as output text appropriate to that model, following the given examples. What decisions do you have to take (a) about the functionality of your model transformation (b) about how it is structured?

3d. 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.]

Books

Likely not to prescribe a compulsory book, but instead to recommend several for reading around, including:

Timothy Lethbridge and Robert Langaniere. Object oriented software engineering: practical software development using UML and Java. McGraw Hill, 2002.

Perdita Stevens with Rob Pooley. Using UML. Pearson (second edition).

Tools

The tool ecosystem relevant to this course is fast-changing and I do not want to mention specific tools in the official course descriptor. I currently plan to use Papyrus, inside Eclipse, as the main modelling tool and ATL as the main model-transformation tool.

SECTION 4 - COURSE MANAGEMENT

4a. Course information and publicity

[Describe what information will be provided at the start of the academic year in which format, how and where the course will be advertised, what materials will be made available online and when they will be finalised. Please note that University and School policies require that all course information is available at the start of the academic year including all teaching materials and lecture slides.]

Introductory video. Course description, both official and extracted from this document. Material from SEOC, together with an explanation of how the new course differs. First set of lab exercises.

In the first year of presentation it will be important to be able to react appropriately to feedback from the student cohort (see below) so I would *not* plan to make all lecture slides available before the course starts. Rather, I will make teaching material (slides, lab exercises) available the week before it is used. In subsequent years one could provide the full set, though I still consider it important to be able to adapt the material when appropriate.

4b. Feedback

[Provide details on feedback arrangements for the course. This includes when and how course feedback is solicited from the class and responded to, what feedback will be provided on assessment (coursework and exams) within what timeframe, and what opportunities students will be given to respond to feedback.

The University is committed to a baseline of principles regarding feedback that we have to implement at every level, these are described at http://www.docs.sasg.ed.ac.uk/AcademicServices/Policies/Feedback_Standards_Guiding_Principles.pdf.

Further guidance is available from http://www.enhancingfeedback.ed.ac.uk/staff.html.]

Feedback from the students:

"Preassessment" questionnaire on prior knowledge and experience and expectations from the course

Interaction in labs and lectures.

Piazza group or similar.

Feedback to the students:

Exercises in the weekly lab will be supported by automated feedback where possible. The lab sessions will be run interactively with opportunities to compare answers with other students, raise questions about them, and get help from the TA and the lecturer.

Where appropriate the course will provide online quizzes with immediate feedback, as is done at present for parts of SEOC.

Students will be encouraged to interact on the Piazza or similar group, where the TA and lecturer will also be available for feedback on questions that need it.

Notes and explanatory videos will be provided on formative exercises, as appropriate, as is presently done for SEOC.

The summatively assessed coursework will be marked by a combination of automated and human feedback, provided as soon as may be and certainly within two weeks. The lecture after the feedback is available will be dedicated to discussion and matters arising.

As is standard in Informatics, individual feedback will not be provided on the written exam. The lecturer will produce brief notes on what was done well or badly, common errors and points to note.

4c. Management of teaching delivery

[Provide details on responsibilities of each course staff member, how the lecturer will recruit, train, and supervise other course staff, what forms of communication with the class will be used, how required equipment will be procured and maintained. Include information about what support will be required for this from other parties, e.g. colleagues or the Informatics Teaching Organisation.]

Lecturer: give lectures and lead labs.

TA: help write lab exercises. Attend labs to be a roving person helping students will difficulties.

Marker: help with marking of both summative and formative exercises.

TA, marker: recruitment and general training by standard mechanisms. I would expect to share the exercises and material they will be supporting with the TA and marker ahead of the students seeing it, and provide any necessary clarification. All direct interaction between the TA and the marker and the students will happen under the lecturer's direct supervision.

Equipment:

DICE lab with one machine per student, all having line of sight to a projection screen. The software requirements will need to be negotiated with computing officers.

Video recording of lectures (but not labs) would be helpful: this should be straightforward to arrange if, as is likely, the course is timetabled to a room where that is available.

SECTION 5 - COMMENTS

[This section summarises comments received from relevant individuals prior to proposing the course. If you have not discussed this proposal with others please note this].

Discussed in outline with 2014-15 SEOC cohort. Comments very positive, students saying that the extra hands-on material would have helped them to understand the design material more easily and have been interesting. Most students who took the 10pt SEOC course said they would have taken the 20pt replacement.

I also solicited comments on this proposal from this year's SEOC cohort. Timing (I only finished the proposal shortly before the BoS deadline) meant that at the time of writing I have comments from only one student, but they are positive:

"I like the fact that it's 20 credits. I'm not a fan of taking 6 distinct courses this semester, especially when I have to keep all the material fresh in our minds until summer for the exams.

I also like the fact that there is coursework which counts towards your final mark - I know not all students would agree, but I'm a fan of high coursework weights (Compiling Techniques and CSLP are 100% coursework). This year is particularly difficult for me in that 4 of my semester 1 courses don't have exams until summer, meaning there will be a lot more pressure during revision period."

5a. 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.]

(only just sent to year organisers: sorry, I had overlooked this section)

5b. BoS Academic Secretary

[Any proposal has to 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.]