SAGE Feasibility Study Support - Proposal

COMS6901 – Projects in Computer Science, Spring 2018

Alex Dziena/ad3363

Johan Sulaiman/js5063

Contents

Abstract

The proposed project aims to help achieve the two immediate SAGE milestones ahead: 1) administer and summarize public surveys and field studies on feasibility and effectiveness of SAGE researches, and 2) assist in publishing a feasibility study that describes the theoretical and empirical underpinnings of SAGE. We look to accomplish these through the completion of some features in the Gameful Intelligent Tutoring, SAGE Integration, and Experimental Design and Publication Strategy Epics, as follows:

Epic	Feature	Aim
Gameful Intelligent Tutoring	Intelligent Hinting MVP	On-demand hints implementation; Multi-layer Hinting POC implementation; Gameful Hinting System design; Selection of Hints once Hints are classified (structure is pre-canned, but the content is on-the-fly); Stretch Goal: POC implementation of a learning chatbot, using human tutor responses as training data
SAGE Integration	Workstream Integration	Coordinate and enhance a one-system-feel of SAGE via integration of SAGE modules even as they continue being developed within the four main SAGE research areas
SAGE Integration	DevOps MVP	Automatic propagation of test results and RCs to all teams; Continuously integrated development environment; Post-On-Green User Acceptance Testing environment
Survey/Field Study Design and Publication Strategy	Spring 2018 Publication - SAGE Feasibility	Coverage of the survey and and tie in with field studies conducted this semester; Publish a summary report

Motivation and Related Work

One of SAGE's priorities in the Fall of 2017 was to achieve an integrated implementation of SAGE back-end and front-end components across all SAGE modules. We now propose a formalization of this SAGE implementation for usage in planned field studies, leveraging some best practices in the area of DevOps and Agile team/project management. The effort will also include operationalizing the Intelligent Hinting functionality in the Student Affinity Space. As the work progresses, feedback received from a newly designed and distributed SAGE feasibility survey will be taken into account and research, analysis,

and implementation performed through these activities will serve as key inputs into the SAGE Feasibility Study publication targeted to complete in the middle of 2018.

Gameful Intelligent Tutoring: Intelligent Hinting MVP

The initial implementation of the Intelligent Hinting POC seeks to build upon the "Intelligent Hinting and Behavior Detection in SAGE" work (Chengwei & Zhang, 2017). Specifically, we seek to implement an algorithm that reads features extracted from the behavior detection system and returns a relevant hint classification of "General Hints", "Hints of Average Informativeness," or "Specific Hints." This feature is intended to improve student retention of programming concepts while minimizing "hint weight", which is a count of hints provided to the student weighted by hint specificity, with "General Hints" having the lowest weight, "Specific Hints" having the highest weight, and "Hints of Average Informativeness" having a weight between the two. The success criteria for this feature will be improved average retention scores for students versus random selection of hints, using the equal or less "hint weight".

The intelligent hinting POC is inspired by prior work comparing human tutors to intelligent tutoring systems (ITS's) (Merril, et al 1992; Hume et al, 1996). A stretch goal for the project is to implement a POC "chatbot" hinting system (Zhou, et al, 1999), that would perform sentiment and intent analysis on student input (i.e. questions entered into a chat system) and return a generated, natural language response, based on responses learned from human tutors. The motivation here is to prove the viability of a "hintless" or entirely on-demand tutoring system, using a natural language interface, that has an equivalent or better impact on student outcomes than the existing time-based or just-in-time hinting systems.

Work on the intelligent hinting system must be coordinated with the Intelligent Hinting 1.1 team. To facilitate this, Alex will join team status calls, review system design, and contribute to that feature.

SAGE Integration: Workstream Integration

The current project seeks to extend the SAGE Gameful Assessment integration work (Sulaiman, 2017) to now cover additional aspect of SAGE more comprehensively while maintain the emphasis on system presentability to end-users. Due to heightened interest in SAGE's earlier works, the Spring 2018 semester sees the highest influx of researchers to SAGE. It is more important than ever for active researchers to stay coordinated and working towards a common version of SAGE as we approach the field study phase of SAGE. We seek to apply Agile methodologies and project management to ensure the various research and implementation trajectories remain cohesive.

