UNIVERSITY of York

Examinations, 2017–2018

DEPARTMENT OF COMPUTER SCIENCE

Software Engineering Project (SEPR)

Group Open Assessment

Issued: Aut/2/Wed, 4 October 2017

Submission due: 12 noon,

Assessment 1: Aut/7/Wednesday, 8 November 2017 [25%]
Assessment 2: Spr/3/Monday, 22 January 2018 [25%]
Assessment 3: Spr/7/Monday, 19 February 2018 [20%]
Assessment 4: Sum/3/Wednesday, 2 May 2018 [20%] *including*
Assessed Presentation: Sum/4/TBA

Feedback and marks due:

Assessment 1: Aut/10/Wed, 20 December 2017
Assessment 2: Spr/7/Mon, 19 February 2018
Assessment 3: Spr/10/Mon, 2 April 2018
Assessment 4: Sum/7/Wed, 30 May 2018

These are department-set dates. Feedback is released as soon as it is ready, so that it can be taken into account in subsequent deliverables. Both SEPR groups have the same questions and deliverables; only the scenario differs.

All groups should submit their answers through the electronic submission system (http://www.cs.york.ac.uk/student/assessment/submit/) by 12 noon on the submission date (above). An assessment (or part of an assessment) that is submitted after this deadline will be marked as if it had been handed in on time, but the Board of Examiners will normally apply a lateness penalty to the whole assessment.

The feedback and mark dates are guided by departmental policy, but, in exceptional cases, there may be a delay. In these cases, all students expecting feedback will be emailed by the module owner with a revised feedback date. The date that students can expect to see their feedback is published on the module descriptor: http://www.cs.york.ac.uk/modules/

Your attention is drawn to the section about Academic Misconduct in your Departmental Handbook: https://www.cs.york.ac.uk/student/handbook/

All queries on these assessments should be addressed to ***by email*, to dimitris.kolovos@york.ac.uk and richard.paige@york.ac.uk**. Answers that apply to all students will be posted on the module VLE.

Rubric:

- The four team assessments account for 90% of the total module mark. The final 10% is for the individual closed examination in the summer common assessment period.
  - The closed examination requires you to critique some aspect or aspects of your (team's) approach, showing appropriate insight into software engineering, its methods and best practice.
- All team deliverables must start with the team's name (this appears on the team lists). The examination numbers of team members must **not** appear anywhere in the assessment deliverables.
- Answer all questions. Note the page limits for each question. Parts of answers that go beyond the page limit will not be marked. References must be listed at the end of each deliverable, and do not count towards page limits.

1 SEPR Assessment Structure

This is a **very long** assessment document.

- Read it all, together, as a team: we have not put in anything that is unimportant.
- It is long because the SEPR assessment forms part of your software engineering training:
  - the things requested, the styles of presentation, and the feedback on the assessments, are designed to help you develop your experience and understanding of team-based software engineering;
  - nothing is there by mistake (we hope!).

In 2017-18, the SEPR cohort is split into two **groups**. Each group has a number of **teams**. The groups create different products: the product brief will be issued to each team at the relevant team-forming practical for that group, in Autumn week 2, and will then appear on the SEPR VLE site.

The **assessed presentation** for Assessment 4 will take place in the summer term: the details will be published on the SEPR VLE page and notified by email to all students.

The team assessments build on each other. Together, the assessments constitute a complete software engineering project. There have been light revisions from the 2016-17 assessment.

1.1 Time

Do not underestimate the individual and team time required for SEPR.

- The module is 30 credits, which is 300 student hours – each.
- We expect that all teams will spend more than 1500 person-hours on the teamwork.
- Work to the set assessments: do not get carried away with, e.g., coding!
- Work out what needs to be researched and undertaken at the start of the project, and revisit it at the start of every assessment. The best results come in teams that are very strategic in their approach to the overall project.
- Talk to the lecturers whenever you need help or reassurance!

In this document, Section 2 outlines team set-up and team-mark arrangements; Section 3 describes the four team assessments, and outlines the deliverables required for each task; and Section 4 gives guidance on the style and marking of each sort of deliverable, as well as team issues, and various forms of reassessment.

The assessment refers to the “scenario”: this is the product brief that is issued at the team-forming practical in Autumn week 2, and is subsequently available on the SEPR VLE site.

2 Project Teams and Team Marks

The project is designed to be undertaken by teams of 5, 6 or 7 students.

- It is up to each team to develop its own structures, team leadership and team arrangements. There is material provided, but teams should research team working (there are some notes from previous teams on the VLE site), and consult the module lecturers as soon as team-related problems arise.
- By analogy to project teams in some software development companies, the lead lecturers (Dr Dimitris Kolovos, Prof. Richard Paige) form a higher management level, and should be consulted for things beyond the control of the team, for instance where team arrangements do not work effectively, or workload becomes unbalanced and the team cannot reach an equitable adjustment.
- The lead lecturers (Dr Dimitris Kolovos, Prof. Richard Paige) are the final arbiters in any dispute arising from the composition, performance or size of teams, and in the distribution of marks to members of a team.

2.1 Team marks and self-assessment

Each assessment is marked out of 100. The first two team assessments are weighted at 25% of the module mark, since this is where the foundational work and research into software engineering takes place. The second two team assessments are weighted at 20% of the module mark, since there is much less new learning in these phases – the concentration is on extension and revision of existing software products.

