Module Code
COM00111M

MSc, MEng and MMath Degree Examinations 2018–9

DEPARTMENT OF COMPUTER SCIENCE

Model-Driven Engineering (MODE)

Open Individual Assessment

Issued: October 03, 2018, 12:00 (noon)

Submission due: January 09, 2019, 12:00 (noon)

Feedback and marks due: February 06, 2019, 12:00 (noon)

All students should submit their answers through the electronic submission system: http://www.cs.york.ac.uk/student/assessment/submit/ by January 09, 2019, 12:00 (noon). An assessment that has been submitted after this deadline will be marked initially as if it had been handed in on time, but the Board of Examiners will normally apply a lateness penalty.

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

Any queries on this assessment should be addressed by email to Prof. Dimitris Kolovos at dimitris.kolovos@york.ac.uk. Answers that apply to all students will be posted on the VLE.

Rubric:

Your submission should include a report and an implementation (in EMF, GMF/Xtext and Epsilon). You must submit a single zip file containing your report in PDF, and your implementation files. Note the page limits: parts of answers that go beyond the page limit may not be marked. Use IEEE citation and referencing: references must be listed at the end of the document and do not count towards the page limit.

Your exam number should be on the front cover of your assessment. You should not be otherwise identified anywhere on your submission.

1 The Exercise

You are to design and implement a domain-specific language (DSL) and supporting tooling for capturing and managing issues (e.g. bug reports, enhancement requests) related to software products.

In this DSL, a model comprises a number of products, issues, and team members. There are two types of issues: bug reports (also referred to as "bugs" below) and enhancement requests. Each issue has an identifier, a title and a longer textual description, and can be flagged as open or closed (i.e. resolved/fixed). Issues can be marked as duplicates of other issues and can be blocked by other issues. For example, issue A can be marked as a duplicate of issue B because they essentially describe the same undesirable behaviour, while bug C can block enhancement request D in the sense that D cannot be implemented until C has been fixed. Issues are reported by team members, and can be assigned to one or more team members to fix/resolve. Team members can also comment on issues and add replies to existing comments. Additional desirable properties for models conforming to the DSL include:

- The identifier of each issue must be unique
- An issue cannot be a duplicate of itself
- There must be no direct or indirect blocking cycles (e.g. if issue A blocks issue B, issue B must not be blocking issue A)
- The textual description of an issue must be at least 10 characters long
- Bugs can block enhancement requests, but not the other way round
- Each product can have several versions in the model (e.g. Word 2017, Word 2018)
- Issues can be related to multiple versions of products
- Closed issues cannot have open blockers

2 Questions

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 the document and do not count towards page limits.

2.1 Metamodel

Use Emfatic/Ecore to define a metamodel for the issue management language described in Section 1. Present and discuss the metamodel, state any assumptions you have made, and explain any alternative design decisions that you have considered and discounted. [15 marks] (max 2 pages)

2.2 Concrete Syntax and Editor

Use any of the technologies introduced during the module's lectures and practicals to define a concrete syntax and implement a supporting editor for your language. Provide a screenshot of your editor, discuss and justify your syntax design and editor implementation decisions, and reflect on the strengths and weaknesses of the selected concrete syntax compared to alternatives. [15 marks] (max 3 pages)

2.3 Model Validation

Use the Epsilon Validation Language to implement any validation constraints specified in Section 1, which cannot be expressed in the metamodel itself. Briefly explain the rationale and implementation of each constraint [15 marks]. Integrate the implemented constraints with the editor discussed in Section 2.2 to provide in-editor errors/warnings [5 marks]. (max 2 pages)

2.4 Model-to-Text Transformation

Use the Epsilon Generation Language to implement a model-to-text transformation that consumes a model conforming to your issue management language, and generates a set of hyperlinked HTML pages through which users can

- list all products and team members in the model
- view issues reported by and assigned to specific team members
- view issues related to a specific product or product version

- view the details and comments of a specific issue
- list blocking issues in order of importance (the more other issues it transitively blocks, the more important an open issue is)

Discuss the model-to-text transformation, and justify the format and organisation of the generated HTML pages. [20 marks] (max 2 pages)

2.5 Model-to-Model Transformation

Use the Epsilon Transformation Language to implement a model-to-model transformation that consumes an issue model and produces a simplified issue model (conforming to the same metamodel), which excludes closed issues as well as team members and products that are only associated to such issues. The transformation **must not** modify its source model. [20 marks] (max 2 pages)

Quality and conciseness of description and depth of analysis are more important than length. The report and the implementation will also be marked for the clarity, precision and conciseness with which your ideas are communicated. [10 marks]

End of examination paper