SAGE Integration: DevOps MVP

The DevOps MVP feature seeks to improve implementation and research velocity, provide a stable environment for field studies, improve cohesion between the work of all research teams, and provide a framework for ongoing work on SAGE. Success criteria will include an increase in research and development velocity over the course of the semester, achievement of at least 99% availability (High Availability, 2018) before the start of field studies, and test coverage of at least 60% of the code base.

We seek to build on top of accepted DevOps best practices for collaboration, automation, measurement and monitoring (Lwakatare, 2015), borrowing concepts and framework from implementation in industry as described in 2013 by Feitelson, Frachtenberg, and Beck.

Survey, Field Study Design, and Publication Strategy

Recent researches (Lee, et al 2013; Ihantola et al, 2016) show there is a continuing shortage of work on data-driven educational technology in general and in-game assessment specifically. This is due to several factors (Evans, 2012): 1) Current programmatic limitation still exists when it comes to distilling meaning from language, so free-form answers are difficult to interpret and assess, 2) a highly effective teaching method is debriefing, but this is challenging to administer, 3) there is a recognized higher risk for students to game or cheat the system. Another research (Harteveld, et al 2014) further boosts motivation for that the space for an educational game that teaches Computer Science or Computational Thinking is still wide open and fertile for new research. These findings validate the conviction that a concentrated research effort on socially gameful approach to teaching Computational Thinking will be able to advance new learnings in the field. Surveys and field studies are the research methods of choice this semester to gather data for further analysis and refinement of the various SAGE research areas.

Proposal

Intelligent Hinting MVP

Initial implementation: Multi-layer Hinting POC

The initial implementation of the Multi-layer Hinting POC will build a decision tree learned using C4.5 or CART. Features would be sourced from the features extracted from Scratch Analyzer. Selection of the actual algorithm will be based on classification accuracy as evaluated using k-fold cross validation over a set of labeled examples (these can be procedurally generated or manually labeled). The decision tree

will classify student's current performance into one of three buckets corresponding to hint levels: "General Hints", "Hints of Average Informativeness," or "Specific Hints." The model will be retrained offline based on success metrics of students in field studies, which would include time and number of actions to complete a task and "hint weight" as defined above, incorporating a reinforcement learning approach. Once an algorithm has been selected, implementation of AdaBoost based on random stumps or a random forest (Breiman, 2001) would give the model resilience against overfitting, and potentially the flexibility to expand to more classes easily.

Once a decision tree is built, and after analysis of the error rate in the final implementation, this could be re-implemented as an Artificial Neural Network which would allow us to take advantage of latent features. This would retain the multiclass classifier, but would remove the need for continued feature engineering from the Programming Behavior Detection feature.

Stretch goal: Chatbot POC

A chatbot could be trained in a semi-supervised fashion using Q-Learning (or SARSA, based on initial results), learning from labeled examples from a human tutor. Once trained, the model would continue to learn by scoring rewards based on student success metrics. This would be a significant implementation project, which would involve sentiment and intent analysis on student input via an n-gram model, and would need to incorporate features engineered in Programming Behavior Detection.

Both approaches require strict definition of success metrics. Is success defined by speed of task completion? Performance in tests administered later? Future success of the student in some field(s)? There are several opportunities for future work here, including long-term research on student outcomes.

Workstream Integration

With the influx of new researchers in Spring 2018 (the lab started with 20 researchers at the beginning of the semester), there is a greater need than ever to keep the researchers' work well coordinated across the Gameful Constructionism, Gameful Direct Instruction, Gameful Intelligent Tutoring, and Gameful Affinity Space areas. Our planned contribution includes helping facilitate Lab Interaction, increase lab productivity, and enhance Lab processes and documentations. The SAGE Assessment Architecture diagram will be expanded to also include other components such as the Recommender Engine and the Intelligent Tutor components. The Theoretical Foundation review on the SAGE wiki will be completed, and the wiki space overall will continue to be updated.

