Executive Summary:
We propose to radically transform the educational practices employed by UMD’s Computer Science Department. This transformation will allow our teaching faculty to teach more students, at lower cost, while providing a deeper, more personalized and more effective learning experience to all of our students, including women, underrepresented minorities, and those who come to our program with little or no computer science background.

Our approach employs the following key components:

**Mastery/Competency-Based Learning**: Introductory and general education courses do not follow a traditional single semester format. Students take these courses at their own pace and move on to later courses only after demonstrating mastery of prerequisite courses. Courses are redesigned to clarify the learning goals of each course.

**The Flipped Classroom**: Courses move rote information delivery out of the classroom and into online delivery platforms. Class time is used primarily for deeper discussion, hands-on exploration and collaboration.

**Advanced Technology**: We develop and use technology that helps faculty better manage, execute and analyze hands-on, in-classroom activities.

**Evaluation and Iterative Improvement**: We collect data about our current educational performance to establish a baseline against which our proposed work can be compared. We will conduct ongoing assessments to determine whether or not our proposals are improving educational outcomes and to provide suggestions for refinements.

Explanation of Proposed Effort:

Computer Science is important to UMD, to our region, and to society. Employment opportunities are widespread and our students are in high demand. Attracted by these opportunities, the number of declared CS majors at UMD, now nearing 1200, is expanding rapidly. This growth, however, pushes faculty to adopt a one-size-fits-all teaching style that, while fine for certain students, leaves behind many others who need approaches that are a bit more suited to their specific needs. Similarly, very strong students often are not identified and challenged. Because institutional
financial models and budgetary constraints won’t allow us to solve these problems by simply hiring more people, the Computer Science Department is seeking to adapt through a better curriculum, better pedagogy, and better technology.

This effort is motivated in part by the recent explosion of online education in the form of MOOCs – or Massive Open Online Courses, such as those offered by Coursera, Udacity and EdX. We believe that these MOOCs will put additional pressure on our department, in part, because parents, students and employers will have much more information about the education “marketplace”, and will, therefore, necessarily reconsider the university’s value proposition.

To jumpstart this effort, Prof. Bederson has been leading a departmental self-evaluation of our educational product. This effort, which enjoys broad faculty participation, has uncovered several ways in which CS education can be improved. These include:

**Broadening participation:** As our numbers grow, we are called on to teach more and different students. This includes non-majors, other campus community members, and CS students with differing backgrounds, learning styles and outside work commitments;

**Building Community:** Most of us are in Computer Science, because we enjoy it and because we feel that we belong. We need to foster a greater sense of community and belonging in our students as well. This will require exposing students to professorial faculty earlier in the program, increasing undergraduate research opportunities, and by incorporating more student-to-student interaction in our classrooms;

**Increasing Relevance:** Students learn best when their work is significant to them. We want to make our offerings more relevant to students. We will do this by emphasizing hands-on experimentation, and by increasing our attention to building practical skills, such as advanced software development tools, collaboration & communication, time management, and more. We will also create course projects that are more connected to student interests.

**Deepening Understanding:** Given technology’s rapid change, we must prepare our students to continue learning long after they leave our program. Thus, we must better balance practical software development and systems skills with abstract thinking and problem solving. This is especially important in introductory courses where the technology can easily get in the way of deeper ideas. We will address this in several ways, starting by refocusing our introductory courses to emphasize computational thinking, while decreasing programming language details. We will also develop a “knowledge map,” that clearly identifies skills that students are expected to master in each class, and connect these skills between courses and throughout the curriculum.

Achieving these goals will require us to advance on several fronts. These include: 1) Moving some courses to a Mastery/Competency-Based approach; 2) Where appropriate, adopting a Flipped Classroom teaching model; 3) Using, creating and refining advanced technology to support the previous two fronts; and 4) Supporting
evaluation and iterative improvement. Each of these is described in more detail below.

**Mastery/Competency-Based Learning**

Currently, our courses are delivered in a rigid one-semester format. Students who need more time do poorly in this format, but are forced to march forward anyway. This is especially problematic for otherwise bright students with limited previous experience. In many cases these students are women and minorities, groups we have struggled to attract and retain. Not having mastered the previous class, they perform poorly again. The pattern keeps repeating with many students muddling through, but never really “getting” computer science – or leaving the major.

