<table>
<thead>
<tr>
<th>Module Code</th>
<th>CS3D4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Name</td>
<td>3D4 Operating Systems and Concurrent Systems</td>
</tr>
<tr>
<td>Module Short Title</td>
<td>N/a</td>
</tr>
<tr>
<td>ECTS weighting</td>
<td>5</td>
</tr>
<tr>
<td>Semester/term taught</td>
<td>Hilary Semester</td>
</tr>
</tbody>
</table>
| Contact Hours | Lecture hours: 33  
Lab hours: 7  
Tutorial hours: 11  
Total hours: 48 |
| Module Personnel | Lecturing staff: Dr Mike Brady |

### Learning Outcomes

When students have successfully completed this module they should be able to:
1. recognise standard concurrent programming problems;
2. solve concurrent programming problems using standard techniques;
3. design and implement concurrent programs using standard threading techniques;
4. develop models of concurrent programs using the Promela modelling language;
5. verify the correctness of simple concurrent programs using SPIN;
6. describe algorithms, data structures and policies used in modern operating systems for thread scheduling, memory management, disk I/O and file management;
7. evaluate, compare and reason about the relative performance of algorithms used by operating systems (e.g. page replacement policies);
8. apply a knowledge of operating system behaviour when developing user-level programs.

### Module Learning Aims

The first part of this module introduces students to concurrency and concurrent programming. The aim is to provide students with the ability to develop concurrent software systems using standard techniques and constructs.

To achieve this aim, students must have a thorough understanding of common problems that arise in concurrent systems and how these problems can be avoided. This module will teach the use of tools and techniques for modelling and verifying the correctness of concurrent systems, applying this through practical laboratory exercises in which small concurrent software systems are developed.

The second part of the module addresses various aspects of the design of modern operating systems. The main aim is to explore how programmers can apply a knowledge of operating system features to the design of efficient applications. This is achieved by examining common algorithms and policies used by modern operating systems, as well as the facilities provided to application programmers. This knowledge is then applied in laboratory exercises.

### Module Content

**Concurrency**
1. introduction to concurrency;
2. simple multi-threaded programs in Java and C;
3. modelling concurrent systems;
4. interference;
5. mutual exclusion;
6. critical sections;
7. verification of concurrent programs;
8. semaphores;
9. monitors.
### Module Descriptors 2016/17

**School of Computer Science and Statistics.**

<table>
<thead>
<tr>
<th>10. Operating Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>11. operating system architectures;</td>
</tr>
<tr>
<td>12. memory management;</td>
</tr>
<tr>
<td>13. processor scheduling;</td>
</tr>
<tr>
<td>14. disk I/O, file systems.</td>
</tr>
</tbody>
</table>

#### Recommended Reading List

- Bovet and Cesati, "Understanding the Linux Kernel, Third Edition", O'Reilly Media, 2005

#### Module Pre Requisite

A working knowledge of C/C++; an understanding of computer organisation.

#### Module Co Requisite

#### Assessment Details

- % Exam: 80
- % Coursework: 20

*Supplemental: Examination: 100%, Coursework: 0%*

#### Module approval date

N/A

#### Approved By

N/A

#### Academic Start Year

N/A

#### Academic Year of Data

N/A