SAGE DevOps, Gameful Intelligent Tutoring, and Publication Proposal

COMS 6901 – Projects in Computer Science, Fall 2018

Alex Dziena / ad3363

Lily Yu Li / yl4019

Abstract

We propose continuing the work done on DevOps in previous semesters, and preparing for the publication of a reference architecture for a computational thinking-focused educational platform based on SAGE's current implementation and plans for future work. Specifically, we propose DevOps enhancements to improve the quality of the code base through code coverage, code standards and linting, and integration testing, and reduce the probability of broken builds by rejecting broken merges to shared branches. We also propose creating an outline, visualizations, and an initial draft of a Reference Architecture for future publication.

We will specifically be addressing the following Epics and Features:

Epic	Feature
SAGE Integration	DevOps MVP
	Workstream Integration
Gameful Intelligent Tutoring	Intelligent Hinting 1.1
Survey, Field Study Design, and Publication Strategy	SAGE Feasibility Study & Publication

Introduction

DevOps work on SAGE to date has been primarily focused on automation and improving the time-to-productivity for new researchers. With the introduction of continuous integration and deployment, and configuration management for shared and local development environments, the focus of DevOps work this semester will shift to improving code quality and overall project health, while avoiding any regressions resulting in a decrease in researcher productivity.

There is also an opportunity this semester to focus on the publication of a reference architecture for a computational thinking-focused educational platform, following the submission of SAGE's feasibility study to SIGCSE last semester. This reference architecture has been discussed in previous semesters, but a draft has not yet been created[1, p. 8].

SAGE Integration

Our focus on DevOps as part of the SAGE Integration epic this semester will comprise four focus areas, listed below by order of priority.

- Integration testing using Newman for APIs, and Selenium for web UI testing

- Documentation of SAGE’s Code Standards for each repository and language
- Post commit hooks to prevent build-breaking changes from integrating into the main (“development”) branch
- Improving test coverage to 60% per repository

Gameful Intelligent Tutoring

Over the course of the past two semesters, there has existed a stretch goal of implementing a “chatbot” interface to SAGE’s Gameful Intelligent Tutoring system. This has been blocked by critical integration work and publication preparation. This semester, we plan to complete integration of the existing intelligent hinting system into the SAGE UI in such a way that future intelligent hinting modules can be easily developed and integrated. We will prove the viability this integration framework by creating a natural language “chatbot” interface for students to “request” hints by asking free text questions.

Publication Strategy

Over the course of this semester, we plan to prepare an initial draft of a preliminary/concrete reference architecture. In particular this draft will include diagrams for both a general architecture and SAGE’s specific implementation, classification of the reference architecture using Angelov’s framework[2, pp. 417–431], and creation of an outline and rough draft of the architecture.

Related Work

SAGE Integration

[Dimensions of DevOps]
Lwakatare, et al. performed a survey of DevOps practices and identified four primary dimensions of DevOps practice: collaboration, automation, measurement, and monitoring. The paper goes on to specify a conceptual framework for characterizing DevOps practices. The paper does not detail concrete practices or provide reference architectures for implementing DevOps practices, but has been useful in conceptualizing DevOps implementation strategies in previous semesters and was used to identify many of the goals for this semester’s DevOps work including configuration management and test coverage measurement[3].

[Role of collective ownership and coding standards in coordinating expertise in software project teams]
This paper explores expertise coordination as an important emergent process through which software project teams manage software development challenges, in particular within the framework of Extreme Programming (XP). Maruping et al examine the relationship between collective ownership and coding standards with software project technical quality in a field study

of 56 software project teams comprising 509 software developers, and found that collective ownership and coding standards play a role in improving software project technical quality. They find that coding standards strengthens the relationship, resulting in higher technical quality. This work inspired and will guide development of SAGE’s coding standards documentation[4, pp. 355–371].

[Parameterizing random test data according to equivalence classes]
Professor Kaiser, et al. present a framework for parameterized random test case generation for machine learning applications. As stated in this paper, there is no reliable test oracle for ML applications; i.e. we can not, with reasonable confidence, predict the correct output for a given random input[5].
This research will guide a testing strategy for our Gameful Intelligent Hinting system, with an aim to reach 60% code coverage, via parameterized random testing, similar to a constrained version of “fuzz” testing.

