Collaborating with Students to Produce High-Quality Production Software

Sean Murthy, Andrew Figueroa, Steven Rollo
Data Science & Systems Lab (DASSL)
Department of Computer Science, Western Connecticut State University
Outline

• Introduction
  – Readiness of new CS graduates
  – Data Science & Systems Lab (DASSL)

• Details
  – Key development approaches and process steps
  – Agile-like development

• Perspectives
  – Student and faculty perspectives
  – Success factors, concerns, potential solutions

• Summary
Background

• Easier access to modern tools and methods seems to be increasing the adoption of software engineering (SE) practice (or improving SE practice)
  – Many tools are free and usable in many SE contexts
• Growth in individual adoption is apparent in open repositories
• Number of employers adopting modern tools and methods also appears to be on the rise
  – More job ads seeking people with modern skillset
  – Expect “agile and DevOps” mindset in employees, even for entry-level positions, which new graduates typically fill
Readiness of New Graduates

• Undergrad CS pgms (understandably) focus on core concepts: programming, algorithms, architecture,…
  – Typical pgm. requires students complete just one SE course
  – Some offer courses on software quality, as electives
• Students do simple exercises/projects as coursework
  – Little to no immersive end-to-end experience
• Industry internships offer dev and maintenance
  – Often emphasize maintenance as in “bug fixing”; interns rarely experience/glimpse the entire product life cycle
• Typical CS pgm is structurally unable or unlikely to produce grads with an “agile and DevOps” mindset
Hands-on Co-curricular Lab as a Solution

• Introduce undergrads to practical lightweight “agile and DevOps” processes

• Emphasize faculty-student collaboration to build, deploy, document, and maintain production software
  – Have students experience the entire software lifecycle

• Help students build online portfolios to readily show accomplishments to prospective employers
  – Give students a competitive edge: distinguish themselves from other “4.0 students”

• The Data Science & Systems Lab (DASSL) is such a lab
About DASSL (read dazzle)

• Started in January 2017
• One CS faculty member, who is also the lab director
• So far engaged 12 undergrads, including 5 presently
• Two released products: ClassDB and Gradebook
  – ClassDB currently used by about 50 students, with plans to deploy system wide for use by ~4000 students
• Four published papers (including at PNSQC 2018)
  – Two papers with student co-authors; one at an ACM conf.
• Alums testify online portfolio and DASSL experience helped in hiring, and continues to help in daily work
Activity Landscape

• Software lifecycle activities
  – Requirements management: planning and prioritization
  – Design and implementation: API, UI, general
  – Documentation: internal, external, technical, end user
  – Issue management: reporting, prioritizing, resolving
  – Release management: milestone planning, versioning
  – Maintenance: update code & data in current deployments
  – Publication: posters, papers, competitions, conferences

• Collaboration, teamwork, social coding
  – Almost all artifacts are in public GitHub repositories
  – Tools: Git, GitHub, wiki, Markdown, MS Teams and Office
Entry and Participation

• Open to all students, faculty, and staff
  – Typical student will have completed CS140, CS170, CS205
  – Ideal entry in 4\textsuperscript{th} semester, but likely in 5\textsuperscript{th} or 6\textsuperscript{th} semester

• **Not** part of CS program; all participation is voluntary
  – Free for students; no academic credit; faculty is unpaid
  – Occasional small stipend to students who help with lab ops

• Key: commitment to learn, with industry to match
  – Many students intend to enter; few actually do (that is OK)
  – Those staying past a sem. are likely to stay until graduation

• Same ground rules for all, **including faculty members**
Operations

• Meetings once a month during the academic year
  – Introduce new concepts; discuss ideas and issues

• Special sessions during summer and winter breaks
  – **Summer DASSL**: 6-10 weeks; held twice so far
  – **Winter DASSL**: 2-3 weeks; held once thus far
  – Each session has set goals; also when much work is done
  – Intense: 6 hrs/day, 4-5 days a week, on premise and online

• **DASSL Day**: one each semester to present new work
  – Practice presentation, recruit new students, inform admin

• Agile learning: incremental, in context, hands on, and continuous
Evidence of Progress

• Almost all of the work product is public
  – Includes discussions on issues and pull requests
  – Most output is free and open for non-commercial use
  – Adoption, collaborations, and suggestions are welcome

• Student testimonials are documented

• ClassDB is actively in use (3 semesters in a row)

• DASSL only scratches the SE surface
  – Many things we do not do: some knowingly
  – Many things we cannot do: not enough resources
  – Many things we plan to do (or do better)
DETAILS
ClassDB

