Overview

Every course specifies a number of learning outcomes to help students studying the course. These are necessarily of a high level nature and arguably do not help students preparing for the course as much as we (the faculty) would like. Indeed, students may not even read them or link them with the material they are studying. Often, unfortunately, the learning outcomes can only be understood fully by someone who has already completed the course.

In order to make these learning outcomes more achievable, more comprehensible, and more accessible, we have been developing a new teaching technique based on an expanded, more fine-grained collection of achievements, which taken together encompass the course’s learning outcomes, but are more achievable when taken individually. These make more explicit what a student needs to understand or master in order to pass and also describes the deeper qualities required in order to gain a higher grade. The points of presenting a course using this approach include:

- making it easier for students to see not only which items are completed, but also which specific course objectives have been completed.
- putting more responsibility on students for their own education, while providing more freedom in how they embrace the subject.
- clarifying for the student what she actually can do thereby building both self-awareness and self-confidence.
- deviating from the model which in practice sends the message “great that you can design a program that does X” to a model in which the message is “great that you can understand design processes”.
- moving the course closer to problem-based learning, which builds stronger and more capable students (than traditional teaching does), by requiring more research and evaluation of information.
- making the examination a dialogue so that feedback occurs naturally.
- avoiding examination formats that do not match the task at hand (e.g., a written exam for a software design process)
- creating reasons for deeper discussion with knowledgable assistants beyond mere correction of assignments and troubleshooting.
ensuring that the effort you expend is both educational and personally satisfying.

The only way to learn how to design is by doing it, by developing your own designs and by reading the designs of others. We also hope that the course is educational on more levels than just systems design.

Our hope is that the system of achievements will make the course better than previous years, and lighten the workload. That said, this is a new system and it may experience some teething problems. As usual, we need your feedback in order to understand how it works and whether any changes need to be made.

Description of Achievements

This document describes the expanded achievements for the course Advanced Software Design. For each achievement, there is a specification of its level and an indication of how it links with course objectives. You are expected to demonstrate achievements continuously during the course. *Avoid an avalanche at the end.* To get grade 3 on the course, all the achievements at level 3 must be demonstrated, for grade 4, all the achievements at level 3 and 4 to demonstrate. For grade 5, demonstrate that you have learnt advanced design principles and finish the single achievement of level 5. **This achievement is done individually and group work is not allowed.** Note that achieving higher levels does not entail doing more work *per se*, as many achievements can be ticked off at the same time. So, be clever about planning your efforts. Note that you may not pass achievements initially, but you can retry them as many times as required, within the time-frame of the course. This is inevitable, as you can hardly be expected to be able to design software well without first learning how to distinguish good and bad design.

The following are the course objectives, coded from IOL-A–IOL-F. These codes appear next to each achievement to indicate how they relate to the course objectives.

**ILO-A** Summarise and explain relevant design principles.

**ILO-B** Use an object-oriented methodology to design and implement larger programs.

**ILO-C** Use a modelling language as a means to communicate realistic problems and their solutions.

**ILO-D** Demonstrate understanding of the connection between modelling languages and programming languages, for example by implementing design models.

**ILO-E** Use design patterns and other known solutions to design problems.

**ILO-F** Evaluate the suitability of different design alternatives based on object oriented design principles, and identify design flaws in programs.

**ILO-Extra** Conjunction of ILO-A, ILO-B, ILO-C, ILO-D and ILO-F.

Each achievement has the form:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>A description of the achievement.</td>
<td>L</td>
<td>ILO-O</td>
</tr>
</tbody>
</table>

**Criteria.** The assessment criteria.

**Documents.** The documentation you need to provide.
**Level** represents the grade level of the achievement.

**IOL** represents the corresponding intended learning outcome.

The following table captures how the achievements are distributed, both across the different grade levels and per course objective.

<table>
<thead>
<tr>
<th>Course Objective</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>Extra</th>
<th>P</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>Level 4</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>Level 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>21</td>
</tr>
</tbody>
</table>

*P* indicates *procedural objectives* which are in place to ensure timely completion of design tasks.

**Meta-instructions**

The remainder of this document describes all achievements. Please make a clear distinction between the design tasks you need to perform and these achievements: the specification of the project (the other document) describes what needs to be designed, whereas the achievements (this document) are what need to be demonstrated to pass the course. The project provides the material upon which to base your demonstrations.

An important insight that you must reach in order to cope with this course is that the achievements are heavily related to the design tasks, and that some are easier if they are solved together. Ultimately, it should be clear that a higher grade is not about quantity, but about synthesis and deeper levels of insight (quality).