The mark awarded to each student is the sum of the team mark, the self-assessment mark, any applicable bonus marks, and any individual penalty (section 2.2). This section explains the self-assessment mark.

- With each assessment’s deliverables, a team may include a self-assessment table, laid out as shown in Table 1. The **`Self-assessment Mark`** for each member of the team must be an **integer** in the range 0 to 10 (i.e. $\{0, 1, 2, ..., 10\}$).

Table 1: Self-assessment Table

Team Name:		**Assessment:**
NAME	**SIGNED**	**Self-assessment Mark**
...	...	...
...	...	...
e.g. D. Duck	*[a signature]*	2

Use the self-assessment if the team wants to recognise unequal contribution to the team work and/or the assessment deliverables. **Note that individuals or teams are strongly encouraged to contact one of the lecturers, as soon as a problem becomes apparent.**

- If a team allocates a low mark to one or more team members, and has not already discussed team issues with the lecturers, it is likely that the team will be contacted by the lecturers. This is important, as both a check of the fairness of marking, and, more importantly, a prompt to discuss and hopefully mitigate team problems that could affect subsequent assessments.

If no self-assessment table is included, or if a submitted table is incomplete, then the self-assessment mark for each member of the team is normally 10.

- If you cannot resolve the team self-assessment within your team, you should consult the lecturers, **immediately after** submission.

2.2 Individual Penalties

It is important that **all** problems with participation in team work and assessment activities are discussed in a timely manner with the management (as above). ***Problems can be raised at any time***, not only during an assessment activity.

Whilst the module lecturers can help, they cannot force equal participation. So, a student who does not participate adequately in team work and assessment activities, for whatever reason, or who is awarded a low self-assessment mark for an assessment, or who otherwise misses a substantial part of SEPR, is likely to have marks deducted from one or more team assessments.

Individual penalties are applied by the lead lecturers (Dr Dimitris Kolovos, Prof. Richard Paige). It is normal that penalties are only applied after discussion with the individual concerned. Total non-participation results in **a zero mark** (a zero assessment mark, no team bonus marks and no self-assessment mark) for the individual student, for all affected team assessments.

3 The Team Assessments

Each team assessment addresses specific software engineering aspects of the game described your group's scenario (product brief).

Teams must bear their own costs (if any), and are responsible for all aspects of management of their own work.

Where deliverables of later assessments build on deliverables of earlier assessments, teams should reuse and extend previous deliverables, including (in Assessments 3 and 4) all material provided with a selected software product. For example, the user manual can reuse or extend either your team's existing manual or the manual you inherited, as appropriate.

Where a deliverable asks for an updated version of an earlier deliverable, teams will be marked on the updates – it is essential that changes are clearly identified.

- Section 4 gives further advice on the content and assessment of each type of deliverable, and complements the requirements for each assessment, below.
- Note that the SEPR lectures and recorded tutorials give a general introduction to the software engineering required for the teamwork; teams are expected to research, and to develop their skills through their research, and through reflection on feedback given in practicals and on assessments.

3.1 KISS

If you do not know what this means, find out!

This project requires a non-trivial amount of research, design and implementation effort. A key principle of software engineering is that products should meet their requirements. There is no merit in producing something that goes beyond what is required. You are STRONGLY DISCOURAGED from adding unjustifiable additional features.

3.2 Electronic Submission

Your submission for each team assessment must be one zipfile. The details of what the zipfile should contain for each assessment are given in tables 2 to 5, below. The zipfile must be named using the team name, *myteam*.zip.

- The electronic submission system is configured so that any of you can submit a SEPR team assessment on behalf of your team.

3.3 Assessment 1: Requirements specification and design, due: noon, Aut/7/Wed

Assessment 1 covers the early stages of requirements specification and design – including the conceptual architecture of your proposed solution; method selection, planning; and risk assessment and mitigation. As part of the assessment, each team will need to (at least) identify appropriate software engineering methods and techniques, set up a regular meeting schedule, and allocate tasks to team members according to team member availability and skills.

3.3.1 Deliverables for Assessment 1

Your team will submit a website plus a single .zip file. The requirements for the deliverables to be included in the zipfile are summarised in Table 2.

Table 2: Assessment 1 zipfile contents

	Deliverable	**Max. mark**	**Page limit**	**File name and format**
1.	website (submit only the URL)	3	—	url1.txt
2.	Requirements	22	$1 + 3$	Req1.pdf
3.	Architecture	25	$3 + 2$	Arch1.pdf
4.	Method selection and planning	20	$2 + 1 + 1$	Plan1.pdf
5.	Risk assessment and mitigation	20	$1 + 3$	Risk1.pdf
6.	Optional self-assessment table	10	—	SelfAss1.pdf

1. Website [3 marks]
   a) The submitted URL must link to the website that is the “public face” of your team’s project, and will be updated as you proceed.
   b) The “management” and other teams can use the website at any time during the project to access all versions of documentation and executables.
   c) You must link each of the assessment documents to your website in a clear and accessible way.
   d) In this assessment, it is the website structure that is marked. You will be penalised if material is not easily locatable and accessible.