Mastery-based learning breaks the semester format, by specifying learning goals and allowing the students to proceed only once they have demonstrated mastery of the learning goals. This approach is one of the reasons for the broad interest in MOOCs – they let students spend as much (or little) time as needed at each phase of the course.

Notionally, this would work as follows. Starting with an initial introductory course, we clearly define regular milestones and assessments that define a level of mastery. But rather than having all students work in lock step for a semester and then take a final exam – students would work at their own pace, and at each assessment stage, they would either be given a pass, allowing them to move forward, or they will be given feedback as to why their work did not display mastery, and they will need to repeat that stage. In other words, rather than keeping time constant and varying grades, we will keep the grade approximately constant, and vary time. Students would be expected to complete the course in some time frame ranging from one month to one year. To prevent students from procrastinating, mastery courses will define a set of fine-grained milestones, representing a minimally acceptable rate of progress. That is, students may take different amounts of time to complete their the entire course, but they must continuously demonstrate forward progress on a finer time scale. Students who cannot meet these progress requirements and demonstrate mastery of the entire course within a year will be counseled against pursuing computer science further.

Creating a mastery-based structure at UMD is a significant challenge for numerous reasons. From an administrative standpoint, courses are currently set up to offer a fixed number of credits for a fixed amount of classroom time, and crucially, students are billed at a fixed rate for these credits. Changing this system in any significant way would have impact throughout the university. Students would have to be able to take ensuing courses after finishing earlier courses off the semester boundaries; faculty workloads would have to be reconsidered; and it is unclear how tuition would be charged if credits could be earned at significantly different time scales.

There are other challenges as well. If students can attempt a topic over and over, and different students are reaching milestones at different times, then it will become very difficult to assess them since it is very hard to develop numerous equivalent assessments. There is also the risk that students will attempt to game the system,
developing skills at passing assessments without actually mastering the material. And from a practical perspective, since the goal is to support students proceeding at different speeds, it will be necessary to support students at different levels in the class simultaneously.

Thus, while we are excited about the concept of mastery based education, we are fully cognizant of the practical challenges of deploying a solution in a traditional academic setting. Nevertheless, we have a number of ideas that we think will enable us to pursue mastery-based learning, and we believe it is important enough to explore doing so, even though we understand that this is somewhat risky, and the details of how we proceed are likely to change over time based on what we learn.

We have already spoken with University administrators in the Dean’s, Provost’s, and Scheduling offices and have received permission to move forward with a mastery based experimental course in the Fall 2013. An appendix to this proposal describes that class, but briefly, the course will be structured as follows. The course will be split into three 1-credit modules. Students will only be allowed to register for subsequent modules if they reach a sufficient level of mastery on the previous ones. The course will be “flipped”, with recorded video lectures and class time being used for discussion, working on projects, etc. Students will be allowed to progress at their own pace, only moving to subsequent modules when they are ready. At the end of the semesters, students will be allowed an “Incomplete” grade to finish a module the following semester.

**The Flipped Classroom**

Today, almost all our classes involve a professor lecturing to students. Projects and homework are then done out of class. While there are pros and cons to this approach, in our experience, backed up by the recent successes of MOOCs, we believe that much basic lecture content could easily be pushed online and supported with class forums, email Q&A, and other asynchronous interaction technology. This would open up class time for intensive, hands-on activities, for which face-to-face interaction with faculty and TAs is invaluable. This approach is referred to as blended learning or the flipped classroom. One other advantage of the flipped classroom is that it can more easily support the mastery-based course structure discussed earlier. If students are to progress at their own pace, we have to avoid the traditional “lock step” teaching model of a lecture-based class.

The basic approach for running flipped classrooms will be for faculty to pre-record their lectures and to, optionally, create lecture-related assignments. In our experience, recording lectures is time-consuming, especially the first few times you try it. Once done, however, the videos can be reused in repeat offerings and can, if desired, by used by other faculty teaching the same course. Students will be expected to watch the lectures online using an existing MOOC system. Some faculty may wish to use video presentations created by others. We believe that this will be most acceptable when the presentations serve as supplementary material in that video presentations are in some ways analogous to textbooks. Until we have more experience with flipped classroom teaching, however, our faculty will avoid the
wholesale adoption of outside video presentations as their primary lecture materials. Even in the long-run, we expect that faculty will carefully determine where and to what extent such materials fit into the design of their course -- the flipped classroom style does not relieve faculty of their responsibility and accountability for their teaching performance.

