CourseForges: Open Source Curriculum Design for Value-Based Software Engineering

Mary Shaw, Carnegie Mellon University
Shawn Butler, Carnegie Mellon University
Hakan Erdogmus
Klaus Schmid

Available at: https://works.bepress.com/mary_shaw/10/
CourseForges
Open Source Curriculum Design for Value-Based Software Engineering

Mary Shaw, Shawn Butler, Hakan Erdogmus, Klaus Schmid
School of Computer Science Carnegie Mellon University
+1-412-268-2589 <etc>
http://www.cs.cmu.edu/~shaw
mary.shaw@cs.cmu.edu

Abstract
As a relatively young discipline within software engineering, value-based software engineering does not yet have an established curriculum. The area draws on models and techniques in so many other disciplines that it is likely to be some time before a single individual is ready to prepare a course or a textbook. Several of the EDSER-4 participants expressed interest and enthusiasm for sharing the effort of developing curriculum and course materials. Inspired by the success of open source software development, especially the distributed collaboration, the free public access to the results, and the lack of administrative overhead; we decided to try to establish a similar community for curriculum development. This report describes progress to date, with emphasis on the community standards for cooperation and sharing.

Keywords
Value-based software engineering education, cooperative curriculum development, open source curriculum development, value-based body of knowledge.

1. Background
Course Forges was initiated at the EDSER 4, the 2002 Workshop on Economics Driven Software Engineering Research. Many EDSER participants want to add cost considerations to our software courses, but we don't see near-term prospects for a unified textbook. Further, we all have different expertise in the area. To complicate matters further, most of us don't have the opportunity to add a full course to our institutions' curricula. We decided that we could help each other by sharing the effort, with different people designing teaching units covering from one lecture to a few weeks' content. Two other communities have tried to establish a similar community for curriculum development. This page is the current draft of our shared values, principles and standards. It includes a declaration of principle, or shared intent and a discussion of rights and responsibilities of members of the community.

2. Community Values
Members of a collaborative community expect to share effort and benefits. This page is the current draft of our shared values, principles and standards. It includes a declaration of principle, or shared intent and a discussion of rights and responsibilities of members of the community.

2.1 Overview
We agree in principle to adapt the open source software development model for our purposes. This implies

- Collaborative development
  - Shared development, without an ego stake in authorship
  - Shared documents in a common, relatively public place
- Community standards
  - Shared development effort, with recognition as the chief incentive
  - Intellectual property ground rules encouraging sharing, with public content on this site and the possibility of extending the work for profit in other venues
  - Consistent structure for curriculum units, in the style of the patterns community

2.2 Declaration of Shared Intent
- Content: We are jointly interested in developing a curriculum for value-based software decision making, that is, for software design and development in which the significance of cost is on a par with that of functionality.
- Collaboration: We would like to share the effort of developing and possibly of offering this material. To this end we need a collaborative environment for developing and distributing the material, so we adapt the open source software model to work for curriculum materials.
• **Adaptability:** Our specific institutional requirements will lead us to courses that differ in detail, so we see the greatest promise in creating a set of short curriculum components that can be combined in different ways. Given the relative youth of this area, it is especially important to have curriculum components that can be incorporated in established courses.

• **Format:** We believe that a lecture-based format is often inadequate for this material. Formats with greater student engagement, such as projects and case studies, are usually more appropriate. Such materials also have greater promise for asynchronous shared offerings.

• **Audience:** The principal target audience is advanced undergraduates and early masters students; some of the components may fit in sophomore-level software engineering courses as well.

• **Resources:** Whenever possible, we would like to rely on external resources, including case studies and open source development tools.

### 2.3 Rights and Responsibilities

Legal obligations, especially with respect to intellectual property, are expressed in the license terms. This draft of license terms is modeled on the Open Source Initiative's Open Source Definition for code:

- **Free redistribution:** Material may be redistributed, by anyone, including as part of a larger redistribution, in printed or electronic form. Fees may not be charged for this redistribution, other than reasonable reproduction costs.

- **Public originals:** If derived forms (e.g., object code of tools) are distributed, the original form (e.g., source code) must be easily available as well. The original form is the preferred form for further development.