2. Requirements [22 marks]:

   a) Write a succinct introduction explaining how requirements were elicited and negotiated, and why they are presented as they are. Your submission should evidence research into requirements specification and presentation (5 marks, $\leq$ 1 *page*).

   b) Give a systematic and appropriately-formatted statement of requirements, including, for each requirement, a note of any relevant environmental assumptions, associated risks, or alternatives (17 marks, $\leq$ 3 *pages*).

   Note that you will need a requirements referencing system (e.g. numbering), and may need to update this for subsequent assessment deliverables.

3. Architecture [25 marks]:

   a) Give an abstract (i.e. conceptual – **not the fine detail**) representation of the proposed architecture (structure) of the team's software, with a brief statement of the specific languages used to describe the architecture (for instance, relevant parts or versions of the UML family), and, if appropriate, the tool(s) used to create the architecture representation (15 marks, $\leq$ 3 *pages*).

   b) Give a systematic justification for the architecture, explaining and justifying the components and overall structure, and noting any non-standard notations used (10 marks, $\leq$ 2 *pages*).

4. Method selection and planning [20 marks]:

   a) Give an outline and justification of the team's proposed software engineering methods, and identify any development or collaboration tools that the team plans to use to support the project or the team working (10 marks, $\leq$ 2 *pages*).

   b) Outline the team's approach to team organisation, and explain why the chosen approach is appropriate for both the team and the project (5 marks, $\leq$ 1 *page*).

   c) Give a systematic plan for the rest of the SEPR project. Your plan should lay out the key tasks for assessments 2 to 4. You should provide a detailed plan for the the software engineering tasks required for assessment 2. Tasks should indicate the earliest starting date and the latest finishing date, as well as task priorities. The plan should also identify a critical path and task dependencies (5 marks, $\leq$ 1 *page*).

5. Risk assessment and mitigation: [20 marks]
   a) Introduce and justify your risk format and level of detail (5 marks, $\leq\ 1\ page$).
   b) Give a systematic tabular presentation of risks to the SEPR project, their likelihood, impact, and mitigation (15 marks, $\leq\ 3\ pages$).

   SEPR is a small project, developing non-critical software. Keep your likelihood and impact measures simple.

See Section 2 for information on the self-assessment table. [10 marks]
You may use additional pages for a bibliography. Any other overrun will be penalised.

3.4 Assessment 2: Preliminary Implementation, due: noon, Spr/3/Monday

Assessment 2 concerns the detailed software engineering design, and implementation of part of the software system, based on the requirements and specification created for Assessment 1. You will revisit the Assessment 1 deliverables, review and maintain non-code deliverables, and apply version control.

Your group's scenario (product brief) indicates which elements are to be designed and implemented for Assessment 2. In Assessment 2, you must not exceed this brief: do not implement any other features.

Your product must be consistent with the scenario, and with your Assessment 1 work, as described below.

3.4.1 Deliverables for Assessment 2:

Your team will submit a website plus a single .zip file. The requirements for the deliverables to be included in the zipfile are summarised in Table 3.

Table 3: Assessment 2 zipfile contents

	Deliverable	**Max. mark**	**Page limit**	**File name and format**
1.	website (submit URL only)	3	—	url2. txt
2.	Architecture report	22	2 + 2	Arch2.pdf
3.	Implementation and report	20	code + 1	see text Impl2.pdf
4.	GUI report	5	1	GUI2.pdf
5.	Testing report	20	1 + 2 (+ website)	Test2.pdf
6.	Updates on assessment 1 deliverables	20	1 + 1 + 1 + web updates	Updates2.pdf
7.	Optional self-assessment table	10	—	SelfAss2.pdf

1. The updated Website for your team's project [3 marks].

   The submitted URL should now link to:

   a) all the project-related Assessment 2 deliverables, as well as the Assessment 1 versions (please note the specific requirements for updates, below);

   b) the executable for the game so far;
   c) your executable test plan and testing results for the game so far;
   d) the user manual for the game so far (including editable source as well as e.g. a pdf manual).

2. Architecture report [22 marks]:
   a) Give a concrete architecture showing the structure of the team's code, with a brief statement of the specific languages used to describe the architecture, and, if appropriate, the tool(s) used to create the concrete architecture representation. (12 marks, ≤ 2 *page*)
   b) Give a systematic justification for the concrete architecture, explaining how the concrete architecture builds from the abstract architecture – and any changes that had to be made to the abstract architecture. Relate the concrete architecture clearly to the requirements, using your requirements referencing for identification, and consistent naming of constructs to provide traceability. Provide precise URLs to any relevant web pages. (10 marks, ≤ 2 *pages*)

3. Implementation [20 marks]:
   a) Provide documented code, and (in a header comment in the code) the precise URL of the executable on the team website, for a working implementation of the part of the game that meets the remit, requirements and concrete architecture for Assessment 2. Code can be submitted in the zipfile, or via a link to a repository with a verifiable date before the hand-in deadline. (15 marks)
   b) State explicitly any of the features required for Assessment 2 that are not (fully) implemented, using your requirements referencing for identification, and consistent naming of constructs to provide traceability. Provide precise URLs to any relevant web pages. (5 marks, ≤ 1 *pages*)

4. Provide a GUI report [5 marks, ≤ 1 *pages*] that briefly summarise and justifies the initial GUI design, with reference to requirements, usability and playability. Provide precise URLs to any relevant web pages.