Faculty will also prepare in-class activities to be done in the normal class session. These class sessions will typically include activities, such as Q&A sessions related to the day’s lecture, demonstrations in which faculty show and explain fully working examples related to the lecture, laboratory exercises in which the students practice applying what they learned in the lecture, and practice exams or problem sets to test whether they've fully understood the material. Some in-class activities will be class-wide, and occasionally the instructor will want to get the whole class' attention to explain something that many students are having problems with. But students will also use the time to work by themselves or in small groups, and the instructor and TAs will help students as needed. While video presentations have received a great deal of attention, the fact is that students will increasingly have access to high-quality video materials, delivered by perfectly knowledgeable people. The real value added by flipping the classroom will only be realized when students truly feel that they lose something by missing class. Faculty will need to “up their game” in the classroom, experimenting with the right mix of whole-class, peer-structured, and independent activities. Faculty will also need to experiment with new kinds of attendance policies as well.

We note that we have not yet decided which MOOC platform we will use. While UMD is currently offering some courses with Coursera, that approach has some significant drawbacks – primarily that: 1) Coursera does not currently share student interaction data with the offerer, and so we would not be able to use the data to study and improve our offerings; 2) Their platform is closed source, and so we would not be able to do technical experiments innovating and improving the platform. Thus, we will at least strongly consider using an open source MOOC platform such as Google’s CourseBuilder or Stanford’s Class2Go systems. Prof. Adam Porter is currently videotaping the lectures for Programming Handheld Devices course and is experimenting with CourseBuilder and Class2Go. You can see his initial experiments with CourseBuilder at http://learn-android.appspot.com.

Advanced Technology

The mastery-based model and the flipped classroom approach put pressure on teachers and TAs who must manage multiple activities performed by large numbers of students. To support them we will use, modify and develop a variety of advanced teaching support technologies. Generally, these technologies fall into several groups: online lecture and assessment, student-to-student interaction, in-class tools and teacher dashboards, and web app support.

Online lecture and assessment: Teachers will develop online lecture materials typically in video format. Video segments will optionally be interspersed with assessment activities to ensure that students are engaging with the video and to
help the student and teacher measure whether students understand the material. We will address video production in more detail later when discussing our infrastructure needs. Initially, we expect to use YouTube-like services to host the video and tools like Class2Go or CourseBuilder to organize video clips and present and measure mini-assessment activities. Current assessment techniques are generally limited to individual students answering true/false and multiple-choice questions. In later phases of the project we will expand these services to support collaborative exercises and richer interaction formats and automatically generated question via “problets” (www.problets.org) or similar technology. One facility that we will support is A/B testing of instructor content. As teachers experiment with different ways to present the same concepts, we will support presenting two or more different videos to different students to compare which video produces the best success rate on follow-on assessment activities. The goal is to help teachers quickly hone in on the best ways to explain and present material. Similar technology will also be used to improve assessment activities as well.

**Student-to-student interaction:**

Currently many of our faculty use web forums such as Piazza. We will initially use these tools, but will expand on them to support more collaborative activities, such as code reviews, gamification- and reputation-building strategies to reward students who answer other student’s questions (similar to StackOverflow), and collaborative approaches to building, refining and storing databases of relevant examples and practice questions.

**In-class tools and teacher dashboards:**

We will develop tools to support collaborative, in-class activities, discussions and brainstorming. One example technology is called XParty (being developed by Bederson), which allows students to perform various activities in a web interface. The tool monitors the student’s actions and presents summary information and troubleshooting reports to the teacher to give him or her greater situational awareness in the classroom. We will also explore brainstorming and wargaming tools such as Mmowgli (portal.mmowgli.nps.edu), which allow moderated exploration of a problem and discussion and generation of problem solving approaches. For example, students in the networking course could be presented with a particular communication requirement and then be asked to create a network configuration that best meets the requirements. Tools like Mmowgli would allow students to generate ideas step-by-step, debate strengths and weakness, and flesh out action plans to realize their ideas in a highly visible and analyzable way.

**Web app support:**

To avoid problems with different student computing configurations, we will incrementally move our computing tools over time to be web or server based. Some such tools already exist, such as MOOCs for delivering content and our department’s “submit server” for giving students objective feedback on their software projects as they are developed. Many of the tools we are currently building, such as XParty, are entirely web based, and going forward, we will endeavor to find and build other
tools that similarly do not have any platform requirements. This is especially important for tools to be used in class where the diversity of the platforms students bring to the classroom is great.

