Masters of Science in Software & Information Systems

To be developed and delivered in conjunction with Regis University, School for Professional Studies

Real-Time Systems

January 2006
# Table of Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Details</td>
<td>3</td>
</tr>
<tr>
<td>Course Outline</td>
<td>6</td>
</tr>
<tr>
<td>Workshop 1</td>
<td>7</td>
</tr>
<tr>
<td>Workshop 2</td>
<td>9</td>
</tr>
<tr>
<td>Workshop 3</td>
<td>11</td>
</tr>
<tr>
<td>Workshop 4</td>
<td>13</td>
</tr>
<tr>
<td>Workshop 5</td>
<td>15</td>
</tr>
<tr>
<td>Workshop 6</td>
<td>17</td>
</tr>
<tr>
<td>Workshop 7</td>
<td>19</td>
</tr>
<tr>
<td>Workshop 8</td>
<td>21</td>
</tr>
<tr>
<td>Appendices</td>
<td>22</td>
</tr>
</tbody>
</table>
Section 2  Module Details

Programme Title: M.Sc. in Software and Information Systems

Module Name: Real-Time Systems

Course Description:
This module will provide a firm foundation in the knowledge and skills needed to develop embedded real-time systems. The module is split into three parts, i.e. (1) foundations, (2) designing and developing real-time software and (3) implementation and performance issues.

Part (1) deals with the foundations and begins with an introduction into real-time systems. It continues with a discussion on dependable software and describes the required steps to produce this kind of software. Therefore we’ll have an overview into requirements analysis and specification as well as program design concepts for embedded real-time systems. An introduction into real-time operating systems (RTOS) will finalise part (1).

Part (2) begins with practical aspects of RTOSs. The module continues with an introduction into diagramming and its benefits for real-time systems design. A couple of practical diagramming methods like UML are recalled and their application to real-time system design will be shown. The module moves on to code-related issues and discusses how (real-time) software should be designed and implemented and how a high-level implementation language with appropriate features might be chosen. Part (2) finishes with software analysis and design methodologies, whereby viewpoint techniques, the Yourdon structured model and object-oriented analysis and design methods are discussed.

Part (3) describes in some detail how source code can be tested and analysed using static and dynamic analysis methods. The module continues with development tools, whereby the various debugging techniques are discussed. It also contains an introduction into mission-critical and safety-critical systems and outlines their difficulties and needs in terms of hardware and software prerequisites. The final workshop deals with performance engineering and the various types of performance modelling. This workshop also includes an assessment of documentation requirements both for system functional specifications and software system specifications.

Prerequisites:
Course Objectives:
At the end of the course, students are expected to be able to competently:
• Describe the architecture and organisation of (embedded) real-time systems.
• Explain the concept of dependable software.
• Recall the requirements and features of RTOSs.
• Describe the importance of the various diagramming methods.
• Choose an appropriate programming language for a real-time project.
• Analyse and test the source code of a real-time application.
• Distinguish between mission-critical and safety-critical systems and their characteristics.
• Describe the concept of performance engineering and the various types of performance modelling.

Required Texts:


Course Emphasis on Fundamental Skills:
This module provides an introduction to (embedded) real-time systems. The emphasis is on the efficient and good design of real-time systems using state-of-the-art methodologies. On completion of this course, students will have a firm foundation in the knowledge and skills needed to develop and produce real-time (and in particular, embedded) systems.

Course Assignments:
Student assessment will take the form of:
- Review questions
- Weekly research or practically based assignments
- A final exam consisting of theoretical questions
Grading:

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Weight</th>
<th>Workshop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participation</td>
<td>15%</td>
<td>1-8</td>
</tr>
<tr>
<td>Labs</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>Final Exam</td>
<td>40%</td>
<td>8</td>
</tr>
<tr>
<td>Weekly assignments</td>
<td>45%</td>
<td>1-8</td>
</tr>
<tr>
<td>Total</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>Workshop</td>
<td>Text Reading</td>
<td>Topics</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Workshop 1</td>
<td>Cooling, chapters 1.1 – 1.4 &amp; 2.1 – 2.4</td>
<td>Introduction, Real-Time Systems and Dependable Software</td>
</tr>
<tr>
<td>Workshop 2</td>
<td>Cooling, chapters 3.1 – 3.5 &amp; 4.1 – 4.6</td>
<td>Requirement Analysis, Requirement Specification and Software Design Concepts</td>
</tr>
<tr>
<td>Workshop 3</td>
<td>Cooling, chapters 5.1-5.8 &amp; 6.1-6.7</td>
<td>Real-Time Operating Systems (RTOSs)</td>
</tr>
<tr>
<td>Workshop 4</td>
<td>Cooling, chapters 7, 8.1, 8.2 (without 8.2.7), 8.3.7, 8.4, 8.5, 9.1, 9.2.1, 9.2.2, 9.2.3, 9.2.4, 9.3, 9.4</td>
<td>Diagramming and Code-Related Issues of Software Design</td>
</tr>
<tr>
<td>Workshop 5</td>
<td>Cooling, chapters 10.1 - 10.3, 11.1 - 11.5</td>
<td>Methodologies of Software Analysis and Design; Analysing and Testing Source Code</td>
</tr>
<tr>
<td>Workshop 6</td>
<td>Cooling, chapters 12 &amp; 13</td>
<td>Development Tools; Mission-Critical and Safety-Critical Systems</td>
</tr>
<tr>
<td>Workshop 7</td>
<td>Cooling, chapters 14 &amp; 15</td>
<td>Performance Engineering and Documentation</td>
</tr>
<tr>
<td>Workshop 8</td>
<td>No new reading</td>
<td>Course review</td>
</tr>
</tbody>
</table>
Section 3 Workshop Syllabus

Workshop One (Introduction, Real-Time Systems and Dependable Software):

➢ To be completed before Workshop One

1. Read this complete module handbook
2. Read Cooling, chapter 1.1 – 1.4 & 2.1 – 2.4.

➢ List of topics to be covered

- Categorisation of computer systems by the speed of response
- Categorisation of real-time systems
- Attributes of real-time systems
- Structure of real-time systems
- Characterisation of embedded systems
- Computing elements of real-time systems
- Single-chip microcontrollers and digital signal processors
- Concept of dependable software
- Categorisation of software errors
- Basics of good real-time software design
- Error avoidance and defensive programming

➢ Course Objectives for Workshop One

Upon completion of this workshop, students are expected to:

- Describe the important features of real-time systems
- Explain how real-time systems may be categorised in terms of speed and criticality
- Describe the range of (embedded) real-time applications
- Realise that environmental and performance factors are key drivers in real-time systems design
- Name the basic component parts of real-time computer units
- Explain the essential differences between microprocessors, microcomputers and microcontrollers
- Explain why, in the real world, we can never guarantee to produce fault-free systems
- Define what is meant by correct, reliable and safe software
- Define what is meant by dependable software
**Suggested Activities**

1. The facilitator and students introduce themselves.
2. The facilitator explains the module objectives, grading criteria, assignments and administrative matters.
3. The facilitator reviews the concepts of the reading material.
4. The class discusses the current topics:
   - The range of (embedded) real-time applications.
   - Basic component parts of real-time computer units.
   - Dependable software.
5. The facilitator reviews this week’s assignment.
Workshop Two (Requirement Analysis, Requirement Specification and Software Design Concepts):

➢ To be completed before Workshop Two

1. Read Cooling, chapters 3.1 – 3.5 & 4.1 – 4.6.
2. Complete previous week's assignment.

➢ List of topics to be covered

- Software lifecycle and statement of requirements
- A realistic software life-cycle model
- The importance of the requirements stage
- Typical mistakes in real-time software development
- Practical approaches to analysis and specification
- Use case analysis
- Specification of functional, non-functional and development requirements
- Software prototyping
- Animation prototyping

➢ Course Objectives for Workshop Two

Upon completion of this workshop, students are expected to:

- Explain the realistic development of software in real-time systems
- Describe the problems associated with formulating, communicating and understanding requirements
- Explain why tool support is important for developing and maintaining analysis and specification documents
- Explain the basics of use case analysis
- Explain the role of prototyping
- Describe the basics of functional, object-oriented and data flow design methods
- Explain how coupling and cohesion can be used as indicators of the quality of a software design
- Define what is meant by hierarchical structuring
- Assess real designs that are based on DFD methods
Suggested Activities