5. A software testing report [20 marks]:
   a) Briefly summarise your testing method(s) and approach(es), explaining why these are appropriate for the project so far. (5 marks, ≤ 1 *page*)
   b) Give a brief report on the actual tests, including statistics of what tests were run and what results were achieved, with a clear statement of any

tests that are failed by the current implementation. If some tests failed, explain why these do not or cannot be passed and comment on what is needed to enable all tests to be passed. If no tests failed, comment on the completeness and correctness of your tests instead. (5 marks, ≤ 2 *pages*)

c) Provide the precise URLs for the testing material on the website: this material should comprise your testing design and evidence of testing, and is marked here (10 marks).

6. Updates on assessment 1 deliverables [20 marks]

In line with software engineering best practice, you are expected to maintain and update your assessment 1 deliverables, such that the up-to-date versions can be easily located on the team's website. For assessment 2, the deliverable itself only requires summaries and the precise URLs, as follows:

a) Updated requirements [6 marks]:

i. *On the Website*, include the updated statement of requirements, highlighting what has been added and what has been changed (1 mark).

ii. *In your deliverable*, include a brief explanation and justification of any requirements changes made for Assessment 2. Include the precise URL of the statement of updated requirements (≤ 1 *page*, 5 marks).

Please ensure that the original and any revised requirements referencing is clear, to support tracing of changes.

b) Methods, plans update [7 marks]:

i. *On the website*, include the updated plan, showing remaining key tasks, and detailed tasks for assessment 3. With the plan, include a brief explanation of any major changes. Also, include any updated documentation components — e.g. those originally provided under "Method selection and planning" in Assessment 1. (2 marks).

ii. *In the deliverable*, include a brief summary of any changes made in relation to the use of methods and tools, with a short explanation and justification of each. Give the precise URLs of the updated plan and any relevant methods and tools web pages (5 marks, ≤ 1 *page*).

Note that you will be marked on the changes, explanations and justifications: please ensure that the changes are clearly highlighted.

c) Risk assessment and mitigation update [7 marks]:

i. *On the website*, include the updated risk assessment and mitigation, highlighting any changes that have been made. If you decided to change your approach to risk assessment and mitigation after Assessment 1, please state clearly that the web page content is new, and link to the Assessment 1 risk assessment to allow comparison (2 marks).

ii. *In the deliverable*, include a brief explanation and justification for any changes made to the risk assessment for the whole project; if you changed your approach, explain and justify your change. If no changes are made, explain why the risk assessment and mitigation, and the approach, are still appropriate. (5 marks, $\leq 1\ page$)

Note that you will be marked on the changes, explanations and justifications: please ensure that the changes are clearly highlighted.

See Section 2 for information on the self-assessment table. [10 marks]
You may use additional pages for a bibliography. Any other overrun will be penalised.

3.5 Assessment 3: Selection, Extension and Integration, due: noon, Spr/7/Monday

Assessment 3 requires each team to work on another team's product. There are thus two phases to the assessment, below. As in the previous assessment, Assessment 3 requires review, maintenance and version control activities.

3.5.1 Assessment 3, Phase 1: selection: completed Spr/3/Friday at 10am

After the submission of Assessment 2 (Spr/3/Monday, above), each team has a short period to consider the products of the other teams. In the **Spring Week 3 practical** class that your team attends (Spr/3/Tuesday or Spr/3/Wednesday), each team will present its product, and can then discuss with other teams. On Spr/3/Friday, each team is required to register, by 10am, its choice of product to work on in Assessment 3, **by emailing richard.paige@york.ac.uk**.

The only constraint on selection is that a team is not allowed to use its own Assessment 2 software in Assessment 3.

In selecting a software product, criteria that may be considered include: (1) the overall quality of the software product; (2) estimates of effort remaining to complete the implementation; (3) clarity and quality of the requirements specification, architecture, testing and implementation.

A bonus of 3 marks will be added to the team's Assessment 2 mark for each registered selection of the team's software product, with the condition that no individual can attain more than 100% for Assessment 2.

3.5.2 Assessment 3, Phase 2: extension and integration

Your selected product should have designed and implemented the initial elements of the game, as indicated in your group's scenario. For Assessment 3, the selected software product must be extended to cover the full product brief.

For this assessment, you are allowed, but not required, to add other features or behaviours, as desired, but these must be fully explained and justified in the deliverables, and must not violate the requirements for the game.

3.5.3 Deliverables for Assessment 3

Your team will submit a website plus a single .zip file. The requirements for the deliverables to be included in the zipfile are summarised in Table 4.

1. The updated Website for your team's project [5 marks].

   The submitted URL should now link to:

Table 4: Assessment 3 zipfile contents

	Deliverable	Max. mark	Page limit	File name and format
1.	website (submit only the URL)	5	—	url3.txt
2.	Change report	30	1 + 4	Change3.pdf
3.	Implementation and report	55	code + 4	see text Impl3.pdf
4.	Optional self-assessment table	10	—	SelfAss3.pdf

a) all the project-related Assessment 3 deliverables, as well as the Assessment 1 and 2 versions (please note the specific requirements for updates, below);

b) the executable for the game;

c) your executable test plan and testing results;

d) the user manual (including editable source as well as e.g., a pdf manual).

