Advanced Software Design: Achievements

Dave Clarke

October 24, 2014

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 objectives 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 objectives, but are more achievable when taken individually. These make it 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 goals 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.

• to put more responsibility on students for their own education, while providing more freedom in how they embrace the subject.

• to clarify for the student what she actually can do thereby building both self-awareness and self-confidence.

• to deviate 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”.

• to move 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.

• to make the examination a dialogue so that feedback occurs naturally.

• to avoid examination formats that do not match the task at hand (e.g., a written exam for a software design process)

• to create reasons for deeper discussion with knowledgable assistants beyond mere correction of assignments and troubleshooting.
to ensure 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. I also hope that the course is educational on more levels than just systems design.

My 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 demonstrate, for grade 4, all the achievements at level 3 and 4 to demonstrate and for grade 5, all achievements must be demonstrated. 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. Also note that achievements can be retried as many times as required, within the time-frame of the course.

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.

Each achievement has the form:

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

Criteria. The assessment criteria.
Documents. This describes the documentation you need to provide.

Level represents the grade level to which the achievement corresponds.

IOL represents the course’s intended learning outcome to which it corresponds.
The following tables capture how the achievements are distributed, both across the different grade levels and per course objective.

<table>
<thead>
<tr>
<th>Level</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>14</td>
<td>11</td>
<td>5</td>
</tr>
</tbody>
</table>

<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>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

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

**Meta-instructions**

The remainder of this document describes all achievements. Please make a clear distinction between the 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 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). Without these, it becomes impossible to demonstrate all the achievements required for a 5.

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 demonstrate 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 is 19 December, 2014.*

*Do not leave everything until the last minute.*
The Achievements

A   General

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describe the objectives of software design and how different design</td>
<td>3</td>
<td>ILO-A</td>
</tr>
<tr>
<td>methods (such as the waterfall and iterative methods) contribute to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>achieving those objectives.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Criteria.* Answer gives a clear and concise overview of relevant objectives, distinguishes waterfall and iterative development methods, and describes how each of them contributes to achieving good software design.

*Documents.* None.

B   Design Principles

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summarise design principles including separation of concerns,</td>
<td>3</td>
<td>ILO-A</td>
</tr>
<tr>
<td>information hiding, coupling and cohesion, and encapsulation.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Criteria.* Answer correctly and concisely defines the various concepts. Note that a simple example will significantly help when explaining abstract concepts.

*Documents.* None.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Illustrate the GRASP principles using concepts from your design.</td>
<td>3</td>
<td>ILO-A</td>
</tr>
</tbody>
</table>

*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.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explain how the design principles from the previous points have been</td>
<td>4</td>
<td>ILO-A</td>
</tr>
<tr>
<td>applied in the system under design, using compelling examples.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*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.* Relevant diagrams from the system-under-design.
C  Domain Modelling

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1 Construct a valid domain model of the core elements of the software under design.</td>
<td>3</td>
<td>ILO-C</td>
</tr>
</tbody>
</table>

*Criteria.* Answer includes the most important domain elements and most important relationships between them, with few spurious elements and relationships.

*Documents.* Domain model.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2 Construct a valid domain model of a substantial subset of the software under design.</td>
<td>4</td>
<td>ILO-C</td>
</tr>
</tbody>
</table>

*Criteria.* Answer includes all important domain elements and several extensions and the relationships between them.

*Documents.* Domain model.

D  Requirements and Software Architecture

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1 Compare and contrast functional and non-functional requirements using examples from the system-under-design.</td>
<td>3</td>
<td>ILO-B</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.

*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>D2 Discuss the purpose of software architecture.</td>
<td>4</td>
<td>ILO-B</td>
</tr>
</tbody>
</table>

*Criteria.* Answer concisely accounts for why it is beneficial to consider software design from an architectural perspective.

*Documents.* None.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3 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>ILO-B</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 appropriate architectural design notations) which addresses those requirements.

*Documents.* Class diagrams. Component diagram.
<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4</td>
<td>5</td>
<td>ILO-B</td>
</tr>
</tbody>
</table>

Determine potentially conflicting requirements and propose means for resolving such conflicts. **Possible new requirements will be provided by TA.**

**Criteria.** Answer identifies the conflicts, concisely identifies the cause(s) of each conflict, and proposes a number of design trade-offs for resolving the conflicts in various ways.

**Documents.** Design documents.

### E Design

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>3</td>
<td>ILO-C</td>
</tr>
</tbody>
</table>

Construct appropriate class models of the core of the software under design.

**Criteria.** Answer includes the most important ingredients using syntactically valid UML and covers all core elements.

**Documents.** Domain model, Class diagrams.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2</td>
<td>4</td>
<td>ILO-C</td>
</tr>
</tbody>
</table>

Construct comprehensive class models of the software under design.

**Criteria.** Answer includes the most important ingredients using syntactically valid UML, and covers all core elements and some extensions in detail.

**Documents.** Domain model, Class diagrams.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>E3</td>
<td>3</td>
<td>ILO-C</td>
</tr>
</tbody>
</table>

Provide sequence diagrams that detail the basic transitions within the system.

**Criteria.** Answer includes behavioural models of the core part of the system.

**Documents.** Domain model, Class diagrams, Behavioural models such as sequence diagrams.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4</td>
<td>4</td>
<td>ILO-C</td>
</tr>
</tbody>
</table>

Construct comprehensive behavioural models for the system under design that includes a sequence diagram for the complete workings of a feature such as combat, movement, trading etc and show how fits in with the rest of the design.

