SAGE Field Study Support

Final Report

COMS6901 – Projects in Computer Science, Spring 2018

Alex Dziena/ad3363

Johan Sulaiman/js5063

Table of Contents

Introduction **2**

Related Work **3**

SAGE Integration: Workstream Integration 3

SAGE as a Growing Research Project 3

Integration of Workstreams Within a Growing Research Team 4

GitFlow 5

SAGE Integration: DevOps MVP 5

Existing SAGE DevOps Infrastructure 5

Dimensions of DevOps 5

Collaboration 6

Automation 6

Measurement 6

Monitoring 6

Development and Deployment at Facebook 7

Survey, Field Study Design, and Publication Strategy 8

Accomplishments **8**

SAGE Integration: Workstream Integration 9

Online, Persistent SAGE Instance for Field Studies 9

Updated Data Model Diagrams 9

Software-tier Reference Diagram 10

Activity Flow Reference Diagram 11

End of Semester Researcher Survey 12

SAGE Integration: DevOps MVP 13

Continuous Integration (CI) and Continuous Deployment (CD) 13

Integration of CI, CD, Version Control System (VCS), and work management in TFS 15

Quality control 15

Development Velocity 16

Environment Stability and Isolation 16

System Capabilities 17

SAGE Feasibility Study 17

Documentation **18**

Future Work **21**

Conclusion **23**

References **24**

Introduction

As the number of SAGE researchers, and consequently concurrent active SAGE workstreams, has grown between Fall 2017 and Spring 2018, and will likely continue to grow, an increased focus on the overall coherence of the system becomes increasingly important. Competing priorities, time and resource constraints, and limited attention to process are critical factors in team efficiency, resulting in decreased progress toward team goals, and limiting the productivity increases that should be associated with team growth (Holton, 2004).