2. Change Report [30 marks]:

   a) Briefly summarise the team's formal approach(es) to change management, including change management of all deliverables, documentation and code. (5 marks, $\leq$ 1 *page*)

   b) For each of the following four items, include a brief explanation and justification of any changes made (other than simple extensions made to complete the product). Include the precise URLs of the web pages where updated material is located. If these is no change to report, please state and justify why no change was necessary.
   (Maximum 7 marks per item, to a maximum total of 25 marks; $\leq$ 4 *pages* in total)

      i. The GUI report.

      ii. The testing report: methods and approaches, materials or presentation of tests and testing statistics (you must provide the precise URLs for Assessment 3 testing materials, as specified in Assessment 2).

      iii. Methods and plans: software development methods and tools; team management approaches; plan for Assessment 4 (you must include the precise URL of the updated plan).

      iv. Risk assessment and mitigation: approach, presentation, risks, mitigations.

3. Implementation [55 marks]:
   a) Provide documented code and (in a header comment in the code) the precise URL of the executable on the team website, for a working implementation of the game that meets the remit, requirements and concrete architecture for Assessment 3. You code comments should highlight new or extended sections of code, and should be consistent with your change report. Code can be submitted in the zipfile, or via a link to a repository with a verifiable date before the hand-in deadline. (25 marks)
   b) Explain how your code implements your architecture and requirements (incorporating your recorded changes for Assessment 3). Briefly explain any significant new features, e.g. non-primitive data types, significant algorithms or data structures. Give a systematic report of any significant changes made to the previous software, clearly justifying each change, and relating it to the requirements and architectures. Note that, if a change has significant side effects, it needs a solid software engineering justification. State explicitly any of the features required for Assessment 3 that are not (fully) implemented. (30 marks, $\leq 4\ pages$)

See Section 2 for information on the self-assessment table. [10 marks]
You may use additional pages for a bibliography. Any other overrun will be penalised.

3.6 Assessment 4: Selection, Requirements Change, due: noon, Sum/3/Wed

Assessment 4 requires you to work on a product other than that which the team worked on in Assessment 3. After product selection, changes will be issued. There are thus two preliminary phases before the main phase of Assessment 4. For Assessment 4, there is also an assessed presentation which will take place in Summer week 4.

3.6.1 Assessment 4: Phase 1: selection: completed Spr/7/Friday at 10am

After the submission of Assessment 3 (Spr/7/Monday, above), each team has a short period to consider the products of the other teams. In the **Spring Week 7 practical** class that your team attends (Spr/7/Tuesday or Spr/7/Wednesday), each team will present its product, and can then discuss with other teams. On Spr/7/Friday, each team is required to register, by 10am, their choice of product to work on in Assessment 4, by **emailing richard.paige@york.ac.uk**.

The only constraint on selection is that a team is not allowed to use its own Assessment 3 software for Assessment 4. A team is allowed to use another team's Assessment 3 extension of its original Assessment 2 product, but must work from the Assessment 3 version.

In selecting a software product, criteria that may be considered include: (1) the overall quality of the software product; (2) estimates of effort remaining to make the required changes to the implementation; (3) clarity and quality of the requirements specification, architecture, testing and implementation.

A bonus of 3 marks will be added to the team's Assessment 3 mark for each registered selection of the team's software product for Assessment 4, with the condition that no individual can attain more than 100% for Assessment 3.

3.6.2 Assessment 4: Phase 2: requirements change: Spr/7/Friday

For each group, a small set of changes will be issued on Spr/7/Friday (after all teams have registered their choice of product); these may include changes to the scenario and/or introduction of new requirements.

Using your selected software product, the team must modify the software product and supporting documentation to address these changes. Teams must also produce an up-to-date website for the game.

3.6.3 Assessment 4: Assessed Presentation [20 marks]

Each team is required to make a presentation of the finished game as if to an external client. The client, who will be an experienced software engineer or games developer, will question the teams about their games and may ask to see the accompanying marketing and technical information on the website.

The presentation should assume that the client is interested in buying or marketing the product, and is thus aware of the requirements specified for the product (including the changes in Assessment 4). Whilst the client will be interested, in general, in major design decisions, the quality of the software, and the playability of the game, s/he is likely to be also interested in the potential market for, and extensibility of, the product.

The presentation will take place in a meeting room with access to the University computer network: within this constraint, teams can use any appropriate media, and any combination of team members may take part. The presentation should take at most **5 minutes**.

Each team will be marked on the clarity and appropriateness of its presentation and its interaction with the client. Marking is independent of the client's award (the prize!) for what s/he considers the best product.

3.6.4 Other Deliverables for Assessment 4 [80 marks]

Your team will submit a website plus a single .zip file. The requirements for the deliverables to be included in the zipfile are summarised in Table 5.

Table 5: Assessment 4 zipfile contents

	Deliverable	**Max. mark**	**Page limit**	**File name and format**
1.	Final Architecture and Traceability Report	15	3	AT4.pdf
2.	Evaluation and Testing report	20	3 + 1	ET4.pdf
3.	Implementation	15	code +	see text
	and report		2	Impl4.pdf
4.	Project review report	20	1 + 1	Review4.pdf
5.	Optional self-assessment table	10	—	SelfAss4.pdf

This final assessment presents an opportunity to (a) pull together the final forms of all the software deliverables, and (b) to reflect on the team management aspects of the project. You should take account of all the feedback from previous assessments, as well as reflecting the team's experiences.