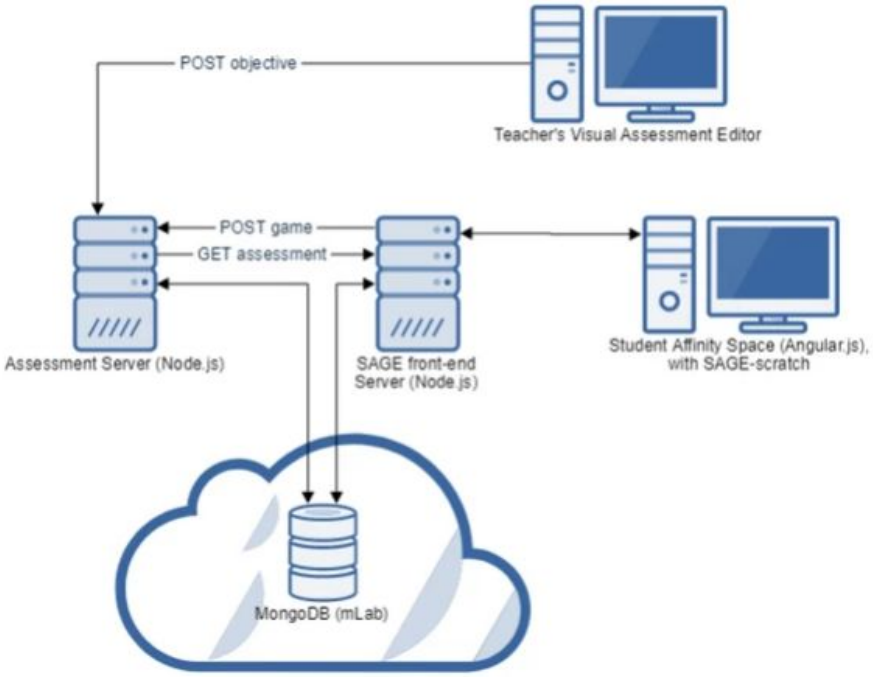

Figure 1: SAGE Assessment Architecture Diagram (to be expanded)

From the Agile project management perspective, there will be more cross-team communications planned this semester, leveraging heavily the SAGE slack channels as well as the Agile-based Epics/Features/User-Stories/Tasks organization on TFS. A periodic researcher forum will also be organized where researchers are free to share their progress, roadblock, and accomplishments with each other and to collaborate across the different teams.

Meeting Guide

Prior to the meeting

- Please add several bullet points representing your activities since the last Researcher Forum. For example:
  - Progress/accomplishment/news (in SAGE, at school, at home, etc.)
  - Hurdle/roadblock
- Read what other researchers wrote

During the meeting

- Go around the room and speak about your highlights
- Field at least one question from the audience

The most important Meeting Metric that matters is engagement!

Figure 2: Planned Meeting Guide for the Researcher Forum

There will also be an online, semi-persistent SAGE environment available on the cloud to achieve to activate a demonstrable and fully functioning SAGE implementation useful as reference points to researchers and testers. This environment will be related very closely to the work to implement DevOps paradigm onto SAGE that is discussed in the section following. This effort will also play an important role to set and maintain SAGE's high state of polish, quality and presentability to external stakeholders including testers, partners, and field study subjects. We would also like to institute a mid-semester code freeze as the most natural check-in point for all current implementation work to be gathered into a stable version, which can subsequently be used for field studies.

DevOps

Automatic propagation of test results and RCs to all teams

To facilitate implementation velocity and maintain a high level of quality in shared in environments, we will automate notifications of test results and new version deployments to feature teams. We will also automate “one-push” rollback for teams if they see abnormalities in test results, which will allow them to restore a shared environment to a previous state, if necessary.

Continuously integrated development environment

The development environment will be built continuously on every commit to the codebase, and will be deployed on every ”unit test green” build (all unit tests pass, but integration or end-to-end tests are not required to pass). This will ensure availability of the development environment for all teams, and automate the merge-test process.

We can also consider requiring code reviews for deployment to the UAT environment.[1] This introduces an additional, time-intensive manual step to UAT deployment and may slow velocity; we will evaluate code quality, build breakage, test coverage, and failed UAT rates mid-semester to determine if this is warranted. Deployment to production would require all tests (unit, integration, and end-to-end) to be green, and a project lead to manually approve the deployment.