Gameful Intelligent Tutoring

[Hints: Is It Better to Give or Wait to Be Asked?]
Razzaq, et al. found that students learned more reliably when they were provided with a mechanism to receive hints-on-demand rather than being provided with hints proactively in a “just-in-time” hinting system[6]. Because this effect was more pronounced in students who asked for a larger number of hints, the study may suggest that the relative benefit of hints-on-demand vs proactive hints increases as a student needs increasing amounts of assistance. This paper provided us the inspiration and rationale for a natural language interface to an on-demand hinting system.

[Autonomously Generating Hints by Inferring Problem Solving Policies]
Piech, et al. performed analysis on a large amount of student solutions data gathered from Code.org and proposed several different path modeling algorithms used to learn a Problem Solving Policy (PSP), a policy that returns a suggested next partial solution for all partial solutions to a given puzzle, i.e. a PSP π is defined as $s' = \pi(s)$ for all $s \in S$, with S being the set of all possible solutions. The algorithms used to learn this PSP were compared, with a class of algorithms called Desirable Path algorithms (Possion Path and Independent Probable Path) performing the best - producing PSPs closest to the paths contained in an expert-labeled set of suggested paths[7, pp. 195–204].
“Poisson Path”-based PSP learning and hint generation were implemented last semester[8]. This semester, we will build on that work by integrating the hint generation engine with SAGE’s UI, and productionizing the policy learning algorithms to allow for near real-time reinforcement learning of policies.

[Delivering Hints in a Dialogue-Based Intelligent Tutoring System]
Zhou, et al present an implementation of CIRCSIM-Tutor, a dialogue-based intelligent tutoring system. The system used a a labeled corpus of dialogue (questions and answers) from expert

tutors to develop a set of hinting strategies used to select appropriate hint templates and parameters as responses to classes of questions. CIRCSIM-Tutor does not support online-updating, but is context aware of it's previous interactions with a student during a single session[9, pp. 128–134].
We seek to build on this work by implementing a reinforcement learning algorithm to allow for online learning by the chatbot agent. Possible candidates that we will evaluate include SARSA, temporal difference learning, and Q-learning[10]. Since we do not have a labeled corpus on which to train, we will rely on semi-supervised learning by assigning reward values to the next student move, and updating the policy in an online (or near online, in the case of SARSA) manner.
The CIRCSIM-Tutor paper also provides an evaluation framework for the efficacy of different hinting strategies, and a set of heuristics to select appropriate hint types. We will use this methodology to evaluate the hinting strategies learning by our chatbot, and display results in SAGE's researcher interface[9, pp. 128–134].

Survey, Field Study Design, and Publication Strategy

[The concept of reference architectures]
Cloutier, et al. examine reference architectures with the goal of providing a more precise definition of their components and purpose. They propose that the value provided by reference architectures lies in the distillation of (in many cases) thousands of person-years of work, a shared baseline for multiple, often cross-functional teams, and guidance for future work. We will reference the architecture evaluation methodologies and structure / component suggestions put forth by this paper while redeveloping SAGE's reference architecture included in the feasibility study draft[11, pp. 14–27].

[A framework for analysis and design of software reference architectures]
This paper provides a tool in the form of an analysis and design framework, for the creation of software reference architectures based on three primary dimensions: context, goals, and design. The paper goes on to define five types of reference architectures into which architectures under analysis can be classified:

1. classical, standardization architectures for use in multiple organizations
2. classical, standardization architectures for use in a single organization
3. classical, facilitation architectures for use in multiple organizations
4. classical, facilitation architectures for use in a single organization
5. preliminary, facilitation architectures for use in multiple organizations

The paper validates the framework by applying it to analysis of 24 reference architectures. We plan to use the framework proposed by this paper to design and evaluate our updated reference architecture[2, pp. 417–431].

[The visual display of quantitative information]
This book is widely used in industry and academia, and provides an exploration and set of recommendations for the design of statistical graphics, charts, tables. It also provides an

analysis framework for selecting appropriate data visualizations for a given data set, attempting to optimize for precision, efficacy, and speed of analysis. We will use this book as a reference when evaluating and potentially redesigning the data visualizations in the feasibility study draft[12].