1. Final Architecture and Traceability report [15 marks]:
   Briefly present, with appropriate models, explanations and justifications, the final architecture of your completed system. Note both how and why you changed the

design that you inherited and how you accommodated the changes that were introduced for Assessment 4. You may include precise URLs to any relevant material on the website (e.g. previous versions of architecture, previous justifications, tables of requirements, etc.). (15 marks, ≤ 3 *pages*)

2. Evaluation and Testing report [20 marks]:

   a) Briefly explain and justify the approach that the team took to evaluation and testing of the final product. For the purposes of this question, *evaluation* refers to how you determined that the product met its brief; *testing* refers to how you determined that your code was of appropriate quality – you need to state and justify what you consider to be appropriate quality. Note any modification that you made to the Assessment 3 testing approach or materials, and how you accommodated the changes that were introduced for Assessment 4. You may include precise URLs to any relevant material on the website (e.g. test plans, testing materials, test statistics, previous versions of testing, previous justifications, etc.). (15 marks, ≤ 3 *pages*)

   b) Comment on how your product meets, and does not meet, the requirements. Include precise URLs for the web pages where the final requirements can be found. (5 marks, ≤ 1 *page*)

3. Implementation [15 marks]:

   a) Provide documented code and (in a header comment in the code) the URL of the executable on the team website, for a working implementation of the game that meets the remit, requirements for Assessment 3, and the concrete architecture and changes introduced in Assessment 4. You code comments should highlight new or extended sections of code. Code can be submitted in the zipfile, or via a link to a repository with a verifiable date before the hand-in deadline. (5 marks)

   b) Summarise how the *software (code and GUI)* was modified to incorporate the required changes, and any other necessary changes made to the software, relating each change to the revised final requirements and architecture, and clearly justifying each change. Explain how and why you changed the software and GUI that you inherited. Explain and justify any extra features that have been included in the software. (10 marks, ≤ 2 *pages*)

4. Project Review Report [20 marks]:

   Now that you have completed an end-to-end software project, including extension and modification, write a short commentary on your team's management, approaches, methods and tools, as follows.

a) Briefly summarise, with appropriate citations of literature and online sources, your team's approach to team management and the team structure at the end of the project. Give a succinct account of how and why your team management and team structure evolved (or did not evolve) over the course of the project. Make reference to the changing needs or risks of the project, as well as your developing understanding of the members of the team and of software engineering and team management. (10 marks, ≤ 1 *page*)

b) Briefly summarise, with appropriate citations of literature and online sources, the software engineering *development* methods and tools that your team chose to use. Give a succinct account of how and why the choice of *development* methods and tools evolved (or did not evolve) over the course of the project. Make reference to the changing needs or risks of the project, as well as your developing understanding of the members of the team and of software engineering. (10 marks, ≤ 1 *page*)

See Section 2 for information on the self-assessment table. [10 marks]
You may use additional pages for a bibliography. Any other overrun will be penalised.

4 Marking notes

To pass SEPR, a student must reach the overall module pass mark. There is no requirement to pass any one of the component assessments.

A student wishing to request exceptional circumstances affecting assessment (ECA) for one or more of the SEPR team assessments is strongly advised to discuss their situation with at least one of the lead lecturers, as there are legitimate ways to handle ECAs that affect team working, some of which avoid formal reassessment. See Section 4.13 for more information.

4.1 General Approach and Style

Teams are expected to *research* their software engineering needs and develop their experience and expertise in relation to team management, software engineering methods and tools, risk management, requirements, design/architecture, change management and version control, coding and GUI design.

- There is a wide range of material on line, and in standard texts such as Sommerville's *Software Engineering* (any recent edition is good: it's also on line).
- Credit will be given for evidence of research and for appropriate consideration of alternatives where explanation or justification is requested.
- The best results are likely to come from regular review and updating of your approaches to the project, rather than leaving these activities til the point when you write each report.
- Good software engineering is not a set of independent activities and documents: you should ensure that your documentation and code are internally and mutually consistent.
- You will **never** be penalised for asking for clarification.

We are looking for *clear, succinct presentation*, not essays. The SEPR lecturers have to assess, mark and provide feedback on a significant amount of material in a short time (during which they have many other obligations to fulfil!): the easier it is to read your reports, the better chance you have of a good mark.

- As a team, try to understand the markers' position, and work out a clear consistent presentation style.
- You must clearly format your reports so that they match the questions and sections of questions: it must be unambiguous what part of the assessment you think you are addressing in every part of every report.

- You will lose marks for ignoring instructions or for doing what you think we might want rather than what the question asks for.
- Take out superfluous opinions and adjectives: they have no part in an engineering report.
- Use bullet lists, or text bullets (as used in this document), and signal topics clearly, e.g. by subheadings or emboldened lead-words.
- Page limits are limits not targets: if you do not need all the pages, you will **not** be penalised for writing less, well. However, you are likely to lose credit for wordy, rambling, imprecise, or poorly-presented work.
- Spell check AND proof read all documents before submission.