[1] https://docs.microsoft.com/en-us/vsts/tfvc/get-code-reviewed-vs

TFS Code Migration from GitHub

Using TFS as a code repository would allow for a more straightforward migration. This effort will require coordination with other team members involving rigor in performing final commits, and putting down a hard prerequisite for all researchers to be properly registered on TFS with the right access. Some rudimentary training on the basic operations of Git will also be necessary for all researchers. We will borrow from publicly available knowledgebases on GitFlow[2] as our source control process.

Build automation

We aim to proceduralize build automation as follows:

- Push on commit for develop
- Push on green for UAT
- Push on green + review for RC
- Manual push + deploy for master

This could be implemented natively in TFS or in open source tools, e.g. Jenkins. Using Jenkins would allow us to leverage a potentially larger ecosystem of open source CI / analysis / build automation tools, but would require an additional build server (or at least a "build server" container living in another virtual machine).

Build Test Coverage

We seek to cover at least 50% of the codebase with unit tests by the end of the semester. Integration test coverage is more difficult to measure. Minimally, we seek at least one integration test suite per research team. As part of the testing initiative, we will include documentation on mocks and fakes in the SAGE wiki. To facilitate testing, we will create golden test datasets for use in integration and end-to-end tests.

Survey, Field Study Design, and Publication Strategy

We plan to support the design and distribution of a survey so we can gather feedback on the feasibility and applicability of SAGE in a formal education setting. Afterwards there will be work

[2] Abhishek Dwaraki, Srini Seetharaman, Sriram Natarajan, and Tilman Wolf. 2015. GitFlow: flow revision management for software-defined networks. In Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research (SOSR '15). ACM, New York, NY, USA, , Article 6 , 6 pages. DOI: http://dx.doi.org/10.1145/2774993.2775064

to collect and organize the data, as well as perform analysis which will be reported in the planned SAGE feasibility study publication.

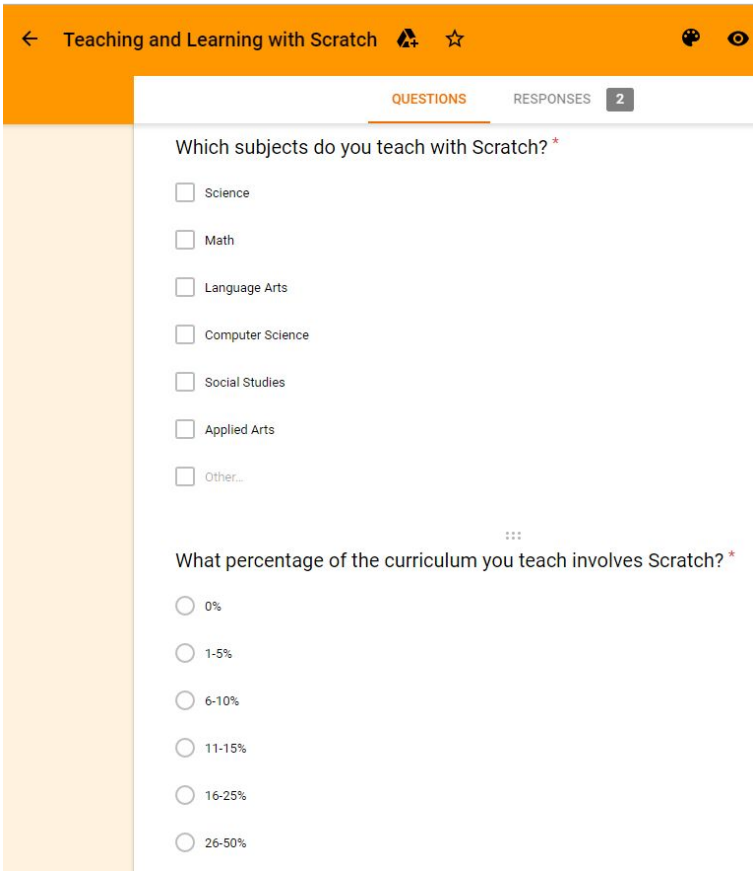

Figure 3. Sample draft of SAGE Survey

The work from the Workflow Integration and DevOps sections above will be instrumental to the success of the field study. We are targeting the following possible venues where the study can be conducted: Computer Clubhouse, techCanal, codeHer, and Girls Who Code. The publication will be on SAGE research leading up to Spring 2018, including: 1) survey results, 2) literature review, 3) discussion on SAGE's theoretical foundation, and 4) feedback/observation report from preliminary user centered field study sessions.