• ClassDB is a database app built mostly in SQL
  – Instructors can use in teaching courses where students work with databases (both intro and advanced courses)
  – Create sandbox for each student/team; gives full control of sandbox to student/team; lets instructor read sandboxes
  – Maintains activity logs to help instructors monitor student progress and provide student-specific feedback
  – Runs unobtrusively in Postgres server instances

• Four releases to date: versions 1.0, 2.0, 2.1, and 2.2
  – 2800 executable LOC; 4700 total production LOC
  – 4400 total test LOC
Key Development Approaches

• Milestone-driven: begin each release with a **public wiki page** outlining informal list of features to add
  – Focus on a small theme of features to add (defects to fix)
  – Discuss and transform the informal list to a to-do list
  – Set due date for milestone and fix product version number
  – Create new issues and epics; tag issues with milestone
  – Comparable to sprints in a traditional agile process

• Issue-driven: log, classify, prioritize in GitHub Issues
  – Discuss alternatives, design soln. in **public** issue comments
  – **Self assign**: work on a pending issue with highest priority
  – Tag commits w issues: mutually trace issues and changes

ClassDB Milestone 2: [https://github.com/DASSL/ClassDB/wiki/Milestone-M2](https://github.com/DASSL/ClassDB/wiki/Milestone-M2)
Key Process Steps

• Version control: both code and non-code artifacts
  – GitFlow strategy: two long-lived branches, master and dev
  – Milestone’s work is off dev; merge dev to master at release

• Pull requests (PR) are required: must be approved
  – Generally all members review, comment, and approve
  – Focus each PR on one issue or closely-related set of issues

• Reviews: extensive for compliance, efficiency, reqs,…
  – Frequent commits ease code reviews; reviewers trained to (expected to, and often do) present solution alternatives

• Testing: unit tests required; object of each PR should pass all unit tests; testing is manual (CI in planning)
Agile-like Development

- Short cycles, milestone-driven dev, issue-driven dev together cause incremental product improvements

<table>
<thead>
<tr>
<th>Count</th>
<th>V1.0</th>
<th>V2.0</th>
<th>V2.1</th>
<th>V2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commits</td>
<td>355</td>
<td>204</td>
<td>234</td>
<td>60</td>
</tr>
<tr>
<td>Branches</td>
<td>51</td>
<td>30</td>
<td>21</td>
<td>14</td>
</tr>
<tr>
<td>Pull requests</td>
<td>52</td>
<td>29</td>
<td>23</td>
<td>14</td>
</tr>
<tr>
<td>Defects addressed</td>
<td>62</td>
<td>31</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>Enhancements</td>
<td>-</td>
<td>2</td>
<td>12</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Count</th>
<th>V1.0</th>
<th>V2.0</th>
<th>V2.1</th>
<th>V2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Attributes</td>
<td>34</td>
<td>34</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Functions</td>
<td>25</td>
<td>59</td>
<td>84</td>
<td>85</td>
</tr>
<tr>
<td>Views</td>
<td>0</td>
<td>14</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Triggers</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

- Conscious change from V2.0 to reduce #issues addressed in each PR
- Rise in #functions in V2.0 and 2.1 due to many API shortcuts added to slice user activity logs, but all are based on just three functions and one view
- Drop in #attributes in V2.1 due to changing a persistent table to temporary table
LOC Growth by ClassDB Version

<table>
<thead>
<tr>
<th>Version</th>
<th>Production</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>978</td>
<td>596</td>
</tr>
<tr>
<td>V2.0</td>
<td>1915</td>
<td>1688</td>
</tr>
<tr>
<td>V2.1</td>
<td>2699</td>
<td>2837</td>
</tr>
<tr>
<td>V2.2</td>
<td>2788</td>
<td>2879</td>
</tr>
</tbody>
</table>

- **Production**: Code lines | Comment lines | Blank lines
- **Test**: Code lines | Comment lines | Blank lines

Collaborating with Students to Produce High-Quality Production Software
A Quantitative Analysis of ClassDB Code