**Evaluation and Iterative Improvement**

In order to ensure that our proposals are actually improving our educational offerings, we are building measurement and evaluation in from the start. We will perform a range of activities to assess, understand and describe our enhanced teaching activities with the goal of both improving our teaching as well as providing best practices and guidelines for others.

It is inherent in this kind of innovative teaching that there are countless variations in every offering. Whether it is the difference in the technology used, a teacher’s attitude, the student body, or any number of other externalities, it is exceedingly difficult to perform a controlled study to analyze the effectiveness of a pedagogical approach in our classes.

Instead, we will perform a range of qualitative analyses to understand our teaching. This will include benchmarking, surveys and questionnaires, regular student feedback, and classroom observation. Before we move a class to our new approach, we will start by gathering data showing how successful students are in current the course. We will then use this data as a benchmark against which to compare future students taking the course with new approach. We will also look at how students perform in ensuing courses by comparing students that finish a traditional approach to the course versus an innovative one.

But we believe the most important mechanism for assessment is through more traditional surveys, questionnaires and feedback. We will give students both directed and open-ended opportunities to describe their experiences in the course, looking at specific features of the course. For open-ended responses, we will perform “grounded theory” qualitative assessments, which are a scientifically appreciated form of analyzing open-ended feedback. In addition, we will incorporate the department’s traditional efforts in having faculty observe other’s classes and providing feedback.

Through this range of approaches, we will be able to triangulate and develop an informed understanding of which elements of the courses are working and where modifications and improvements are needed.

**Infrastructure**

Basic infrastructure support for creating and editing videos suitable for online lecture delivery is needed. While the minimal hardware requirements are modest (screen capture software, lighting, and a simple video camera), we need to be able to produce professional level material while minimizing faculty production time. We also recognize that we need to support a range of video styles to accommodate both individual faculty preferences as well as the different needs of the wide variety of subject matter that is covered by the department.
The department will commit to providing a standard sized small office for this purpose. We will buy equipment for this office that includes support for video production including pan and zoomable camera, lights, and backdrop. We will buy a desktop class computer with significant storage for video along with screencast and video production software. We will also buy at least one tablet computer suitable for high quality handwriting with screen capture (suitable for hand drawn figures and equations.)

We expect more student projects to need to be able to run on servers for purposes of continuous and collaborative operation. We also expect that in order to provide deeper infrastructure for projects, faculty will often want to control that infrastructure and not have to support multiple platforms. In addition, we expect that more faculty will want students to be able to write web apps. For all of these reasons, we plan on providing server space for student projects. This will require both server hardware as well as software infrastructure to support the security and isolation requirements of student projects. An example “private cloud” approach that we are considering is AppScale (https://github.com/AppScale).

Another area of infrastructure support that is important is in the design of our classrooms. As we move classes to be less lecture-based and students are more engaged in a variety of activities, we expect that modifications to some of the classrooms will be valuable. One of the obvious changes is that most of the classrooms in our classroom building have fixed row tables that are well-suited to lecture, but poorly suited to student group work. Reconfiguring that furniture to enable more flexible work situations while maintaining the number of students that can fit in a room is challenging, but is something that we believe is important. Similarly, we think it is likely that enhanced technology in the classroom is also likely to be important, but we defer a decision about what the technology might be until we get specific proposals from faculty about what they would like to do in the classroom.

**Implementation Plan**

This proposal represents a significant new departmental effort. This effort will be managed by an “educational transformation” committee, which itself will be chaired by two faculty members. The faculty chairs will each serve 2-year, overlapping terms (The first terms will be of differing lengths to get the phases alternating). Professors Ben Bederson and Adam Porter will be the initial heads. New committee chairs will be appointed by the department chair, based on input from the faculty.

These committee chairs will work closely with the department professorial and instructional faculty, and will develop communication and meeting practices as they see fit. But we expect that this will include broad meetings with some regularity to understand faculty needs, develop new approaches, and communicate best practices.