Milestones

The following table shows the steps involved in this proposal and their associated expected completion dates:

Task	Date
Facilitate Lab Interaction, increase Lab productivity, & enhance Lab's workflow/processes	Sprint 0 through Sprint 5
DEV/CI Environment Deploy Automation and Unit Test Framework complete	Sprint 1 - 03/01
Complete baseline version and enable of SAGE Continuous Integration	Sprint 1 - 03/01
Design, prepare and distribute Survey	Sprint 1 - 03/01
Design Field Studies	Sprint 1 - 03/01
Complete Intelligent Hinting (text hint, block-shaking) with Decision Tree POC	Sprint 1 - 03/01
Midterm Code Check-in by all SAGE Teams; pass all Continuous Integration Tests	Sprint 2 - 03/15
UAT Push-on-green, Prod push-on-green + project lead approval complete	Sprint 2 - 03/15
Intelligent Hinting - ANN POC	Sprint 3 - 03/29
Conduct and Summarize Field Studies	Sprint 4 - 04/12
Interactive Intelligent Hinting Analysis and Framework - *Stretch Goal*	Final Report Sprint - 05/03
Code Freeze and Final Spring 2018 CI Build, Final Report and Presentation	Final Report Sprint - 05/03
Complete First Draft of SAGE Feasibility Study	Final Report Sprint - 05/03
Documentation and Code migration	05/31

Future Work

There are potential future trajectories in the following areas after the main work on this proposal has concluded:

- Gameful Intelligent Tutoring: Intelligent Hinting POC / Machine Learning
    - **Chatbot training data:** Conduct field research to gather labeled data from human tutors to train the dialogue based hinting system.
    - **Multi-student reinforcement learning:** Online classification of students via k-means clustering over student milestone states (as mentioned in Piech), and propagation of

rewards learned from performance of similar students back to the hint selection and generation models.

- SAGE Integration: DevOps
    - **Release Candidate Environment:** An environment that is identical to production, but contains only the next release candidate. This environment would have the same access control, security configuration, and external availability as production, and would be similar to a "public alpha" environment.
    - **Code Review:** Require a code review from another researcher, who has expertise in a relevant technology stack, before merging to the "develop" branch.
- SAGE Integration: Workstream Integration / Cohesion
    - **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.

We also hope to continue the findings from the survey and the field report to perform more targeted follow-on research and experiment on SAGE Intelligent Tutoring research area.

.

References

Bloom, B. S., 1984. “The search for methods of group instruction as effective as one-to-one tutoring.” Educational Leadership, 41(8), 4–17.

Breiman, L., 2001. “ Machine Learning,” 45: 5. https://doi.org/10.1023/A:1010933404324

Butz, C. J. , Hua, S., and Maguire, R. B., 2006. "A web-based Bayesian Intelligent Tutoring System for Computer Programming," Web Intelligence and Agent Systems, vol. 4, no. 1, 2006.

Chengwei, B., Zhang, M., 2017, “Intelligent Hinting and Behavior Detection in SAGE,” Columbia Programming Systems Lab. Columbia University, New York, NY

Csikszentmihalyi, M., 2000. "Flow," in The Encyclopedia of Psychology, A. Kazdin, Ed. American Psychological Association and Oxford University Press, pp. 381-382

Docs.microsoft.com. (2018). Get your code reviewed with Visual Studio. [online] Available at: https://docs.microsoft.com/en-us/vsts/tfvc/get-code-reviewed-vs [Accessed 3 Feb. 2018]

Dwaraki, Abhishek , Seetharaman, Srini, Natarajan, Sriram, and Wolf, Tilman. 2015. “GitFlow: flow revision management for software-defined networks.” In Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research (SOSR '15). ACM, New York, NY, USA, , Article 6 , 6 pages. DOI: http://dx.doi.org/10.1145/2774993.2775064