- **Free evolution:** Modification and derivative works are encouraged. They must be redistributed in accordance with this license, without requirement for additional licenses. Redistribution may not discriminate against people, groups, or fields of endeavor.

- **Noncontamination:** If this material is distributed with other material, it may be separated from the package for further redistribution. However it does not "contaminate" the other material. (e.g., if incorporated in a textbook, the rest of the book can be copyrighted by the author, but not the incorporated material -- and the difference must be clear)

This license is very similar to the Creative Commons ([http://creativecommons.org/](http://creativecommons.org/)) Attribution-Noncommercial License ([http://creativecommons.org/licenses/by-nc/1.0](http://creativecommons.org/licenses/by-nc/1.0))

We also recognize moral obligations. The academic community operates on credit and attribution, especially for promotions and other recognition. Showing influence on other institutions' curricula can be significant at some schools. Therefore we strongly encourage anyone who uses our materials to:

- **Acknowledge:** Record the use of material from a Forge at the obvious place in that Forge. Say what course and institution, what level and how many students, evaluation

### 3. THE COURSE FORGE

Just as the SourceForge web site provides a development environment for many independent open source software development projects, we intend the CourseForges site to provide a development environment for many independent cooperative curriculum developments. We begin with one Forge -- for value-based software engineering.

#### 3.1 The Web Site

We began with a conventional web site with discussion areas for public comment and other pages edited by a few core people. This site, [http://courseforges.org](http://courseforges.org), contains some discussion of tooling requirements and community values.

We have recently decided that the ability for participants to update the site easily is, at least for now, more important than sophisticated structure or layout. Accordingly, active development, of the organization, of content outlines, and of individual curriculum units, is now taking place on a Wiki at [http://seg.iit.nrc.ca/yawc/courseforges/public/wiki.cgi](http://seg.iit.nrc.ca/yawc/courseforges/public/wiki.cgi)

While we are a small community, simple password protection is sufficient -- anyone who can edit anything can edit everything. We rely on good will, change logs, and the Wiki's built-in version management to keep things under control. This should suffice until the materials are adopted, or even considered for adoption, by people who are not developers. At that time we will need either more sophisticated security or open discussion groups, and we'll move to the open source model in which everyone can review materials, download content, and discuss changes -- but only credentialed people can actually make changes.

#### 3.2 The CourseForges Alliance

The CourseForges Alliance is a group of software engineering researchers committed to collaborative development of curriculum material in the same spirit of sharing that is demonstrated on the Open Source Software development community.

Anybody can browse the pages or download the documents on this site. Members of the Course Forges Alliance may create new pages, upload files and images, and modify existing pages. Password protection enforces this restriction.

The friends of the Alliance encourage and support the activity, and they may use the materials, but they are not actively involved.

### 4. COURSE CONTENTS

While the CourseForges Alliance is open for everyone to contribute, it is fundamentally grounded in the EDSER-community: a group of researchers who address software engineering predominantly from an economic point of view. Consequently, it is planned that the first full course material will be available for a lecture on the EDSER topics. Such a lecture does not yet exist, nor does there exist full agreement on what the topics should be covered would be.

#### 4.1 Pragmatic Considerations for CourseForges Course Development

When looking at the problem of world-wide course construction, we need to accept that due to differences in local situations, there will never be a single, generic set of course materials that can be arbitrarily used at each university. Rather, the local situation like
the available time for students, lectures the students previously took, the specific focus of the lecturer, and so forth, must be taken into account. This poses a need for adaptability when developing a course. Being software engineers, we recognize of course the similarity between this situation and the development of a reference architecture for a line of software products [1].

Thus, we need to define a customizable approach to developing such a course. The modules of the course must be scalable and, while consistently building on each other, we need to ensure mostly independence of the modules. One approach to achieve this could be to define a basic skeleton of topics that each course must cover (the basics of value-based software development). This base content could be supplemented by other parts that are by themselves not required by other parts. Thus, a module – respectively its content – should be tagged as either required or optional.

In order to enable the lecturers to tailor or replace modules according to their needs, and to reorder modules, we need to define what is it each module provides, i.e., the objectives of the module, as well as the specific concepts and methods it provides, so that other modules can build on this. In order to support the reordering and replacement, it is also important that each module makes known the specific preconditions it has on other modules.