1. The class reads the workshop presentation slides
2. The class discusses the current topics:
   - Benefits of use case analysis
   - Modularisation techniques, coupling and cohesion
   - Prototyping techniques
   - Data Flow Diagrams
3. The facilitator reviews this week’s assignment.
Workshop Three (Real-Time Operating Systems (RTOSs)):

➢ To be completed before Workshop Three

1. Read Cooling, chapters 5.1 - 5.8 & 6.1-6.7.
2. Complete previous week's assignment.

➢ List of topics to be covered

   o Benefits of operating systems in a real-time environment
   o Basic features of real-time operating systems
   o Scheduling - concepts
   o Control of shared resources
   o Deadlocks
   o The priority inversion problem
   o Inter-task communication
   o Memory management
   o Common architectures for embedded systems
   o Basic structures and features of RTOSs
   o POSIX

➢ Course Objectives for Workshop Three

Upon completion of this workshop, students are expected to:

   o Explain the basics of RTOSs
   o Explain the role of scheduling in the operation of a RTOS
   o Define the concept of resource sharing
   o Define the concepts of flags, semaphores and monitors
   o Define the purpose of flags, pools, channels and mailboxes
   o Explain how interrupt-driven designs provide a simple form of multitasking
   o Describe the concepts and code organisation of nanokernels and microkernels
   o Describe the role of MMUs within these kernels
   o Describe the fundamentals of POSIX

➢ Suggested Activities

   1. The class reads the workshop presentation slides
   2. The class discusses the current topics:
      • Common architectures for embedded systems
• Concepts operating system models
• Interrupt-based multitasking systems
• Strategies for dealing with deadlocks
3. The facilitator reviews this week’s assignment.
Workshop Four (Diagramming and Code-Related Issues of Software Design):

➢ To be completed before Workshop Four

1. Read Cooling, chapter 7, 8.1, 8.2 (without 8.2.7), 8.3.7, 8.4, 8.5, 9.1, 9.2.1, 9.2.2, 9.2.3, 9.2.4, 9.3, 9.4.
2. Complete previous week's assignment.

➢ List of topics to be covered

- Benefits of diagrams
- The essentials of software diagrams
- Real-time systems diagramming
- Diagrams for structured and data flow designs
- UML diagrams
- Extensions, variations and project-specific diagrams
- Diagrams and the design process
- Fundamental design and construction methods
- Software components
- Important features of programming languages
- Choosing a high-level programming language for embedded systems

➢ Course Objectives for Workshop Four

Upon completion of this workshop, students are expected to:

- Explain the importance of diagramming as a core design tool
- Explain how diagrams bring rigour, clarity and formality to the design process
- Describe how diagrams are used for analysis, design, documentation and maintenance
- Describe how diagrams fit into the overall software design process
- Define a basic range of diagrams that would support the development of real-time systems
- Identify which basic modelling features need to be supported by diagrams
- Define the terms monolithic, modular and independent when applied to the development of software
- Explain how these relate to the current mainstream embedded programming languages
o Describe the benefits of using component technology in modern software systems
o Assess prospective candidate languages for use in embedded systems

➢ Suggested Activities

1. The class reads the workshop presentation slides
2. The class discusses the current topics:
   • Diagramming methods
   • Important features of programming languages
   • Languages for use in embedded systems
3. The facilitator reviews this week’s assignment.
Workshop Five (Methodologies of Software Analysis and Design; Analysing and Testing Source Code):

➤ To be completed before Workshop Five

1. Read Cooling, chapter 10 & 11.
2. Complete previous week's assignment.

➤ List of topics to be covered

- The development process
- Requirement analysis using viewpoint techniques – CORE
- Yourdon structured method
- General concepts of software testing
- Static analysis
- Source code metrics
- Dynamic analysis
- Integration testing
- Metrics for object-oriented designs

➤ Course Objectives for Workshop Five

Upon completion of this workshop, students are expected to:

- Distinguish between concepts, diagramming and processes when applied to software systems
- Explain the underlying concepts of software testing of source code structures
- Define the major quality and performance assessment points during software testing
- Define what static analysis is and describe how it is carried out
- Describe the features of automated static analysers
- Describe the features and use of McCabe cyclomatic complexity metric
- Define black-box testing, white-box testing and dynamic analysis
- Explain how dynamic testing can be carried out in both manual and automated environments
- Describe which test techniques can be applied to integration testing