<table>
<thead>
<tr>
<th></th>
<th>V1.0</th>
<th>V2.0</th>
<th>V2.1</th>
<th>V2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count (growth % from previous version)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Files</td>
<td>12</td>
<td>23 (90%)</td>
<td>24 (4%)</td>
<td>26 (8%)</td>
</tr>
<tr>
<td>Code lines</td>
<td>978</td>
<td>1915 (96%)</td>
<td>2699 (40%)</td>
<td>2788 (3%)</td>
</tr>
<tr>
<td>Comment lines</td>
<td>350</td>
<td>729 (108%)</td>
<td>972 (33%)</td>
<td>1031 (6%)</td>
</tr>
<tr>
<td>Blank lines</td>
<td>249</td>
<td>646 (159%)</td>
<td>869 (34%)</td>
<td>898 (3%)</td>
</tr>
<tr>
<td>Total lines</td>
<td>1577</td>
<td>3290 (108%)</td>
<td>4540 (38%)</td>
<td>4717 (4%)</td>
</tr>
<tr>
<td><strong>Distribution: % of total lines</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code lines</td>
<td>62%</td>
<td>58%</td>
<td>59%</td>
<td>59%</td>
</tr>
<tr>
<td>Comment lines</td>
<td>22%</td>
<td>22%</td>
<td>21%</td>
<td>21%</td>
</tr>
<tr>
<td>Blank lines</td>
<td>16%</td>
<td>20%</td>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td><strong>Ratio of non-code lines to code lines</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comment to code</td>
<td>1 per 2.8</td>
<td>1 per 2.6</td>
<td>1 per 2.7</td>
<td>1 per 2.7</td>
</tr>
<tr>
<td>Blank to code</td>
<td>1 per 3.9</td>
<td>1 per 3.0</td>
<td>1 per 3.1</td>
<td>1 per 3.1</td>
</tr>
<tr>
<td><strong>Density: average LOC per file (also min-max LOC)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code lines</td>
<td>82 (4-284)</td>
<td>83 (2-347)</td>
<td>112 (2-610)</td>
<td>107 (2-610)</td>
</tr>
<tr>
<td>Comment lines</td>
<td>29 (7-64)</td>
<td>32 (8-134)</td>
<td>41 (8-163)</td>
<td>40 (8-152)</td>
</tr>
<tr>
<td>Blank lines</td>
<td>21 (3-69)</td>
<td>28 (4-109)</td>
<td>36 (4-184)</td>
<td>35 (3-184)</td>
</tr>
<tr>
<td>Total lines</td>
<td>131 (14-417)</td>
<td>143 (16-590)</td>
<td>189 (14-957)</td>
<td>181 (13-936)</td>
</tr>
</tbody>
</table>

Table 4 of the paper reproduced for convenience; Section 5.1 describes the content; salient data is bolded, e.g., some files contain large amount of code and need refactoring.
PERSPECTIVES
Figueroa

• Real responsibility, unlike classrooms or internships
  – Work is ungraded, but can carry significant consequences
  – Self-guided (as a team), requiring maturity and reflection

• Clear personal growth seen in ClassDB’s progression
  – Consistent improvement in all aspects: code quality, documentation, communication, project planning

• Presenting DASSL work at university events helped prepare for an international presentation

• DASSL has provided unique opportunities and career-long benefits
Rollo

- Focus on effective and efficient teamwork
  - Projects and team skills receive equal effort and attention
  - Both technical and social teamwork skills are utilized
- DASSL experience applies directly to the real-world
  - Helps to easily jump into new workflows
  - Gives enough experience to improve existing workflows
- DASSL products serve as an attractive portfolio
  - Provides employers with a very tangible proof of ability
  - Allows employers to assign responsibilities that are a better match for a new employee
Murthy: Success Factors

• Experience, time, and energy
  – Extensive industry and academic experience
  – Spend much of breaks with students: ~80% of the break
  – Work long hours, about 8 hours a day: a full-time job

• Lead by example
  – Participate in every aspect of product dev and mgmt.
  – Share critical review of own work as model for students
  – Make students feel comfortable to submit work for review
  – Earn and maintain student trust

• Continuously learn new tools and techniques
Murthy: Concerns and Potential Solutions

• The DASSL process is repeatable but not necessarily scalable or easily sustainable
  – Extremely labor and time intensive for faculty member
  – Small student pool: many interested; few (can) spend effort
  – Engagement only in breaks, only for a few sems/student
  – Different modes for new and seasoned members

• Some possible solutions
  – Increase faculty participation; maintain anchor students
  – Stipends: some students need outside jobs for sustenance
  – Add “agile and Devops” content (early) to curriculum
  – Support of industry (who benefit) and university admin

“DASSL process” refers to any process/effort comparable to that of DASSL.
SUMMARY
Summary

• DASSL addresses real gaps in experience and mindset new graduates likely have (employer perspective)
  – Engage students in all stages of SW lifecycle using much of the same processes and tools professionals use
  – 12 undergrads trained; 7 graduated beneficiaries testify

• Scaling and sustaining the process requires:
  – Critical mass of compatible students; experienced faculty
  – Much time and energy, and making up “opportunity cost”, on the part of both students and faculty members
  – Small but meaningful changes to CS curriculum
  – Support from university admins and industry beneficiaries