---

1 Campus now provides free copies of Adobe video products such as Premiere Pro, Lightroom, and AfterEffects.
The committee will support faculty training in several ways. For example, it will bring in external educational visitors, consultants, from both inside and outside the department and university, and acquire training materials as appropriate. The committee will arrange regular faculty gatherings, such as Brown Bag Lunches and presentations at the department’s annual retreat to showcase the department’s current educational offerings, to discuss best practices, to provide training, etc. Given the range of innovation and amount of change envisioned, the committee will also endeavor to engage educational specialists to more formally study our program to understand more rigorously the impact of these changes.

Starting Fall 2013, Professors Bederson and Porter will each teach a course using the new approaches described in this proposal. Bederson will teach a new introductory course for non-majors, and Porter will adapt a course that he has been teaching for some time on mobile software development. Both courses will use the flipped classroom approach, but only the introductory course will be mastery based. This is consistent with our general expectation that the mastery-based approach is initially better-suited to introductory courses. As we gain more experience with mastery-based courses, we will revisit this working assumption. Professors Bederson and Porter will document their experiences (i.e., in a blog or other report) in preparing and delivering these new courses in order to smooth the transition for other faculty members.

In general, we will roll out new courses and develop a comprehensive “knowledge map” of course requirements and outcomes incrementally – building them up over time. This will enable us to accommodate change as we learn from our efforts. Thus, the first year effort will enable us to both improve those courses, and to identify candidates for more courses to be updated.

We have been discussing with several faculty who are considering leading the second round of improved courses. In Spring 2014. After that, we will solicit faculty volunteers to lead the next improved course offerings. Our expectation is that courses with the most stable content will be the best candidates for early transition to a flipped classroom approach. Our goal is to have at least ½ of our regular courses using these approaches, where appropriate, after 5 years.

We will also reconsider the best way to manage the administration of mastery-based courses over time. As described, we will start by structuring the courses as 1-credit modules so that we can easily integrate them into the existing university administrative structure. But going forward, we will consider other administrative structures that might require changes at the university level (such as, for example, offering varying amounts of credit depending on how much material the student masters.)

A detailed budget is included which is based on assuming a total of $1m over five years. The bulk of the funding will go to staff support, to help produce the online teaching materials, and to graduate student effort, to build online teaching support systems and tools. In addition, there is support for media hardware and supplies, server hardware, and professional fees such as graphic design support. In addition,
there is modest support for conference travel to send the most engaged faculty and
students to relevant education conferences. Finally, there is funding to update our
classrooms to support modified teaching strategies which might include different
furniture and updated technology.

Appendix A: Example New First Course

One of our first new courses will be a redesigned introduction to programming for
non-majors. This new course will feature several new characteristics, building on
the themes described in this proposal. It will be mastery-based, and will be taught in
the flipped classroom format. Another key aspect of the course is that it will be
problem-based. That is, instead of focusing on the technology – assignment
statements, while loops, recursion, etc. – as many of our existing introductory
courses do, this course will focus on a single problem that is deep, fun, and relevant,
and that also requires students to learn the technologies mentioned above. By
breaking the problem down into manageable components, we will introduce the
relevant technologies and language features on demand, motivated by real need.

For this course, the problem will be to build virtual creatures subject to genetic
variation within an environmental simulation. The idea is to connect an outside
domain (basic biology and ecology) to motivate students by building something that
makes sense and connects to the real world. The creatures will have a simple
genome that can vary by standard genetic algorithms. The creatures will live in an
environment and have senses to perceive each other and the environment. They will
be able to act, such as by moving, performing actions, and affecting their
environment. This is all implementable with basic data structures, variables, loops,
conditions, functions and objects – in other the words, the same things we normally
teach in our introductory courses. Rather than simply run and stop, the programs
will run continuously in a server environment. This will enable an understanding of
a more modern view of computing that enables the introduction of considerations of
concurrency, security, and performance. Other characteristics of the course and
problem are:

- **Interactive & visual**: The students will focus on building the back end creatures
  and simulation, but we will provide a web-based front-end that enables a
  graphical view of the underlying simulation. This will make their algorithms
  come alive, but also provide a mechanism to directly interact with the creatures,
  enabling external input into the simulation. This interactive and visual approach
  will further enable the introduction of a more modern view of computing where
  user experience and interaction matter, along with back end systems that have
to deal with unknown input.

- **Real-world connections**: One of the things that makes computing exciting is that
even pure algorithmic work has real-world implications. Making the connection
between what is going on in a simulation and the physical world will help
students to better understand this relationship. To that end, we will provide
mechanisms to insert connections between the student simulations and real world. For example, students will be able to send text messages from their phone to their creatures – perhaps feeding them. And the creatures themselves will be able to reach out to the world by emailing, perhaps when they need help.