Here are a number of tips to guide you:

1. Develop a plan for how to address the achievements, globally. When is the best time to demonstrate a particular achievement?

2. Develop a plan for each achievement. Without a plan, you may not be allowed to demonstrate your mastery. An important part of the plan for ticking off an achievement is determining who will say what — if one person says all that there is to be said, it will be difficult for the others to demonstrate the achievement. Similarly, if you say nothing, then we cannot give you the achievement.

3. Try to **group related achievements together** in order to address them together. This will be particularly helpful when you are going for a higher achievement that encompasses a number of lower achievements.

4. You need to convince the TAs that you have enough mastery to tick off the achievement. The TAs will not (indeed cannot) extract the knowledge from you.

5. Try to address achievements as early as possible in the course, to avoid last minute rush (and panic).

**Achievements can be demonstrated individually or as a group, so long as all members present demonstrate appropriate mastery. Silence will earn no credit. Absent group members earn no credit.**
Final Deadline

The last date to tick off achievements of level 3 and 4 is 18th of December, 2017
Do not leave everything until the last minute.
The Achievements

A  Design Principles

A1  Goal  Level  ILO
Illustrate the GRASP principles using concepts from your design.  3  ILO-A

Criteria. Answer correctly identifies each GRASP principle using concepts from the system under design.
Documents. Some relevant diagrams, such as class diagrams, possibly independent of your ongoing design.

A2  Goal  Level  ILO
Explain how GRASP design principles have been applied in the system under design, using compelling examples.  4  ILO-A

Criteria. Answer correctly applied GRASP principles to the existing design. Note that your design needs to be sufficiently sophisticated before you can attempt this achievement.
Documents. Class diagrams from the system-under-design.

B  Domain Modelling

B1  Goal  Level  ILO
Construct a valid domain model of the core elements of the software under design.  3  ILO-C

Criteria. Answer includes the most important domain elements and most important relationships between them, with few spurious elements and relationships, for Level R of the specification.
Documents. Domain model.

B2  Goal  Level  ILO
Construct a detailed domain model of one of the configurations of the software under design.  4  ILO-C

Criteria. Answer includes all important domain elements and the relationships between them, for Level B of the specification.
Documents. Domain model.
C Requirements and Software Architecture

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td></td>
<td>ILO-B</td>
</tr>
<tr>
<td>Compare and contrast functional and non-functional requirements using examples from the system-under-design, and discuss the purpose of software architecture.</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Criteria. Answer concisely and correctly identifies various functional and non-functional requirements, and formulates a general characterisation of the difference between the two. Answer also concisely accounts for why it is beneficial to consider software design from an architectural perspective.

Documents. A short document listing the relevant non-functional requirements for the system-under-design.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td></td>
<td>ILO-B</td>
</tr>
<tr>
<td>Analyse the non-functional requirements of the software system under design and produce an appropriate software architecture for the most pressing requirements.</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Criteria. Answer correctly identifies the non-functional requirements for the system under design, without any spurious requirements, and produces a software architecture in terms of class diagrams and component diagrams, which addresses those requirements.

Documents. Class diagrams and Component diagram. Optional: other architectural design notation.

D Design

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td>ILO-C</td>
</tr>
<tr>
<td>Construct appropriate class models of the core of the software under design.</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Criteria. Answer includes the most important ingredients using syntactically valid UML and covers all core elements. (Level R.)

Documents. Domain model, Class diagrams.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>D2</td>
<td></td>
<td>ILO-C</td>
</tr>
<tr>
<td>Construct comprehensive class models of one end-to-end configuration of the software under design.</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Criteria. Answer includes the most important ingredients using syntactically valid UML, and covers all the subsystem in detail. (Level B.)

Documents. Domain model, Class diagrams.
D3  
**Goal**
Provide behavioural models that detail the basic transitions within the system.

**Criteria.** Answer includes behavioural models of the core part of the system. ([**Level R.**](#))  
**Documents.** Class diagrams, Behavioural models such as sequence diagrams.

D4  
**Goal**
Construct comprehensive behavioural models for one end-to-end configuration of the system under design.

**Criteria.** Answer include behavioural models of the subsystem in detail. ([**Level B.**](#))  
**Documents.** Class diagrams, Behavioural models such as sequence diagrams.

E  **Implementation**

E1  
**Goal**
Demonstrate the connection between diagrams and code, motivating any deviations between design and code, using examples from the system under design.

**Criteria.** Answer identifies UML elements and corresponding code in the language of your choice, without spurious connections, covering classes, inheritance, encapsulation, associations, etc. and behavioural models. Differences are accounted for concisely and correctly. The program does not need to run, but it should compile without any warning or errors (preferably a static language)  
**Documents.** Class diagrams, Behavioural diagrams and Code. Include a Makefile to easily compile your code.