Evans, M., Jennings, E., and Andreen, M., 2012. "Assessment through achievement systems: a framework for educational game design," in Developments in Current Game-Based Learning Design and Deployment. IGI Global

Feitelson, D.G., Frachtenberg, E., Beck, K.L.: Development and Deployment at Facebook. IEEE Internet Computing 17, 8–17 (2013)

Harteveld, C., Smith, G., Carmichael, G., Gee, E., and Stewart-Gardiner, C., 2014. "A design-focused analysis of games teaching computer science," in Proceedings of the Games, Learning and Society Conference

Helminen, J. , Ihantola, P., Karavirta, V. , and Malmi, L., 2012. "How do students solve parsons programming problems?: an analysis of interaction traces," in Proceedings of the ninth annual international conference on International computing education research

High availability. (2018, January 11). In Wikipedia, The Free Encyclopedia. Retrieved 03:22, February 3, 2018, from https://en.wikipedia.org/w/index.php?title=High_availability&oldid=819841553

Hume, G., Michael, J., Rovick, A., & Evens, M.,1996. “Hinting as a tactic in one-on-one tutoring,” The Journal of the Learning Sciences, 5(1), 23-47.

Ihantola, P., et al, 2016. “Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies,” in Proceedings of the 2015 ITiCSE on Working Group Reports, pp. 41-63

Koedinger, K. R., Brunskill, E., Baker, R. S., McLaughlin, E. A., andStamper, J. , 2016. "New Potentials for Data-Driven Intelligent Tutoring System Development and Optimization," AI Magazine, vol. 34, no. 3, pp. 27-41, 2016

Lee, M. J., Ko, A. J., and Kwan, I., 2013., "In-game assessments increase novice programmers' engagement and level completion speed," in Proceedings of the ninth annual international ACM conference on International computing education research, pp. 153-160

Lwakatare, L. E., Kuvaja, P., & Oivo, M., May, 2015. “Dimensions of DevOps,” In International Conference on Agile Software Development (pp. 212-217). Springer, Cham.

McNamara, D. S., et al. 2010. “Intelligent Tutoring and Games (ITaG)”.

Merrill, D. C., Reiser, B. J., Ranney, M., & Trafton, J. G., 1992. “Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems,” The Journal of the Learning Sciences, 2(3), 277-305.

Piech, C., Sahami, M., Koller, D., Cooper, S., and Blikstein, P. 2012. “Modeling How Students Learn to Program,” Stanford University, Stanford, CA, ACM

Pmi.org. (2018). Agile Approaches on Large Projects in Large Organizations. [online] Available at: https://www.pmi.org/learning/academic-research/agile-approaches-on-large-projects-in-large-organizations [Accessed 3 Feb. 2018].

Pmi.org. (2018). Applying work breakdown structure to project lifecycle. [online] Available at: https://www.pmi.org/learning/library/applying-work-breakdown-structure-project-lifecycle-6979 [Accessed 3 Feb. 2018].

Rupp, A., Matthew G., Mislevy, R., Shaffer, D., 2010. “Evidence-centered Design of Epistemic Games: Measurement Principles for Complex Learning Environments,” JTLA, Vol 8, No 4

Shernoff, D. J., Csikszentmihalyi, M., Shneider, B., & Shernoff, E. S., 2003. “Student engagement in high school classrooms from the perspective of flow theory.” School Psychology Quarterly, 18(2), 158-176. doi:10.1521/scpq.18.2.158.21860

Sheth, S., Bell, J., and Kaiser, G., 2013, "A competitive-collaborative approach for introducing software engineering in a cs2 class," in Proceedings of IEEE Conference on Software Engineering Education and Training

Sulaiman, J. 2017, “Responsive and Gameful SAGE Assessment,” Columbia Programming Systems Lab. Columbia University, New York, NY

Zhou, Y., Freedman, R., Glass, M., Michael, J. A., Rovick, A. A., & Evens, M. W. (1999, July). Delivering Hints in a Dialogue-Based Intelligent Tutoring System. In AAAI/IAAI (pp. 128-134).