

## PROJECT SUMMARY

---

Successful engineers must be able to recognize and fill gaps in their education, work effectively in groups, and use past experience to predict future performance. In order to prepare a student to be successful engineers the university must teach them the set of scientific knowledge underpinning their particular discipline. In addition, it must also teach them to recognize holes in their knowledge and fill those holes, to work with others in both knowledge acquisition and artifact creation, and to plan their activities based on an accessible history of previous projects.

Translating these general educational objectives into specific classroom environments is a serious challenge. First, limitations of available design tools tends to constrain instructors to the use of relatively well-defined, simple problems that bear little resemblance the problems that are faced by practicing engineers. Second, engineering curricula typically do not include courses specifically intended to develop skills needed for group work. Since group projects are typically only one of the many components of existing courses, these projects are inadequate in teaching students the necessary collaborative skills.

Indeed, data from a compiler construction course at the University of Colorado suggests that students actively seek strategies to avoid genuine collaboration in their work, relying instead on dividing up the task or placing the whole responsibility on a minority of the group. When the class was extended to incorporate instruction in group interaction, students reported feeling overwhelmed at having to learn what they saw as two completely unrelated skill sets. This evidence suggests that one cannot simply assign students to work in groups and expect group skills to develop, regardless of whether or not instruction is provided. One needs to provide them with the opportunity to work on non-trivial problems in a complex environment.

This proposal describes a novel approach to addressing the above mentioned limitations in classroom implementation projects. The key idea is to involve students in realistic projects beginning with the introductory courses but to *use an infrastructure to selectively hide the complexity from the students*. As students progress to more advanced courses, more of the complexity is revealed, until finally, students are involved in realistic projects involving significant collaborative work. The implementation of this approach has two components: (i) an infrastructure that manages the complexity that is exposed to the students and (ii) a collection of tools that allow students to undertake larger projects than would normally be possible in a classroom setting.

The core of the infrastructure is a shared repository that maintains code and bug reports. An instructor may seed the repository with modules; students may compile against these modules or examine them using tools. A policy subsystem indicates what modules each student is allowed to access, in what form (source or binary representation), and when. For example, for an introductory class, an instructor may seed the repository with all the code for a project but only allow students to access the binary representation of the code. During the course of the project, students will replace one or more of the instructor's modules with their own code. Students can thus tackle larger projects (since at any time they can link to the instructor's code and have a working prototype) without the full complexity of a large project. Projects in more advanced classes will involve collaboration where a group of students must collectively replace the instructor's modules with their own code.

Besides traditional program development tools, three types of tools are available to students: (i) Analysis tools, such as program slicers, which students can use to understand and debug code; (ii) Abstraction tools, such as parser generators, which students can use to create robust implementations by writing compact declarative specifications; and (iii) Planning and reflection tools that capture information and insights about a project as it is being carried out and make them available as a basis for planning future work. Several of these tools exist as research prototypes. To our knowledge, this is the first time these ideas will be used for undergraduate teaching.

The ideas in this proposal will be incorporated into a three class sequence: Data Structures–Programming Languages–Compiler Construction. The tools and infrastructure developed as part of this proposal will be made publicly available.