UNIVERSITY OF GLOUCESTERSHIRE
at Cheltenham and Gloucester

CT4017 Introduction to Software Development

Module Guide

2015-16, Semester 2

Faculty of MAT, School of Computing and Technology

Table of Contents

Large print copies of this booklet can be provided on request. We can also arrange production of braille or audio versions.
Please email disability@glos.ac.uk if you have any special requirements that you would like to discuss.

University of Gloucestershire 2016

Module Content

Module Team

Dr Ahsan Ikram	Park WW103	Ext 4255	aikram@glos.ac.uk
Dr Joanna Isabelle Olszewska	Park WW103	Ext 4267	jolszewska@glos.ac.uk
Jamie Stewart	Park WW103	Ext 4261	jstewart2@glos.ac.uk

Module Guide, Module Content, Exercise Content, Assignment

Paula Thomas	Park WW103	Ext 4256	pthomas2@glos.ac.uk
David Johnston	Park WW103	Ext 4269	djohnston@glos.ac.uk

Module Level and Credit Value: Level 4, 15 CAT points

External Examiner:TBC

Brief description

The module introduces the basics of software development. It develops the student's software engineering techniques, including the design, development and testing of software units; and project management skills.

Learning outcomes

A student passing this module should be able to:

1. Design, develop and test small units of software according to a specification.
2. Appreciate the stages of the software development lifecycle.
3. Explain and use the Application Programming Interface (API) to develop software.
4. Develop understanding of project management packages and utilize them to plan and manage projects.
5. Apply skills relevant for academic progression and career development within the sector.

PDP Statement

Students should, by the end of the module, achieve a high level of understanding of a range of software development life-cycle and group project management

techniques and their useto develop, design, and test a software. They should be in a position to use an appropriate method depending on the application. They will also gain a knowledge of various software design methods and of procedural language programming.Students will develop a variety of material suitable for inclusion in a Personal Development Portfolio.

Teaching &Learning Approach

This level 4 module aims to familiarize students with software development, and it is designed to beinteractive and to contain a variety of activities during the 3-hours timetabled sessions. Besides these sessions, each student is expected to spend at least 7 hours of independent study per week.

The theoretical background underpinning software developmentwill be delivered in the 1-hour lectures and illustrated with running examples and videos. It will be followed by 2-hour lab sessions of immediate practice of these concepts within a practical problem-based approach.These timetabled lab sessions are also the common allocated timeslot to allow the students' groups to meet about their project. Indeed, the exercises are designed to provide students with guidance they need to understand the main concepts in this field and to complete their group assignment.

A workshop will be also organized with research and industry study cases in that field.

You are advised not to leave the assignment to the last minute as designing and developing an entire project takes time.

Feedback on Work

All students are entitled to both formative and summative feedback during the module. Formative feedback is designed to enable you to improve your work/performance and takes many forms. Students will have several opportunities to receive feedback on their progress, e.g. during lab sessions, tutorials or seminars; as part of general feedback in lectures. Summative feedback is received when your submitted coursework has been marked. As a minimum, this will comprise constructive comments from the tutor and a grade.

Indicative Resources

Core textbooks:

Chauhan, N. (2010) *Software Testing*.Oxford University Press.

Foster, E.(2014) *Software Engineering: A Methodical Approach*. Apress; 2014 Ed.

Sommerville, I. (2010) *Software Engineering* (9th ed.), Pearson Education.

Wagner, F., Schmuki, R., Wagner, T. and Wolstenholme, P. (2006) *Modeling Software with Finite State Machines: A Practical Approach*.Auerbach Publications.

Watkins, J. and Mills, S. (2011) *Testing IT : an off-the-shelf software testing process* 2nd Ed., Cambridge University Press.

A selection of journal/conference papers supporting the lectures,e.g.:

Bhat, M. and Olszewska, J. I. (2014) 'DALES: Automated tool for Detection, Annotation, LabElling and Segmentation of multiple objects in multi-camera video streams'.In *Proceedings of the 25th ACL International Conference on Computational Linguistics Workshop (COLING'14)*. vol. 17, no. 3, pp. 87-94.

Gunter, C. A., Gunter, E.L., Jackson, M., and Zave, P. (2000) 'A reference model for requirements and specifications'. *IEEE Software*. vol. 17, no. 3, pp. 37-43.

Wirth, N. (1971) 'Program development by stepwise refinement'. *Communications of the ACM*. vol. 14, no. 4, pp. 221-227.

A variety of relevant languages and environments, such as C and MatLab.

The module will be supported by a Moodle web site http://moodle.glos.ac.uk. This site contains module information; reading list;lecture notes; links torelevant, online journal/conference papers underpinning the lectures;and lab material. In addition, there are online *tutorials* available to help with specific software and techniques.