When submitting your assessments, please ensure that all the deliverables are in accordance with the instructions for that element of assessment (Section 3), including zipfile information on naming.

4.2 Website and website deliverables

The website is used to provide a working software-engineering resource for you, other teams, and the markers, giving access to documents (e.g. requirements, architecture, testing materials, etc.) and executables.

- You will be marked on the structure of your website: how easy it is to locate required content, and how you manage the need to present "marketing" and software engineering content.
- Markers will **not** spend time hunting for material on the website, so you need a good site structure, and the route to specific software-engineering resources (including all required product materials) must be clear.

Please note that, if you cannot stay within the assessment page limits for "factual" deliverables such as the requirements or risk assessment, diagrammatic models, test evidence, etc., it is often appropriate to include the full material (suitably introduced, linked and formatted) on the website, and present an appropriately-chosen subset (with explanation and cross-reference links for the full material) in the assessment deliverable.

4.3 Requirements documentation

- You will be marked on the clarity and appropriateness of your approach, and appropriateness and presentation of requirements, **not** the number of requirements that you state.

- Good software engineering pays attention to traceability: (a) from requirements through architecture and design, to code; and (b) across the development lifespan. Marking will reflect how well your requirements presentation supports traceability, e.g. through its approach to requirement identification.
- Your requirements are assessed on their objectivity: is there a test that can demonstrate whether the requirement is met? For requirements that are inherently difficult to make objective, you may get credit for explaining how, subjectively, a requirement can be shown to be met.
- You will gain credit for concise, appropriate commentary on, for instance, how particular requirements could be affected by environmental assumptions, how you address any development or project risks related to specific requirements, and realistic alternatives for "risky" requirements.

4.4 Architecture documentation

You are asked variously for abstract (conceptual) and concrete architectures.

- You will be marked on how well your architecture and reporting conforms to the appropriate level of abstraction: you will lose marks for making the architecture too detailed, or ignoring instructions.
- You will be assessed on the clarity and appropriateness with which you state the language(s) and tools that you use.
- You will be marked on how clearly your architecture justification follows the structure of your architecture, and accounts for unusual features or notations.

4.5 Implementation and GUI

- You will be marked on the software engineering quality of your code, **not** its cleverness.
- When summarising design decisions, you should identify and focus on the key features and major decisions, rather than enumerating every data type, etc.
- Use formatting, naming conventions, etc. to make it easy to trace between your code and all relevant documentation.

The project is primarily about software engineering of the game: there is no teaching on GUI design, and an appropriately *small amount of credit* is available for your GUI.

- It is up to each team to decide how much effort they put into the visuals of the game: this may attract other teams (and the presentation client) to your product, may help to meet requirements, etc., but you do not need to put a lot of work into the GUI write-up!

4.6 Software testing

Please read the sections on software testing carefully. The bulk of the testing design and results should be on your website.

- The testing report is only part of the assessment: do only what is requested.
- For the material on the website, we are looking for clarity of test design and purpose, as well as evidence of actual testing. Please remember that the markers will **not** hunt for testing material: it needs to be easy to find on the website (e.g. because the URL in the report takes the user straight to the testing section of the website, etc.) and easy to understand.
- Even if there is no requirement to report on testing, your website must evidence that you have appropriately tested your software!
- You need to ensure that your test planning and execution is consistent with the whole software engineering product, not just the code.
- A software engineering rule of thumb is that testing needs to be related to the criticality (however measured) of what is being tested: markers will be looking for appropriate (and justified) levels and scales of testing.

4.7 Methods and plans, and their updating

- Make sure that you understand what is meant by the words used in the questions: terminology is used in its software engineering sense, not general English usage.
- Only answer the questions asked: you will not get credit for exploring other aspects of your team work in your answers.
- When you are asked to update or replace a report, you will gain credit for doing what is asked – you will lose credit, for instance, if it is unclear what you changed.
- You will be marked on the *appropriateness* of the contents and granularity of your plans: there is no point in going into huge detail for things that are a long way ahead. There are no marks for planning the past (although you need a plan for assessment 1, you are not asked to submit it, because it is history by the time you submit assessment 1!).

4.8 Risk assessment and mitigation, and its updating

A good risk assessment and mitigation aims to identify things that are risky and to mitigate effects, **not** to try to prevent occurrence of risks.

- Follow the guidance for Methods and plans, above.
- Research risk assessment and mitigation: you will get credit for the appropriateness of your approach and presentation to the type of project.
- You will be marked on the clarity and appropriateness of your approach, and appropriateness and presentation of risks and mitigation, **not** the number of risks that you state.

4.9 Change reporting

Change management, and software maintenance, are key activities in practical software engineering. You need to research these, and to plan for change even at the start of your project.

- Try not to report the same changes in different deliverables: you can cross-refer between reports as needed (give URLs if reports are not in the current deliverables).
- You should focus on discussing and justifying major changes, rather than listing minor (uncontentious, no side-effects) changes.
- You will gain credit for the consistency with which large changes are tracked and traceable through your project documentation (including any relevant risk factors, requirements, testing, etc.).

4.10 Project Review Report

The project review report is the only document that we ask for which is not directly part of good software engineering practice. However, the process of continuous self-analysis and improvement is part of mature software engineering (e.g. the CMM standards – look this up!).