- **Mobile & Social application:** By providing an interactive and visual web-based front-end, the student projects automatically become mobile (since we will ensure that our provided web apps work on common phone and tablet browsers). And by making the projects mobile, students will be more likely to share what they are doing with their friends and family. They will be more likely to interact with their apps throughout the day. The projects will also be connected to social networking websites, enabling students to share them with their friends. Together, that will help students to develop a sense of pride and confidence.

Once the students have reached a milestone that shows that their creatures and simulation work to a certain level, they will be able to release their creatures into a course-wide simulation that we will provide. In this way, their creatures will be able to interact with other students' creatures – opening the door to discussion of complex systems, network effects, and unanticipated interactions.

- **Competitive and cooperative models:** Prof. Bederson previously developed a much simpler version of interaction among student programs in an introductory course. In Spring 2010, he built a “tournament server” which enabled student programs to compete against each other. There was automatic feedback and a real-time leaderboard along with the ability to see complete game play history. This turned out to be incredibly motivating to some students (those that liked competition) – but notably, there were many students (typically women and students that were not doing as well in the class) who were not motivated by this model at all.

Thus, we are designing this problem and framework to explicitly design for a wide range of motivations including competitive and cooperative models. By building in a wide range of tasks and metrics of success, we believe we can engage and motivate a wider proportion of the class.

- **Collaborative teamwork:** An ongoing challenge in our undergraduate courses is balancing the need for individual students to learn the material themselves vs. the benefit of learning from and being motivated by their peers. There is a growing literature on how peer learning can be effective [Lasry et al, 2008]. But at the same time, we are aware of the ease of copying code and other forms of cheating. We have the additional challenge that students will be completing different sections of the course in different time frames, and thus some students will have completed one phase of the course before others have even started it. Being aware of these tensions, we will include a balance of individual and group activities. The core “platform” that students build will be created individually
(and will be checked with plagiarism detection software). But at the same time, they will be encouraged to help each other – in and out of class. However, we will provide strict guidelines as to what kind of help is acceptable. We will also have occasional, smaller mini-assignments outside of the scope of the primary project that will be performed by assigned pairs to encourage more communication and awareness of various programming and problem-solving styles.

- **Not Java based:** We also will reconsider what language we initially teach. While Java is a fine language and certainly something that students should eventually learn, there are so many details that students must be facile with in order to write even the simplest program, that we find it is a significant stumbling block for students that don’t already have programming skills. We have not yet decided which language we will choose, but a front runner is Python which is becoming the language of choice in a number of other introductory course sequences. Among other things, it has very nice characteristics of having an interpreter so students can get immediate feedback well not requiring significant syntax and programming structures when only small operations are desired. It is well supported on servers including Google App Engine which supports free hosting and very easy deployment for small use projects. And it supports both very simple functional programming as well as sophisticated object-oriented programming. There are also very good freely available online references, tutorials, and tools to support learners.

This level of production will be challenging for incoming students with no programming experience, but we believe it is possible. The Udacity course on building a web search engine has already shown that even complex problems can be broken down into simple components that inexperienced students can handle. Similarly, we will break down this problem into many small components. We will also provide infrastructure to enable many of the seemingly hard parts. Finally, the mastery-based model will ensure that each student masters the level they are at before they move on. This is crucial as it solves the common problem in our existing courses where students have to move from one level to the next, even if they never fully understood the first level.

To summarize, this course structure will neither reduce what students learn, nor will it reduce the effort of faculty. The same technical topics will be covered as would be in our existing courses that are driven by programming language features. It is just that they will be introduced as needed, and will be motivated by driving problems rather than to “cover the language”. Instructors will experience a very different set of requirements, much as students will. They will still have to prepare lectures (via online technology), and will have to face the challenge of managing students progressing at different paces. They will also have to give high quality, subjective assessments that lets students understand clearly what is insufficient about their work if it does not meet the A standard.

The course will initially be taught at UMD, but all the software we write will be made open source, and all the instructional material will similarly be made open following
existing Open Educational Resource practices. Thus we will aim to both improve our own undergraduate curriculum and at the same time demonstrate to others how they can improve their teaching – and that UMD is a leader in innovative, technology-based teaching.