Module Evaluation

This module is run first time. In this current academic year, you will be given the opportunity to undertake module evaluation which will feed into the course board of studies meeting and will inform module design for the following year. This will be conducted centrally.

Assessment

1. Module code and title	**CT4017 Introduction to Software Development**
2. Module tutor	Paula Thomas
3. Tutor with responsibility for this Assessment	David Johnston- this is your first point of contact
4. Assignment	001: 100% Coursework: Groupwork(2000 words or equivalent). You will be penalised according to the **Academic Regulations for Taught Provision** if you exceed the size limit.
5. Submission deadline Your attention is drawn to the penalties for late submission; see Undergraduate Modular Handbook.	**By 23:55, Thursday April 29th 2016** Your attention is drawn to the penalties for late submission; **see Academic Regulations for Taught Provision.**
6. Arrangements for	**Electronic submission via *Moodle*.**

submission	• *You should use the following convention for naming your folder and files: 'Bloggs_J_s0123456_CT4017_Report_ Group X' (for example,Joe Bloggs –student number s0123456 –report for assignment of module CT4017 – of the Group number X). You are also advised to add this information to the header section of your submitted document.* • *All files must be submitted in a single, main folder per group. The folder should include your group's report (doc file), a video, and subfolders such as:* *- CODE containing all the code files;* *- TEST with training/testing images or any test material used in the project or readme file with link to it;* *- RESULTS with results material/images/videos* • *Information about electronic submission, including a set of Frequently Asked Questions, is available at:* https://infonet.glos.ac.uk/departments/adu/ema/Pages/emastudents.aspx
7. Date and location for return of work	Written feedback and provisional mark should be within 20 working days of submission(working days comprise the normal working week, excluding periods when the University is shut and weekends).
8. Students with Disabilities	Alternative assessment arrangements may be made, where appropriate, for students with disabilities. However, these will only be implemented upon the advice of the Disability Advisor. Students wishing to be considered for alternative assessment arrangements must give notification of the disability (with evidence) to the Disability Advisor by the published deadlines.
9. University Regulations for Assessment	All assessments are subject to the **Academic Regulations for Taught Provision**. These include regulations relating to Errors of Attribution and Assessment Offences. In exercising their judgement, Examiners may penalise any work where the standard of English, numeracy or presentation adversely affects the quality of the work, or where the work submitted exceeds the published size or time limits, or where the work fails to follow normal academic conventions for acknowledging sources.

Assignment worth 100% (2000 words equivalent)

Based on a provided piece of software, which will be analysed in groups during the first weeks (bydiscussing appropriate SDLC and reflecting about it; identifying its requirement, specifications, functionalities; building test cases and performing tests;

etc.), students will have to extend this software and develop a version 2 of this software with a new functionality of their choice.

Complex Calculator:

1. Develop a calculator, using C++, whichtakes an input from the user. Input can be a combination of numbers (1-9), operators (+, -, *, /)
2. Calculator should solve basic calculations (for example, 2+2, 3*4, 8*2 etc.)
3. *Extend* the calculator to solve unary calculations (for example, inverse, power, square root etc.)
4. *Extend* the calculator to have a Graphical User Interface (GUI).
5. *Extend* the calculator to have a memory function to store and retrieve last calculation.
6. *Extend* the calculator to handle multiple operator precedence (for example, 2+4*5/3)
7. Create tests for the calculator.

Students should respect at all times the University Regulations, in particular Ethics Procedures, and BCS Code of Conduct and the project should comply with all H&S regulations.

Submission

Students need to submit the following three components*:*

1. Group Report (LO1, LO2, LO3, LO4, LO5)

The assignment will be a group report consisting of:

- a brief overview of the studied problem, including a mind map;
- comments about the adopted Software Life-Cycle Development (SLCD) as well as reflections about the project management process within the team, using the SWOT analysis and providing a work breakdown structure and a Gantt chart;
- a descriptionof your adopted approach (e.g. used techniques, assumptions, links to the stateof the art) and your adopted design, in particular using DFD diagrams and finite-state machine techniques;
- detailed requirements of the proposed version 2 as well as the analysis of the its specifications and functionalities (in particular, do the MOSCoW analysis of requirements and risk analysis) and justifications how the version 2 is related/improves the version 1 of the software;
- explanations about the software implementation, parameters, and adopted software testing process and metrics;
- a discussion about your results (reflection on testing approach, reflection on performance such as time, accuracy, etc.,);
- conclusions (reflection on the adopted method and alternatives, reflection on the development – what went right/wrong, reflection on Ethics, reflection on possible improvements).