➤ Suggested Activities
1. The class reads the workshop presentation slides
2. The class discusses the current topics:
   - Requirement analysis using viewpoint techniques
   - Yourdon structured method
   - Static and dynamic software testing
   - Metrics for object-oriented designs
3. The facilitator reviews this week’s assignment.
Workshop Six (Development Tools; Mission-Critical and Safety-Critical Systems):

➢ To be completed before Workshop Six

1. Read Cooling, chapter 12 & 13.
2. Complete previous week's assignment.

➢ List of topics to be covered

- The development process
- Software debugging – general concepts
- Software debugging on the host
- Software debugging in the target – software-based techniques
- Software debugging in the target – hardware-based methods
- Software debugging in the target – combined techniques
- Debugging in host-as-target combinations
- In-target analysis tools
- Installing firmware into the target
- Integrated development environments
- Overview of critical systems
- System specification aspects of mission or safety critical systems
- Application software aspects of mission or safety critical systems
- Real-world interfacing of mission or safety critical systems
- Processor problems with mission or safety critical systems
- Hardware-based fault tolerance

➢ Course Objectives for Workshop Six

Upon completion of this workshop, students are expected to:

- Describe the details of the various software development processes
- Name the three major environments in which software may be tested and debugged
- Describe the nature and characteristics of the various development environments
- Recall the features of the various debug-tools
- Describe the key features and requirements of mission or safety critical systems
- Explain how redundancy is used to increase reliability of systems
o Explain how software techniques can alleviate problems caused by hardware and software

➢ Suggested Activities

1. The class reads the workshop presentation slides
2. The class discusses the current topics:
   - Software debugging techniques
   - System specification and application software aspects of mission or safety critical systems
   - Hardware-based fault tolerance
3. The facilitator reviews this week’s assignment.
Workshop Seven (Performance Engineering and Documentation):

➢ **To be completed before Workshop Seven**

1. Read Cooling, chapter 14 & 15.
2. Complete previous week's assignment.

➢ **List of topics to be covered**

   o The importance of performance engineering
   o Requirements, targets and achievables of performance engineering
   o Requirement driven performance modelling
   o Result driven performance modelling
   o Risk driven performance modelling
   o Some practical issues in performance engineering
   o The role of documentation
   o Software life-cycle documentation
   o System functional specifications
   o Software system specification
   o Source code aspects of documentation
   o Configuration management and version control

➢ **Course Objectives for Workshop Seven**

Upon completion of this workshop, students are expected to:

   o Describe the various types of timing requirements of (embedded) real-time systems
   o Describe the difference between reactive and proactive design processes
   o Define what software performance engineering (SPE)
   o Explain the underlying concepts of the three types of performance modelling
   o Explain why comprehensive, correct and usable documentation is an essential part of computer-based projects
   o Describe the roles, content and usage of software and system specification documents
   o Recall the importance of properly structured source-code documents
   o Explain the concept of version control
   o Explain how version control relates to configuration management
Suggested Activities

1. The class reads the workshop presentation slides
2. The class discusses the current topics:
   - Performance engineering and performance modelling
   - Documentation issues
   - Version control and configuration management
3. The facilitator reviews this week’s assignment.
Workshop Eight (Course Review):

➢ To be completed before Workshop Eight

1. Review the entire course reading.

➢ List of topics to be covered

   o All topics as discussed in the previous 7 workshops

➢ Course Objectives for Workshop Eight

Upon completion of this workshop, students are expected to:

   o Present a body of work that demonstrates their knowledge of real-time systems.

➢ Suggested Activities

1. The class completes the final exam.
2. The students submit the course evaluation forms and written comment sheets to the student representative.
Section 4 Appendices

(a) Grading Criteria for Written Submissions
A standard template will be devised for this section, however if writers are using a grading template for similar courses, please include here for discussion with Faculty group.

(b) References
See textbook for additional references.

(c) Information Sources
See textbook for additional information sources.

(d) Guidelines for software piracy, plagiarism, using sources in academic works
Standard response will be provided in each module handbook.