University of Lancashire School Engineering and Computing	**Coursework Assessment Brief**	2025 – 2026
	Module Title: Advance Software Modelling Module Code: CO3408	Level 6
	Formal Specification and Modelling of a Real-Time Embedded Control System	This assessment is worth 50% of the overall module mark

Learning Outcomes Assessed

1. Develop and interpret specifications in a formal notation.
2. Use and evaluate models in software development.
3. Analyse concurrency problems in a range of applications.
4. Use appropriate tools and methods to develop a small concurrent system in an appropriate language.
5. Evaluate software development techniques.

Scenario

You are part of an embedded software engineering team developing control software for a real-time temperature regulation unit used in smart laboratory equipment. The system continuously monitors multiple temperature sensors and controls cooling and heating components to maintain a stable operating temperature.

As this system may be deployed in high-integrity electronics, its software must be formally specified, verified, and tested to meet strict reliability and safety standards. You will apply formal specification techniques, UML and OCL modelling, Design by Contract, and concurrent system design to demonstrate software correctness and quality assurance.

System Description and Decomposition

Functional Requirements (examples)

1. The system must continuously read temperature values from multiple sensors.
2. The system must compare each reading to a target temperature range.
3. The system must activate cooling or heating components when temperature deviates from the set range.

Non-Functional Requirements (examples)

1. Accuracy: Temperature readings must be accurate within ±0.1°C.
2. Performance / Real-time response: The system must respond to temperature changes within 1 second.
3. Reliability: The system must ensure consistent operation under varying environmental conditions.

System Decomposition

The system can be decomposed into the following modules:

- SensorInput – captures and filters real-time temperature data from multiple sensors.

- TemperatureController – compares readings to the target range and determines corrective action.
- ActuatorControl – sends commands to heating and cooling components.
- SystemMonitoring – logs readings, monitors performance, and triggers alerts when thresholds are breached.

Assessment Tasks

Task 1 – System Requirements and Decomposition (≈ 500 words + diagram)

- Identify the functional and non-functional requirements of the temperature control software.
- Decompose the system into subsystems or modules (e.g., SensorInput, TemperatureController, ActuatorControl).
- Present a high-level UML use case diagram showing system actors and interactions.
- Explain how each requirement supports system reliability or performance goals.

Deliverables: Use case diagram + requirements table + brief justification.

Task 2 – UML Class and Behavioural Modelling with OCL (≈ 600 words + diagrams)

- Create UML class diagrams representing the system structure, including key classes, relationships, and multiplicities.
- Apply Object Constraint Language (OCL) to formally specify:
  - Invariants (e.g. temperature range must stay between defined limits),
  - Preconditions (e.g. a fan can only activate if the sensor value exceeds the threshold), and
  - Postconditions (e.g. after activation, the system state must reflect the new mode).
- Provide UML sequence or activity diagrams to model key system behaviours (e.g. temperature reading and control loop).

Deliverables: UML class + behavioural diagrams with embedded OCL expressions and explanation.

Task 3 – Formal Specification and Design by Contract (≈ 700 words)

- Translate part of your system design (e.g. TemperatureController class) into a formal specification using JML or OCL, enriched with formal contracts.
- Define preconditions, postconditions, and invariants for key operations.
- Explain how Design by Contract (DbC) principles ensure correctness and prevent software faults.
- Discuss how the specification supports verification and testing.

Deliverables: Formal specification (JML or annotated Java code) with commentary on correctness and reliability.

Task 4 – Concurrency and Real-Time Behaviour (≈ 700 words + diagrams/code)

- Model the concurrent components of the system (e.g. multiple sensor threads accessing shared data).
- Use state machines or activity diagrams to represent communication and synchronisation.
- Identify potential concurrency issues such as race conditions, deadlocks, or non-determinism, and propose solutions (e.g. semaphores, mutexes, monitors).
- Provide real Java code demonstrating thread management or real-time scheduling.

Deliverables: State/concurrency diagrams + brief technical discussion and Java code examples.

Deliverables Summary

- Use case and UML diagrams (Tasks 1–2)
- OCL constraints and Design by Contract specifications (Tasks 2–3)
- State/concurrency models and actual implementation code (Task 4)
- Technical report task1 – task4 (~2,500 words total ±10%)

5. Submission Deliverables

You are required to submit this assessment via Blackboard. Your assessment will be evaluated against the published marking criteria. If you are unable to complete a lower-level requirement but successfully implement a higher-level feature, this may still be credited where appropriate and justified in your report.

Submission Components in zip file

Technical Report (≈ 2,500 words ±10%)

- Include a title page (student ID, assignment title, and word count).
- Structure the report clearly using headings for Tasks 1–4.
- Integrate diagrams, tables, and state/concurrency models within the relevant sections.
- Use University of Lancashire Harvard referencing style for all citations and references.

Formal Specification Files / Actual Code

- Submit formal specification files (e.g., JML or annotated UML diagrams) or real Java code demonstrating contracts, OCL expressions, and concurrency solutions.
- Include clear comments or docstrings explaining the purpose and logic of each major section.
- Ensure files are readable, logically organised, and reflect Tasks 2–4.