The report should also contain in appendix pictures of flow charts, design diagrams, printscreens of set-up/windows, images/tables with the testcases, test images, and visual results.

The report should include relevant references to the source materials and tools used. Please note that appendices and references are not included in the word count.

2. Code (LO1, LO2, LO3, LO4)

Debugged source code in C++, should be structured and commented.

The code should show extensions as different versions of the software.

3. Video (LO2, LO4, LO5)

A short (less than 5 minutes) video should summarize the key points of your coursework (e.g. general presentation/context of the problem, technical explanations of theproblem, proposed architecture/design, achievements/results, conclusions). This video will be a useful addition to your Personal Development Portfolio and will help you when applying for placements or job positions. You can create this video using CAM Studio or some other screen-capture software such as recordMyDesktop (only for Linux). Please note that you need to upload your video in *Moodle* or provide it via a hard support (e.g. DVD, USB key, etc.), depending on the size of the file.

The assessment contributes to all learning outcomes as indicated.

Take regular backups of your work. This will enable you to recover quickly should the system fail and also allow you to backtrack if your development goes astray.

Ensure that the work submitted will execute on University computers.

Keep evidence of the submission of your assignment, and a copy of your assignment in case of the unlikely event of any loss.

Marking Criteria(based on the School's Assessment Grid)

The following grade table will be used for marking. Note that the overall grade will be determined by the application of the School of Computing & Technology Assessment Criteria Grid (see pp.10-11).

Grade	Content
<30	Requirements not met; not recoverable.
30 to 39	Report is inadequate. Code is not running. Video is missing.
40 to 49	Report meet basic requirement correctly but limited, just adequate but not innovative or interesting. Code quality (functionality, documentation) is limited. Video is provided.

50 to 59	Report is coherent and organised, some evidence of self-criticism concerning deliverables. Code quality (functionality, documentation) is good. Video is correct.
60 to 69	Report has good reflection, is coherent and well organised; good integration of academic & practical knowledge, good evaluation of deliverables. Code quality (functionality, documentation) is excellent. Video is interesting.
70+	Report has excellent reflection, evidence of elegance, analytical analysis, innovation; very good evaluation of deliverables. Code quality (functionality, documentation) is excellent and deals with complex functionalities. Video is excellent.

Special Instructions

'Ground rules' for Group Work:

Permissible group size: 3 to 5 students.

Managing group work is part of the assessment and a brief paragraph of the report should summarize this experience you could highlight when you apply for placements or job positions. Students are advised to document their group activities or keep evidence of them. They could include (some of) them in an Appendix into the report. Changes on memberships of work groups will be not allowed after Week 5.

In each group, one person should volunteer to be responsible for submitting the group work in time by uploading the folder with all the components into Moodle (see also above-mentioned explanations).

For each group, all students will have the mark for the group, unless there is strong evidence one student had not contributed to the group project and this student having poor attendance, i.e. attendance of less than 75% of the tutorials.

Warning on Assessment Offences:

Careful referencing of sources is vital when making use of the work of others. You are expected to employ the referencing conventions recommended in the Course. These conventions apply to information taken from internet sources, as well as books, journals and lectures. If you are unsure of the way to reference properly, seek advice from a member of staff before you submit the assessment.These are some of the points you should check before submitting your work:

- are all direct quotations, from both primary *and* secondary sources, suitably acknowledged (placed in quotation marks or indented)?

- have you provided full details of the source of the quotation, according to the referencing convention used in the Course?
- have you acknowledged the source of ideas not your own, even if you are not quoting directly from the source?
- have you avoided close paraphrase from sources? (Check that you are not presenting other people's words or phrasing as if they are your own.)
- if you have worked closely with others in preparing for this assessment, is the material you are presenting sufficiently your own?

A Harvard referencing tutorial can be found at:
http://ist.glos.ac.uk/referencing/harvard/

The tutorial should take approx 20 minutes to work through and includes lots of examples of different types of resources.It is based on the "Cite them right" book (Richard Pears and Graham Shields (2008).*Cite them right: the essential referencing guide*. Newcastle upon Tyne, Pear Tree Books)available on Amazon and copies in the Learning Centre. More details about referencing, including a quick referencing guide and a tutorial about plagiarism, details of how to reference sources such as websites, online journals, newspaper articles, and official publications, are available at:
http://insight.glos.ac.uk/departments/lis/resources/Pages/referencing.aspx

If you are unsure of the way to reference properly, seek advice from a member of staff before you submit the assessment. In submitting your work for assessment, you are making a statement that it is your own work, it has not been submitted for any other assessment, and it does not infringe the ethical principles set out in the University's **Research Ethics: Principles and Procedures.**

School of Computing & Technology Assessment Criteria Grid

Mark %	Grade & Characteristics	Theory & Academic Approach	Practice & Deliverables
0	Fail	plagiarism, collusion, non-pres., name only	as theory
1-9	Fail	no understanding, very short, inadequate	no effective deliverables
10-29	Fail	factual but little interpretation, lacks coherence, short, errors, misconceptions	requirements not met; not recoverable
30-39	Reassess: inadequate but recoverable with limited effort	coherent but mechanical notes, basic task OK but limited - partial - rudimentary answer, limited interpretation, lack of knowledge of topic, weak English but some appropriate use of language of topic	deliverables partially complete, not all requirements met, limited response to brief.