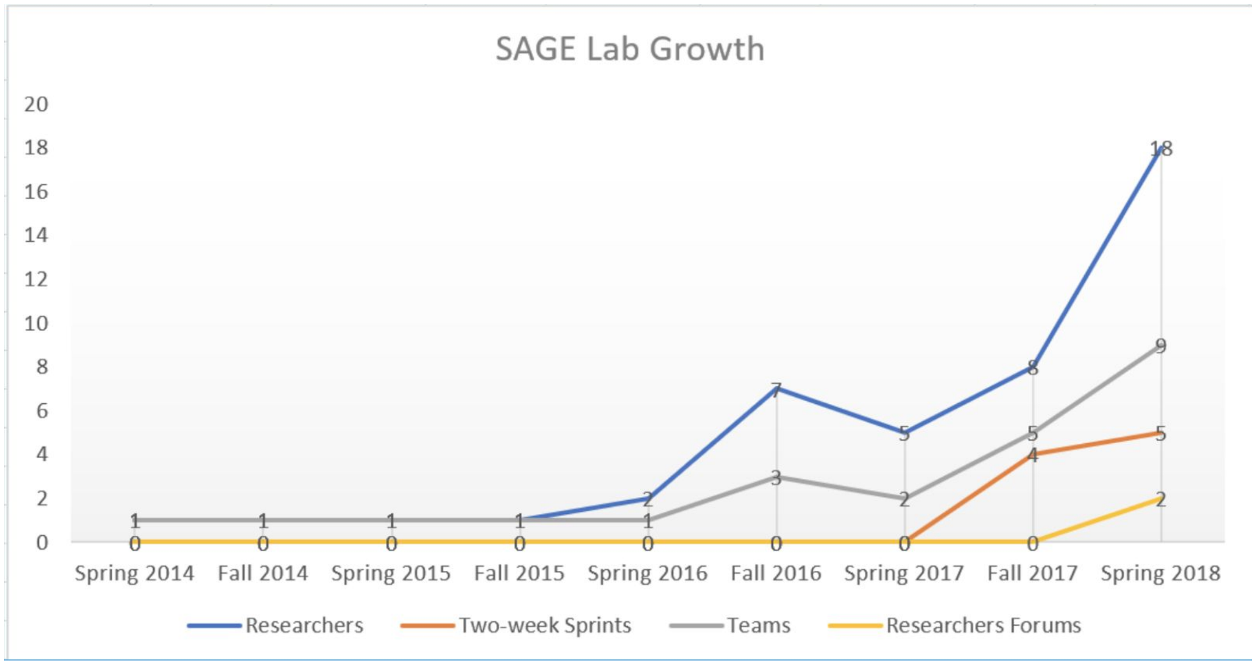


Figure 1: SAGE Lab Growth across several metrics

Our project's aim has been to unify and create a process and set of control mechanisms for the multiple SAGE workstreams, with an aim to increase research team productivity in order to accelerate the two immediate SAGE milestones ahead: 1) administer and summarize public surveys and field studies on feasibility and effectiveness of SAGE research, and 2) publish a feasibility study that describes the theoretical and empirical underpinnings of SAGE. We accomplished this through the completion of some features in the SAGE Integration, and Experimental Design and Publication Strategy Epics, as follows:

Epic	Feature	Aim
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

Figure 2: Sage Field Study Support Spring 2018 Aim

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.

SAGE Integration: Workstream Integration

SAGE as a Growing Research Project

Our work this semester extends prior semesters' SAGE Gameful Assessment integration work to integrate new workstreams and research goals in SAGE, including field studies and publication, enhance the coherence of SAGE's feature set, and present a more unified user experience to instructors, students, and researchers. Due to heightened interest in SAGE's earlier works, the Spring 2018 semester saw the highest influx of researchers to SAGE, reaching 18 active researchers (see Figure 1). Coordination across multiple research teams and workstreams was critical to the success of this semester's field studies and feature development. We applied a light-weight Agile project management methodology and techniques to enhance collaboration in a distributed team to ensure the various research and implementation trajectories remained cohesive.

Integration of Workstreams Within a Growing Research Team

Agile methodologies have historically been widely applied to smaller software projects with simple governance rules, and large organizations are actively, experimentally applying agile methods on larger enterprise projects. There are conflicts between agile methods and principles and traditional software development in large bureaucratic organizations (Pmi.org., 2018). PMI highlights the team self-organization that are inherent to agile projects as a benefit to small and growing teams. This methodology is particularly applicable to SAGE, which is currently growing in its number of researchers, feature set, and exposure to users (via field studies).

In Brotherton, et al.'s research, project success may be attributed specifically to the use of a Work Breakdown Structure (WBS) (Brotherton, 2008). SAGE research tasks and objectives were broken down into Epics, Features, and Tasks to form a WBS that was used to measure and facilitate research progress this semester.

Attention to process and management of conflicting priorities, time, and resource constraints are critical to productivity in a growing team (Holton, 2004). Our overall focus this semester was maximizing the productivity of the growing research team through process improvement, automation, tooling, project management, facilitation and support, further developing collaborative documentation, and creating shared development environments through continuous deployment.

GitFlow

Per Phillips' survey on branching and merging practices (Phillips, 2011), Git was used by the second highest number of survey participants (narrowly less respondents than were using Subversion), and some of the primary determinants of success, or satisfaction with, a branching strategy were 1) minimizing merge conflicts, 2) increasing the frequency of upstream merges, and 3) using Experiment, Feature, and Release branches. GitFlow (Driessen, 2010) is a branching and merging workflow, using Git, that minimizes merge conflicts, enables Continuous Integration systems to perform frequent upstream merges, and accommodates experiment, feature, and release branches. As a result, we selected GitFlow as SAGEs branching model.

SAGE Integration: DevOps MVP

Existing SAGE DevOps Infrastructure

The existing DevOps infrastructure included repositories hosted on Github and a shared development environment, with manual builds and deployment of SAGE binaries. Compute, storage, and database infrastructure were, and continue to be, provided by mLab and Azure. There was some pre-existing, but unused, work on build automation via Gulp and Bower, and test suite management via Mocha.

Dimensions of DevOps

Lwakatare, et al. (Lwakatare, 2015) identify four main dimensions of DevOps: collaboration, automation, measurement, and monitoring. It also introduces a conceptual framework for characterizing DevOps work. While this paper is essentially a survey, and does not present a reference architecture or formal statistical analysis of DevOps practices, it was very useful in conceptualizing our approach to this semester's DevOps work and identifying opportunities for future work.

Collaboration

The paper focuses on the collaboration between development and operations organizations, but within SAGE, there is no such distinction. Therefore, we chose to implement collaboration improvements through continuous integration and deployment, and improved communication via Slack and email, as well as various touchpoint meetings ranging from 1:1 meetings, research team meetings, and all-hands researchers forum.