Figures, Screenshots, and Visual Outputs

- Provide screenshots or exported figures that display diagrams, sequence flows, state machines, or sample outputs.
- All figures must be clearly labelled, numbered, and referenced within the report.

Note: There is no demonstration requirement for this assessment. Marks will be awarded solely based on the submitted report, files, and supporting diagrams/code.

6. Marking Scheme

40–49% (Pass)

- Basic system analysis and decomposition provided (Task 1).
- UML diagrams or OCL annotations present but limited in detail.
- Formal specifications/pseudocode demonstrate partial understanding of contracts.
- Concurrency/state diagrams included but may lack clarity or correctness.
- Report includes essential sections but lacks depth or accuracy.
- Figures/diagrams may be incomplete or inconsistently labelled.

50–59% (2.2)

- Clear system decomposition and consistent UML models.
- OCL or formal contracts correct but may omit some invariants/pre/postconditions.
- Concurrency and real-time behaviour partially addressed; some synchronisation concepts demonstrated.
- Report demonstrates reasonable technical discussion with some analysis.
- Figures, Java code, and diagrams correct but analysis limited.

60–69% (2.1)

- Accurate and efficient system decomposition and UML modelling.
- Formal specifications (JML/OCL) demonstrate clear preconditions, postconditions, and invariants.
- Concurrency and real-time issues modelled appropriately, with correct synchronisation strategies.
- Report is technically accurate, well-structured, and demonstrates analytical thinking.
- Figures, diagrams, and runnable Java code clearly labelled and integrated.

70–100% (1st Class)

- Outstanding clarity and depth in problem definition, system decomposition, and abstraction (Task 1).
- Excellent UML/OCL modelling with complete, correct, and insightful constraints (Task 2).
- Highly modular, efficient, and well-documented formal specifications and runnable Java code (Task 3).
- Comprehensive concurrency and real-time modelling, with professional-quality diagrams and clear explanation of solutions (Task 4).
- Deep critical analysis linking formal models, specifications, and system behaviour.
- Figures, diagrams, and documentation are professional quality.

PREPARATION FOR THE ASSESSMENT

Before attempting this assessment, you should complete the exercises in weeks 1 – 5 on Blackboard

RELEASE DATES AND HAND IN DEADLINE

Assessment Release date: 22/10/2025

Assessment Deadline Date and time: 06/02/2026, 17:00 hours.

Please note that this is the latest time you can submit – not the time to submit!

SUBMISSION DETAILS

You should compress your source code work into a single zip file and submit using the assignments tab on Blackboard. You should also upload your report in a Microsoft Word document and submit in the appropriate section using the assignments tab on Blackboard. Do not use other compression formats. For each part of your submission, clearly identify the question you are answering.

Report marking:
Your submitted report will be marked, and the results and feedback will be made available on Blackboard. Marks will be awarded based on the quality of your submitted report, supporting diagrams, formal specifications, and real Java code. There is no demonstration requirement.

This is an individual project and no group work is permitted:
You may conduct your own research into topics which have not been explicitly taught within the module (this will be required for higher marks) however, you should include a justification for its use alongside links to any sources in your program comments. Failure to do so may result in a plagiarism investigation. Only use concepts that you are confident you understand.

Use of AI-based Tools:
AI-based tools may be used for research purposes only. You should not attempt to generate code for use within your assignments, as doing so may result in an academic misconduct investigation. You should submit screenshots of any prompts used alongside your assignment coversheet. You should also reference where each prompt has been used within your code.

HELP AND SUPPORT

- Questions regarding this assessment should be asked through the CO3408 Module Teams Channel "Lab Work and Assessment"
- For support with using library resources, please contact Bob Frost, r.frost6@lancaster.ac.uk or SubjectLibrarians@uclan.ac.uk. You will find links to lots of useful resources in the My Library tab on Blackboard.

- If you have not yet made the university aware of any disability, specific learning difficulty, long-term health or mental health condition, please complete a Disclosure Form. The Inclusive Support team will then contact to discuss reasonable adjustments and support relating to any disability. For more information, visit the Inclusive Support site.
- To access mental health and wellbeing support, please complete our online referral form. Alternatively, you can email wellbeing@uclan.ac.uk, call 01772 893020 or visit our UCLan Wellbeing Service pages for more information.
- If you have any other query or require further support you can contact The <i>, The Student Information and Support Centre. Speak with us for advice on accessing all the University services as well as the Library services. Whatever your query, our expert staff will be able to help and support you. For more information , how to contact us and our opening hours visit Student Information and Support Centre.
- If you have any valid mitigating circumstances that mean you cannot meet an assessment submission deadline and you wish to request an extension, you will need to apply online prior to the deadline.

Disclaimer: The information provided in this assessment brief is correct at time of publication. In the unlikely event that any changes are deemed necessary, they will be communicated clearly via e-mail and a new version of this assessment brief will be circulated.	Version: 1