Pair Programming and Project Management for CS2606

This semester, CS2606 will be using a programming methodology called Pair Programming (PP). We will also be overseeing the process by which you do your programming projects, probably to a greater degree than you are used to. You will be required to present a project schedule and make regular updates to that schedule. This document is an introduction to the various procedures that we will be using this semester.

1 What is Pair Programming?

Adapted from Wikipedia:

Pair programming (PP) requires two software engineers to participate in a combined development effort at one workstation. Each member performs the action the other is not currently doing: while one types unit tests, the other thinks about the class that will satisfy the test, for example. The person who is doing the typing is known as the driver, while the person who is guiding is known as the navigator.

Yes, that means you’re going to do your projects with a partner this semester. This is not optional, all students will work with a partner for all projects. We expect you to work with your partner throughout the project design, implementation, and testing phases. This is not a divide-and-conquer task. Like the Wikipedia definition says, one of you will drive while the other navigates. You should not break up the project, work on it separately, and combine your efforts later. Instead, you should take turns acting as a programmer, typist, and thinker versus a debugger, idea generator, and devil’s advocate.

Your partner will be selected for you by the instructor. You will change partners for each project in the class. This arrangement simulates how you might do PP in a corporate or research environment. Rarely in professional life can you choose your teammates. Teams will be announced roughly when the project spec is posted. See below for information about how changing teams affects code re-use.

For each deliverable, one member of the pair will make the submission. Only the entry with the latest (most recent) timestamp will be accepted, no matter who submitted it. It will lead to less confusion if your pair sticks to one person doing all submissions related to a given assignment.

2 Lab Hours

To facilitate teams working together on code, the TAs will hold office hours for 15-20 hours a week in McBryde 116/118. As a part of your grade, you must spend at least 2 hours a week in the lab, with your partner, working on the project in PP mode. Specifically, certain weeks during the semester will be designated as “lab weeks” during which weeks you and your partner must do a two-hour block in the lab. Available lab hours will be announced soon on the course website.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Task</th>
<th>Coder</th>
<th>Debugger</th>
<th>Tester</th>
<th>Deadline</th>
<th>Orig Est</th>
<th>Curr Est</th>
<th>Elapsed</th>
<th>Remain</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR Quadtree</td>
<td>implement nodes</td>
<td>JH</td>
<td>MC</td>
<td>JH</td>
<td>9/15</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>PR Quadtree</td>
<td>implement tree</td>
<td>MC</td>
<td>JH</td>
<td>JH</td>
<td>9/19</td>
<td>6</td>
<td>10</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Documentation</td>
<td>write README</td>
<td>MC</td>
<td>JH</td>
<td>-</td>
<td>10/2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Usage notes:**
- Your tasks should be measured in hours.
- Each feature should consist of several tasks.
- Pick fine-grained tasks.
- Update the elapsed column every time you work on the software.

Table and notes adapted from the “Painless Schedule Method” at http://www.joelonsoftware.com/articles/fog0000000245.html.

---

**3 “Painless Schedule”**

*I love deadlines. I like the whooshing sound they make as they fly by.*

—Douglas Adams

Another aspect of the PP experience will be keeping to a schedule. We have adapted Joel Spolsky’s “Painless Schedule Method” (see [http://www.joelonsoftware.com](http://www.joelonsoftware.com)) for this class. You will be required to submit, each week, an updated spreadsheet. This spreadsheet will show the various tasks required to complete the project. For each such task, there will be columns to indicate: name of task; indication of who is assigned to code, who to write tests, who to debug; real-time deadline for task completion; original estimate for total time required (in hours) to complete that task; estimate of hours spent so far on that task; revised estimate of total hours required; and estimated time remaining (the difference of the previous two columns).

See the sample spreadsheet printout in Figure 1 for an example. A template spreadsheet for each project will be made available online. Note that we will provide a suggested feature breakdown, but your team will be responsible for further subdividing this into tasks.

You don’t have to get your scheduling exactly right the first time! This exercise, in addition to helping you pair program the work, is intended to help you learn to estimate how long various
aspects of your projects will require. Prior evidence shows that most people (experienced programmers included) tend to underestimate how long most features will take to code. We expect these estimates to change (drastically during the first project, and hopefully somewhat less for following projects) and thus will not be grading you on how closely you meet your estimates. We will dock points if you don’t produce what we consider to be a reasonable, good-faith schedule. Please, be honest with yourself and your teammate in filling out these schedule sheets. The exercise benefits you to the extent that you take it seriously and provide accurate information.

With scheduling deadlines and lab hour requirements, a typical project life cycle might look like Table 1.

<table>
<thead>
<tr>
<th>Week 1</th>
<th>Project and team assignment handed out Tuesday; Initial schedule due Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 2</td>
<td>2 hours lab time required; Schedule revision due Friday</td>
</tr>
<tr>
<td>Week 3</td>
<td>2 hours lab time required; Schedule revision due Friday</td>
</tr>
<tr>
<td>Week 4</td>
<td>Project early deadline Tuesday; Ontime deadline Thursday</td>
</tr>
</tbody>
</table>

Table 1: Typical project life cycle.

4 Code Ownership

Since both partners will be working on code throughout the project, it would be difficult or impossible in most cases to determine who “owns” any given bit of code. Therefore, both partners will retain equal shares of ownership in all code produced.

How does this impact you? In a typical non-PP situation, you’re allowed to use code written for previous projects for all future projects. Even in a PP situation, so long as pairs stay the same, there’s no question about who can reuse code: the pair can be thought of as an individual programmer. In this class, though, you will be changing partners after each project. And some projects are intended to reuse code from earlier projects. Because you share equal ownership of the code, you may both take your code with you to the next project (and the next pair).

This has two important repercussions. For one, this means that on Projects 2 and above, two pairs may start from the same previous codebase, since both partners from the previous project might elect to use code from that previous project. The other side of that coin, though, is that your new partner will bring code from a previous project, so the two of you will have to negotiate out whose codebase to start from. For the final project, we are likely to require that project components from both team member’s prior efforts be included in the final submission.

Note that the Virginia Tech and Computer Science Honor Codes still apply. In particular, after a project has been turned in, old pairs should not continue to work together on their shared codebase, especially not once new pairs have been announced. The codebase should in essence be “forked” at the point when the old project is due, and each new pair may work on one branch of the fork. Backporting new features (to an earlier partner) or sharing new code between the forks will be considered cheating.