**Criteria.** Answer include behavioural models of the core parts of the system and several extensions in detail, and includes a top-level behavioural model linking such models.

**Documents.** Domain model, Class diagrams, Behavioural models such as sequence diagrams.
F  Implementation

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demonstrate the connection between class diagrams and code, and motivate any deviations between design and code, both using examples from the system under design.</td>
<td>3</td>
<td>ILO-D</td>
</tr>
</tbody>
</table>

Criteria. Answer identifies UML elements and corresponding code in the language of your choice, without spurious connections, covering classes, inheritance, encapsulation, associations, etc. Differences are accounted for concisely and correctly.
Documents. Class diagrams and Code.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demonstrate the connection between behavioural diagram and code, and motivate deviations between design and code, both using examples from the system under design.</td>
<td>3</td>
<td>ILO-D</td>
</tr>
</tbody>
</table>

Criteria. Answer identifies UML elements and corresponding code in the language of your choice, without spurious connections, covering several behavioural models. Differences are accounted for concisely and correctly.
Documents. Class diagrams, Behavioural diagrams and Code.

G  Design Patterns

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify 2 or 3 non-trivial design problems within the assignment which could be solved using non-trivial design patterns.</td>
<td>3</td>
<td>ILO-E</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply the design patterns to the identified problems after approval of TA.</td>
<td>4</td>
<td>ILO-E</td>
</tr>
</tbody>
</table>

Criteria. Answer correctly applies design patterns without introducing spurious elements, and justifies the choices made (justify the selected pattern, justify how applied).
Documents. Class diagrams, Behaviour diagrams (sequence diagrams).

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply design patterns to a large a scope as possible in order to keep the design simple and supple, without duplicate functionality, and employing significant amounts of reuse.</td>
<td>5</td>
<td>ILO-E</td>
</tr>
</tbody>
</table>

Criteria. Answer correctly applies design patterns without introducing spurious elements, justifies
choices made (as above), and demonstrates where reuse occurs and illustrates the flexibility of the
design, for instance, by dealing with new variants of existing concepts. Answer applies design pat-
terns beyond those discussed in class.

Documents. Class diagrams, Behaviour diagrams (sequence diagrams).

H  Design Extension and Improvement and Refactoring

<table>
<thead>
<tr>
<th></th>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>Incorporate substantial new use cases into a design.</td>
<td>3</td>
<td>ILO-B</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></th>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
</table>
| H2  | Refactor design to improve it to overcome identified weaknesses Weak-
ness will be determined in conjunction with TA.                    | 4     | ILO-F |

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></th>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
</table>
| H3  | Estimate how a request for changes to medium-sized piece of software
will affect the product. Requirements will be provided by TA.         | 4     | ILO-F |

Criteria. Answer is realistic and identifies where the software needs to change. Changes need
only be identified, not made. Design needs to be sufficiently advanced before attempting this.

Documents. All design documents.

<table>
<thead>
<tr>
<th></th>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
</table>
| H4  | Develop a project plan for renewing a medium-sized piece of software
on the basis of a request for change Requirements will be provided by
TA.                                                        | 5     | ILO-F |

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.

Documents. All design documents.
I  Design Review

<table>
<thead>
<tr>
<th>ILO</th>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>Objectively review own final design. Identify weaknesses and reason about completeness of various use cases. Use GRASP and walk-throughs.</td>
<td>3</td>
<td>ILO-F</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.

<table>
<thead>
<tr>
<th>ILO</th>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2</td>
<td>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.</td>
<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. Group needs to talk for at least 10 minutes.

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

<table>
<thead>
<tr>
<th>ILO</th>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3</td>
<td>Compare and evaluate two different designs. Identify design choices made differently and discuss pros and cons of alternatives. Identify parts missing from each design and evaluate how the designs could be combined.</td>
<td>5</td>
<td>ILO-F</td>
</tr>
</tbody>
</table>

**Criteria.** Answer analyses the complete designs of the two different groups and identifies differences, evaluates comparative strengths and weaknesses, focussing on the most important design decisions. Answer correctly identifies parts missing from one of the designs and evaluates how well the part from the other design could fill the gap, by identifying weaknesses and hooks where parts of the two designs could combine.

**Documents.** All design documents of yours and the other team’s projects.

J  Teamwork

<table>
<thead>
<tr>
<th>ILO</th>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Enumerate difficulties your team has encountered throughout the course and propose reasons for those difficulties.</td>
<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.
Assume that your team was to continue working together until the completion of the system under design. Pick 3 ways of restructuring your team to address the difficulties your team has encountered and weigh these up against each other.

Criteria. The three ways identified are sensible, distinct and well-justified. Valid connections between the problems of the group and these three ways are provided. A well-reasoned comparison between the three approaches is made, leading to a conclusion about which one is best. Look at the document about Group Roles available on the course web page for inspiration.

Documents. None.

**K Deliverables**

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>K1</strong> 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 before the end of the course.

Documents. All design documents.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Level</th>
<th>ILO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>K2</strong> Submit final deliverable on time (as stated in assignment description).</td>
<td>4</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 30.

**Tips**

- **Construct a domain model first.**
- Build a prototype soon. If stuck, start coding.
- Divide design work amongst group members.
- Think about architecture early. Will give a useful overview. Details of deployment (servers, etc) should be delayed until after detailed design has been done.
- Group achievements together to reduce overall workload.