If a book or website raves about a method (or tool, etc.), it does not mean that it will be ideal for your team and your project; your ability to analyse and understand this is an important part of software engineering management.

The review report is the hardest part of the team assessment. The best results are likely to be obtained if you prepare for this report from the start of the team project.

- We are looking for teams' abilities to analyse the successes and failures of their experience in SEPR – and to relate these to the teaching and to the team's own research.
- We are looking for rationale and evidence for change over time, whether due to changes in understanding, changes in the project, changes within or beyond your control – credit will be gained for evidencing how the evolution of the team and its software engineering experience is reflected in changing approaches to team management, risk, and other relevant aspects of software engineering.

4.11 Team issues, reassessment, and mitigation

As endlessly repeated in the assessment document, and in lectures and practicals, if a team is not functioning or you feel that the work is not being shared in an equitable way, then you are ***strongly encouraged to contact the lead lecturers*** as soon as possible (email dimitris.kolovos@york.ac.uk, richard.paige@york.ac.uk). As outlined on the SEPR VLE (Teams: TeamInfo.html), there are many ways we can try to help, but we cannot help if we do not know that there is a problem, or if we only find out at or after an assessment submission date.

4.12 Reassessment

If an individual fails SEPR, there are two reassessment elements: there is a resit of the individual examination (a new paper), and there is a reassessment essay on the teamwork. The setters will specify part of the project, and frame the reassessment essay to address the key software engineering learning objectives. The reassessment essay will also require some self-reflection, as in the final part of Assessment 4, to assess the candidate's understanding of the teamwork learning objective.

The candidate is expected to contextualise the stated part of the project, and to complete the reassessment on their own (i.e. without interacting with their team or other SEPR students), using any of their team's materials that they have access to. Reassessment candidates will **not** be able to request or require materials: they must make their own arrangements to access team products.

Remember that SEPR is a 30-credit module, and that a very large number of hours are allocated to the teamwork assessments. It is not going to be a short reassessment or one that is easy to pass.

4.13 Exceptional circumstances affecting assessment (ECA)

Any student can apply for ECA for SEPR in the normal way, providing evidence of the effect of their exceptional circumstances on SEPR. However, we would strongly advise anyone considering this to discuss their situation with the lead lecturers *first* – and, ideally, well before an assessment deliverable may be affected by their circumstances.

The ideal situation is to handle individual problems within the assessment context: we can, for instance, adjust load or help teams to manage a temporary absence.

If ECA is requested and granted for one SEPR team assessment, we would normally suggest a write-off: this gives the student having a mark of 0 for at most 20% of SEPR (so, for assessments 1 or 2, the student carries 0/5; for assessments 3 or 4, the student carries 0/0). We can use the 20% write-off rule because the learning objectives of SEPR are met cumulatively across the 4 assessments, rather than each objective being assessed in only one element. The effect of a write-off can be quite difficult to understand, so here is a hypothetical example.

> *Sam has ECAs accepted for assessment 1 (worth 25% of SEPR). Sam's ECA claim asked for a write-off of the assessment (0/5). Sam experienced no further exceptional circumstances, and received marks of 76, 65 and 70 for the next three assessments. In the individual exam, Sam got 93. So, with the component weighting, Sam's scores are* $0 + (0.25 * 76/100) + (0.2 * 65/100) + (0.2 * 70/100) + (0.1 * 93/100)$ *but this result is out of 80 (20% was written off). Normalising to an integer percentage, Sam gets 70*
>
> *Jo has ECAs accepted for assessment 3 (worth 20% of SEPR). Jo's ECA claim asked for a write-off of the assessment (0/0). Jo experienced no other exceptional circumstances, and received marks of 76 and 65 for the first two assessments, and 70 for assessment 4. In the individual exam, Jo also got 93. So, with the component weighting, Jo's mark is* $(0.25 * 76/100) + (0.25 * 65/100) + 0 + (0.2 * 70/100) + (0.1 * 93/100)$, *which normalises to 73*

If a student with an ECA granted prefers to opt for a "sit as if for the first time" for the affected component, then the setters will construct an essay question that meets the learning objectives covered in that assessment (a mini version of the reassessment described for a complete SEPR fail, in Section 4.12). Please note that, however well-intentioned and generous the setter of the essay, this is not an easy option: because the assessments are used for learning as well as summative marking, each team assessment represents a quarter (assessment 1 or 2) or a fifth (assessment 3 or 4) of the SEPR load, which translates to 75 or 50 hours of student time. An ECA essay is likely to require at least half this original input of time.

To illustrate, let's assume that Sam and Jo opted to sit as if for the first time on the element that was ECA'd; Sam scored 64 and Jo scored 65.

Sam's SEPR mark is thus, $(0.25 * 64/100) + (0.25 * 76/100) + (0.2 * 65/100) + (0.2 * 70/100) + (0.1 * 93/100)/100)$, which is 71.

Jo's SEPR mark is, $(0.25 * 76/100) + (0.25 * 65/100) + (0.2 * 65/100) + (0.2 * 70/100) + (0.1 * 93/100)$, which is 72.

So, for a wrecked summer vac., Sam got one mark more and Jo got one mark less than was on offer for doing no extra work at all (in the write-off case). Even noting that SEPR is 30 credits (a quarter of the year), the gamble isn't worth it!