40-49:	3rd, D Pass: Sufficient for award of credit adequate mainly descriptive approach, fair, limited conceptual or theoretical ability	adequate response, demonstration of basic knowledge, relevant content, clear intention communicated, evidence of reading, acceptable minimum level of English for business presentation but may lack precision, some limited analysis / application of knowledge / theory / weighting of evidence, inconsistent	deliverables meet basic requirement correctly but limited, just adequate but not innovative, interesting or exciting, for higher marks, 45+ just exceeds minimum specification, might be good in some areas but not consistent
50-59	2ii, C Satisfactory Satisfactory with some conceptual ability but lacks good evaluation or synthesis of ideas	good response to task, collates info, *satisfactory* analysis & judgement, constructs generalisations based on evidence & opinion, argues clearly, logically & constructs a case, some limited ability to state a personal position, correct English with few imprecise statements	good deliverables, some evidence of good design or execution, coherent and organised product, some limited evidence of self criticism concerning deliverable, some independence, initiative, autonomy, appropriate techniques, integration of knowledge for task
60-69	2i, B Good. Good analysis, evaluation, synthesis, integration & argument.	evaluates info. & synthesises generalisations, good ability to state & defend personal position, good analysis & judgement, applies knowledge to new situations, sound on theory, critical, understands limitations of methods, selective coherent & logical approach, well written with clear, correct and precise English	all criteria met to good standard, evidence of good design or execution, good integration of academic & practical issues, solid evidence of self critique/evaluation of deliverables, products well organised - documented - coherent. Evidence of independence, initiative, autonomy, creativity, adaptability, resourcefulness. Integration of knowledge,
70-79	first class, A, Excellent. as above but also stronger evidence of excellent, original, innovative, articulate work	very strong ability to state & defend position, uses criteria & weighting in judgements, wide knowledge and theoretical ability, full understanding of possibilities and limitations of methods & theories, *75+* more original, innovative approach, command of critical positions, lively articulate writing, excellent grasp of material - synthesis of ideas	most criteria met to high standard, strong evidence of evaluation of deliverables, *75+:* deliverables excellent - all criteria met in clear and definite manner, evidence of excellent design or execution, elegance, innovation, very good evaluation of deliverables,

80-89	Outstanding. as above but also authoritative, superlative, creative	as above but also :- seen all possibilities in task, gone beyond accepted conceptual/critical positions, evidence of creative, intelligent, innovative approach consistently & forcefully expressed	as above but also :- all aspects of deliverables superlative beyond 80% emphasis on theory rather than practice/deliverables
90-100	Faultless	as for 80-89 but also :- all work superlative & without fault	as for 80-89

Provisional Scheme of Work

Lectures: Wednesdays 10.15 to 13.15 in LC204 or
Thursdays 13.15 to 16.15 in LC204

Date	Lecture Topic	Exercise
W1	Software Development Life Cycle.	GANTT chart creation and update. Create a flowchart for the assignment. Translate the flowchart to Pseudocode. Break the pseudocode down to classes and functions.
W2	Software Development Methodology	Use 'Pair Programming' to develop that asks users for a shape (circle, triangle) then prints it on screen using any character. ($, >, ? etc.)
W3	Versioning	Use bit bucket and the pair from last week to pull, push, branch and label code.
W4	Finite-State Machines	Develop (on paper) a FSM for an ATM machine.
W5	Project Management	Create a SCRUM team, run a dummy sprint for a software that adds two numbers.
W6	Programming Data Structures	Implement a tree using your own data structures. Write an algorithm to visit each node of the tree.
W7	Programming Classes	Create two classes for a game; Game, Player.
W7b	ESY Week	
	Easter Break – 2 weeks	
W8	Software Verification & Validation	Identifying test cases and preparing test suite for a calculator.
W9	Problem Solving	Collections and sorting. Create a collection of numbers given by a user and then sort it in an order again selected by the user.
W10	User Interface Design	MVC exercise + event handling. Create a GUI that prints the key pressed on a screen.
W11	Assignment Workshop	Hand over your assignment code to another group for user testing and feedback.
W12	Software Engineering Workshop	