Proposal

SAGE Integration: DevOps MVP

**Note: Newman and code standards proposals have been included in either initial proposals or "future work" proposals for several semesters. The proposals for these two focus areas are retained here.*

We will integrate Newman to facilitate testing of sage-node and sage-frontend APIs in local development environments, and allow for automated API testing in our continuous integration system. Continuous testing of our APIs will further enhance system stability, and reduce the number of API regressions in future semesters, which will be critical to successful integration of workstreams in the fall and future semesters.

Adding coding standards documentation has been found to play a role in improving software project technical quality[4, pp. 355–371]. We believe that adding coding standards to SAGE's documentation will also reduce the time to productivity for new researchers, maintain the health of the SAGE codebase, and allow for easier future integration of linting, mutation, and static analysis tools.

To improve researcher productivity, and improve the code health of the main development branch, we propose to implement post commit hooks that will reject commits that break builds or fail tests in the primary, "development" branch of each repository. This will require a re-architecture of our continuous integration system, requiring a mechanism similar to pull requests[13, pp. 364–367] with the CI process playing the role of "maintainer" of the main branch.

Requiring all tests to pass for each repository will not be sufficient unless a minimum test coverage threshold is met, as new code could be committed that is not covered by any test cases. To this end, we will improve test coverage to 60%, and implement an additional post-commit hook that checks for this minimum level of test coverage.

Gameful Intelligent Tutoring: Intelligent Hinting MVP

**Note: This is a modified version of the GIT work proposed over the course of the summer semester[1]. Because the focus in that semester was primarily DevOps and Publication*

Preparation, and because the originally proposed GIT work remains relevant this semester, we are proposing a largely similar, but slightly modified version.

Since the existing HMM and Poisson Path for modeling and clustering student learning paths as detailed in Intelligent Hinting 1.1 Final Report[8] already have a basic API for serving hints, we will begin by querying this API from the sage-scratch swf to make hint data available to the UI. Afterwards we will reuse the block suggestion UI engine built in a previous semester to render the hint. Once this basic integration is complete, we will extend it to support textual hints (rendered in the avatar UI), allowing us to build a chatbot engine and interface.

To make the chatbot as responsive as possible, we will extend the existing hinting system to allow for near-continuous updating of student behavior models. This will require building a new API endpoint to allow for model updating, and productionizing the model training pipeline built last semester. This pipeline is built in Python, which will require extending our continuous build system and other DevOps infrastructure to support a new language.

The chatbot responses will utilize the student's current path cluster assignment and inferred student classification (Extreme Movers, Movers, Stoppers, and Tinkers) to select an appropriate hint on demand. We will infer intent from the student's input using multiple methodologies, including n-gram, bag of words, and word2vec, then select the appropriate hint using the intent, student classification, path cluster assignment, and path cluster progression as inputs.

Parameterized, randomly generated test inputs would allow us to efficiently test both the existing GIT system, and the added chatbot interface. A potential strategy to explore is generating a large set of examples, each of which has a number of attribute values and a label. If we use one-hot encoding to represent the attribute values and labels, genetic algorithm evaluation could allow us to continuously determine the test input-set for which the approximate maximum number of test cases fail, reducing the test generation problem to a maximization problem.

Publication Strategy: SAGE Feasibility Study & Publication

We will focus on developing the reference architecture proposed in SAGE's draft feasibility study to prepare for eventual publication. Specifically, we will create an initial outline, classify our reference architecture using Angelov's framework[2, pp. 417–431], and generate diagrams for the architecture, with a particular emphasis on data visualization best practices[12].