F  **Design Patterns**

F1  
**Goal**
Identify 2 or 3 non-trivial design problems within the system under design which could be solved using non-trivial design patterns.

**Criteria.** Answer correctly identifies design problem and which pattern(s) could address the problem, without playing pattern bingo, that is, without guessing and hoping for the best. Trivial patterns such as singleton and wrappers cannot be considered.  
**Documents.** Documents with concise explanation.

F2  
**Goal**
Apply the design patterns to the problems identified in F1.

**Criteria.** Answer correctly applies design patterns without introducing spurious elements, and justifies the choices made (justify the selected pattern, justify how applied). Check with your TA that you’ve selected a sufficiently complex design problem.
Documents. Class diagrams, Behaviour diagrams (sequence diagrams).

G Design Improvement and Refactoring

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>Incorporate substantial new use case into a design.</td>
<td>3</td>
</tr>
</tbody>
</table>

Criteria. Answer shows design documentation before and after the change, highlighting where changes have been made. New use case must be non-trivial, and changes correctly capture the new requirements.

Documents. Domain model, Class diagrams, Behavioural diagrams. Description of use case.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>G2</td>
<td>Refactor design to improve it to overcome identified weaknesses. Weakness will be determined in conjunction with TA.</td>
<td>4</td>
</tr>
</tbody>
</table>

Criteria. Answer includes both original and refactored design with refactoring steps clearly noted. Each refactoring is justified according to the design principles, and the refactoring must arguably be an improvement in the design.

Documents. All design documents.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>G3</td>
<td>Estimate how a request for changes to the software-under-design will affect it and develop a plan for implementing those changes. Requirements will be provided by TA.</td>
<td>4</td>
</tr>
</tbody>
</table>

Criteria. Answer is realistic and identifies most or all of the places in which the software needs to change, including any refactoring required, and what those changes entail. All proposed changes are justified based on design principles. Plan is well-thought-out. Design needs to be sufficiently advanced before attempting this.

Documents. All design documents.

H Design Review

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>Objectively review own final design. Identify weaknesses and reason about completeness of various use cases. Use GRASP and walkthroughs.</td>
<td>3</td>
</tr>
</tbody>
</table>

Criteria. Answer systematically and concisely applies GRASP principles and walk-throughs on own design and identifies weaknesses and justifies whether use cases are complete. You will need to clearly give the use cases being considered, using the project description as a basis.

Documents. All design documents and a report that concisely identifies weaknesses and strengths of your design. Link this report with the design documents.
### Goal: Assess the quality of another group’s design in relation to the specification. Identifying both positive and negative attributes based on GRASP principles, walk-throughs, and other methods.

<table>
<thead>
<tr>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>ILO-F</td>
</tr>
</tbody>
</table>

**Criteria.** Answer systematically and concisely applies GRASP principles and walk-throughs on other design and identifies some weaknesses and justifies whether some use cases are complete or not. At least 3 non-trivial use cases are walked-through and 4 GRASP principles are applied. (This achievement can be demonstrated only in week 50).

**Documents.** Design documents from another team to be provided.

### I Teamwork

#### Goal: Enumerate difficulties your team has encountered throughout the course and propose reasons for those difficulties.

<table>
<thead>
<tr>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>ILO-B</td>
</tr>
</tbody>
</table>

**Criteria.** Answer concisely identifies a number of difficulties and provides a reasonable account of the cause of them (without attributing blame). **This can only be done towards the end of the course.**

**Documents.** None.

### J Individual work

#### Goal: Construct appropriate domain, class and behaviour model for an application of your choosing. Consider the functional and non-functional requirements (agreed with your TA) and produce an appropriate software architecture. This achievement can be delivered until the 8th of January 2017 upon agreement with TA.

<table>
<thead>
<tr>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>ILO-Extra</td>
</tr>
</tbody>
</table>

**Criteria.** Answer concisely identifies important domain elements, classes, behavioural models, software architecture and applied design patterns. The student can continue working in the current design if so desired, extending the design with **Level X** features.

**Documents.** Class diagrams, Behavioural models, Architectural design, Functional and Non-functional requirements and explanation of used design patterns.
K Deliverables

<table>
<thead>
<tr>
<th>K1</th>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Produce comprehensible documentation of design and modelling artefacts, and bundle these together in a final deliverable.</td>
<td>3</td>
<td>ILO-P</td>
</tr>
</tbody>
</table>

Criteria. A significant volume of design documents are submitted to the TAs in time for the review. Documents. All design documents.

Total number of achievements 21.