4.2 Possible Content of an EDSER-Course

We are far from having a final definition of how an EDSER-course could look like. However, in order to provide focus to such a discussion, we provide here one specific curriculum proposal. In particular, we use this opportunity to point out the range of variability such a curriculum would still support.

A high-level structure of such a course could consist of the following four blocks:

- **Software Business**: This part focuses on providing the students with an understanding of the intrinsic relation between a product, its characteristics, and their relation to the market place. This would also provide the basics for financial and strategic analysis.

- **Value Models**: This part teaches the students the fundamentals of the different forms of value that are relevant in software development and provides them with the key concepts relevant to building value models of software engineering activities.

- **Decision Making**: A key part of value-based software engineering is the need to make decisions based on value tradeoffs. Thus, in this part decision making techniques are taught.

- **Applications**: A collection of examples should illustrate the main technologies taught in the course. These could be discussed either as a fourth part at the end of the course, or scattered throughout the lecture. For this reason, each example description should also describe the required student knowledge for its discussion.

While this schema could provide a common skeleton for all EDSER-courses, the individual instantiation would probably strongly vary in terms of the extent of their treatment of the various topics. We provide here a key list of some topics and point out some parts that could be left out (are treated superficially) in specific courses.

**Software Business**

The key topics in the software business part are the market-oriented viewpoint (TTM, cost, pricing, customer value, etc.) and the financial concepts (NPV, compound interest, etc.). In addition, this part could address technologies from marketing science in more detail (analysis of customer preferences) and the issue of business strategy (e.g., balanced scorecard approach).

**Value Models**

This would be the main part and probably also take the main time of such a course. The basis for any kind of model is to make certain aspects measurable, thus this part would start with the topic of metrics and measurement and would further discuss. Then different forms of models would be discussed along with model-building approaches. This point could also link back to the first part by discussing more complex topics from finance like options and decision tree analysis. Finally, aspects important to the development of models like simulation-based approaches and model validation would be discussed. A full list of topics could look like this (<opt> marking typical parts that could be left out):

- Metrics and measurement (types of metrics, GQM, etc.)
- Utility theory <opt>
- Metrics estimation (data elicitation/ gathering techniques) <opt>
- Model types (rule-based, quantitative, etc.) <opt>
- Model building approaches (regression models, CoCoMo, simulation models <opt>, DTA <opt>, QFD <opt>)
- Financial models (options, book-keeping methods) <opt>
- Model validation

**Decision Making**

Here, typical decision making techniques like the analytical hierarchical process (AHP) or other multi-attribute techniques would be discussed. Further, AI approaches to decision making (e.g., rule-based, case-based-reasoning, and so forth) could be part of this section.

**Applications**

The applications could of course be distributed across the various sections, wherever appropriate. However, in a specific applications section more voluminous topics could be addressed like the analytical methods for software design: the CBAM-approach, the approach by Sullivan et al. on the value of modularity, or constructive techniques like product line scoping or process optimization (e.g., Agile development). This section would mainly serve a better anchoring of the previous topics and would actually enable the students to understand the software engineering concepts they learned so far better.

5. INVITATION

For CourseForges to succeed, even within the value-based software engineering community, it needs to provide enough useful curriculum material that faculty find it worth the time to look for content there. If it's successful, it will attract other new authors. The challenge for us is to bootstrap the activity so that it has a chance of achieving critical mass.

So this is an invitation to participate in the refinement of community values, the structure of the Wiki, and the development
of curriculum materials. Contact any of the authors or visit the CourseForges Wiki for further information.

6. ACKNOWLEDGMENTS

The CourseForges Wiki is hosted by the National Research Council of Canada at http://seg.iit.nrc.ca/yawc/courseforges/public/wiki.cgi. The static web site http://courseforges.org is hosted by Carnegie Mellon University. This work is supported by the National Science Foundation under Grant ITR-0086003, by the Sloan Software Industry Center at Carnegie Mellon, by the High Dependability Computing Program from NASA Ames cooperative agreement NCC-2-1298, by the Software Engineering Institute, under the Networked Systems Survivability Program., and in part by the Eureka 2023 Programme, ITEA project ip00004, Café

7. REFERENCES