Timeline

Date	**Milestone**

2018-10-05	First draft of code standards completed
2018-10-12	Coding standards complete, Post-commit hooks completed
2018-10-19	Newman Configuration
2018-10-26	Reference Architecture outline completed
2018-11-02	Complete integration of prior GIT work into SAGE UI
2018-11-09	Improve test coverage for all repositories to 30%, including AsUnit test suite for sage-scratch and parameterized random testing for ML codebase.
2018-11-16	Continued work on chatbot prototype
2018-11-23	Chatbot prototype, test coverage at 60%
2018-11-30	Chatbot UI integration
2018-12-07	Reference Architecture diagrams / data visualizations completed
2018-12-14	Reference Architecture first draft completed

Future Work

Gameful Intelligent Tutoring

- **Context-aware hinting mode selection:** Based on the path progression and wall time of a student in a given puzzle or constructionist game, switch the hinting system between “just-in-time” and “on-demand” hinting. Possibly, decrease the specificity of the hints based on inputs from the outer loop, i.e. how many assignments a student has completed weighted by the “dominant” mastery of each of those assignments.
- **Chatbot semi-supervised training:** Use student outcomes resulting from a set of provided hints to train the chatbot hint provider in real time. Improve the pedagogical efficacy of hints by retraining the hint provider on a batch basis, potentially using question/hint/outcome triples from non-SAGE corpora.

SAGE Integration

- **Code Review:** Require a code review from another researcher, who has expertise in a relevant technology stack, before merging to the “develop” branch. This would increase collective ownership, help reinforce adherence to the documented coding standards, and improve the overall quality of the code base.

- **Automated team performance metrics tracking:** Automated tracking of team velocity, schedule performance index (SPI), cost performance index (CPI) based on time spent per task, improvement index (weighted average rate of change across velocity, SPI, CPI and other performance metrics). This information is provided back to the team to identify areas for improvement.
- **Automated "fuzz" testing:** Generate random inputs for test cases in all code bases, with automated refinement using genetic algorithm or SGD optimization, as opposed to only the GIT/ML codebases. Surface test failures as bugs in the task tracking system.

Survey, Field Study Design, and Publication Strategy

- **Future publication:** Our draft of a reference architecture for a computational thinking-focused educational platform could be revised and submitted for publication in the future. This architecture could form the basis for field studies of the validity of the architecture in future semesters.

References

[1] A. Dziena, "SAGE Intelligent Hinting, DevOps, and Publication Proposal," p. 7, Jun. 2018.

[2] S. Angelov, P. Grefen, and D. Greefhorst, "A framework for analysis and design of software reference architectures," *Information and Software Technology*, vol. 54, no. 4, pp. 417–431, 2012.

[3] L. E. Lwakatare, P. Kuvaja, and M. Oivo, "Dimensions of DevOps," in *Lecture Notes in Business Information Processing*, 2015, pp. 212–217.

[4] L. M. Maruping, X. Zhang, and V. Venkatesh, "Role of collective ownership and coding standards in coordinating expertise in software project teams," *European Journal of Information Systems*, vol. 18, no. 4, pp. 355–371, 2009.

[5] C. Murphy, G. Kaiser, and M. Arias, "Parameterizing random test data according to equivalence classes," in *Proceedings of the 2nd international workshop on Random testing co-located with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007) - RT '07*, 2007.

[6] L. Razzaq and N. T. Heffernan, "Hints: Is It Better to Give or Wait to Be Asked?," in *Lecture Notes in Computer Science*, 2010, pp. 349–358.

[7] C. Piech, M. Sahami, J. Huang, and L. Guibas, "Autonomously Generating Hints by Inferring Problem Solving Policies," in *Proceedings of the Second (2015) ACM Conference on Learning @ Scale - L@S '15*, 2015.

[8] Y. Ding, W. Luo, and J. Zhang, "Intelligent Hinting 1.1 Final Report," May 2018.

[9] Y. Zhou, R. Freedman, M. Glass, J. A. Michael, A. A. Rovick, and M. W. Evens, "Delivering Hints in a Dialogue-Based Intelligent Tutoring System," *AAAI/IAAI*, Jul. 1999.

[10] R. S. Sutton, A. G. Barto, Co-Director Autonomous Learning Laboratory Andrew G Barto, and F. Bach, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[11] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, and M. Bone, "The Concept of Reference Architectures," *Syst. Engin.*, vol. 2, 2009.

[12] E. Tufte and P. Graves-Morris, "The visual display of quantitative information.; 1983." 2014.

[13] M. M. Rahman and C. K. Roy, "An Insight into the Pull Requests of GitHub," in

Proceedings of the 11th Working Conference on Mining Software Repositories, Hyderabad, India, 2014, pp. 364–367.