Automation

Continuous integration, deployment and testing are highlighted as common best practices in DevOps, and “Infrastructure as a Code” (IaC) is discussed as a component of common automation infrastructure. Improved development velocity, system stability and predictability, and improved software quality result from implementing automation. Our approach was to begin with continuous integration via TFS, and use that as a basis for implementing continuous testing and deployment, also via CFS. IaC implementation was begun, in the form of build automation using Gulp and Gradle, and opportunities for further IaC work on configuration management is discussed in Future Work.

Measurement

Measurement in this case refers to the measurement of the software development process itself, and is distinct from Monitoring, which refers to monitoring the system under use for system metrics and issues. As SAGE is not yet available publicly, and this paper emphasizes measurement of usage and performance of software in a production environment, we chose to begin by measuring the rate of commits to the repositories as a proxy for increased development productivity.

Monitoring

Monitoring system health involves gathering and analyzing performance metrics, logs data, and system metrics (e.g. process failures) to quickly identify and correct problems in a live system. Because SAGE is not available to end users, we did not focus on this dimension, but discuss opportunities to implement monitoring in Future Work.

Development and Deployment at Facebook

Feitelson, et al. (Feitelson, 2013) emphasize continuous deployment, personal responsibility of each developer for their own changes, collaborative documentation, and the balance of risk against development velocity, and these are the elements that we focused on. A key principle discussed that we did not address is testing on real users at scale via A/B testing, something that's not possible with SAGE as a pre-production system.

According to the paper, increased deployment frequency implies smaller change sets in general for each deployment, reducing the risk of regressions or new problems, and making debugging easier. Individually testing each commit to the code base also helps with change isolation and reduces the flakiness of improperly encapsulated tests.

Continuous deployment also reduces the time it takes for an individual contributor to have an effect on the system, improving that contributor's confidence and accelerating their path to productivity.

The authors' approach utilizes a code review mechanism that includes a check for relevant documentation in their wiki. SAGE chose not to implement code reviews early in the semester, as the increased overhead was not worth the increased system stability in a pre-release system. However, SAGE has implemented a collaborative documentation wiki, and emphasizes documentation, as discussed in Workstream Integration.

Because of the rapid rate of SAGE's development and pre-release status, we made a choice to emphasize development velocity over risk minimization. As mentioned in this paper, the impact of system regressions or failures, and the likelihood of these events increases with scale as defined by the number of users, developers, and lines of code. Because SAGE is early in the development lifecycle and has not been released to users, the impact of these events is minimal, their likelihood is low, and time to resolve issues is not a significant factor.

Survey, Field Study Design, and Publication Strategy

Recent researches (Lee, et al 2013; Ihantola et al, 2016) show there is a continuing shortage of research 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) debriefing, a highly effective teaching method, is challenging to administer, and 3) there is a recognized higher risk for students to game or cheat the system. Another research (Harteveld, et al 2014) further boosts motivation with its affirmation 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. This semester's SAGE project then selected surveys, field studies, and user-centered design (UCD) sessions as research methods of choice to gather data for further analysis and refinement of the various SAGE research areas.

There are nine CT aspects (Mannila, et al 2014) highlighted by the CSTA/ISTE (data collection, data analysis, data representation, problem decomposition, abstraction, algorithms, automation, simulation and parallelization). These CT aspects are included as part of the Spring 2018 SAGE survey that we further extracted and analyzed via various Tableau Dashboards.

Accomplishments

Over the course of the semester, our team has significantly improved research and development velocity for the entire team, and began work on a near-term publication of a study on SAGE Feasibility. We have focused on introducing automation into the development and deployment workflow, integrating and stabilizing the development and shared environments, adding additional capabilities to the system and shared environments, integrating the workstreams of the various research teams, and facilitating execution, data gathering, and analysis of field studies. We've also kept future research in mind, and set the groundwork for additional improvements outlined in Future Work.

SAGE Integration: Workstream Integration

Online, Persistent SAGE Instance for Field Studies

We were able to setup an online, persistent, web-hosted SAGE environment (dev.cu-sage.org, later cloned into uat.cu-sage.org) for the first time, a natural next-step evolution from the local environments that past researchers have relied on in the past. The online SAGE environments become instrumental in supporting multiple field study and UCD sessions conducted throughout the semester, as researchers are able to perform demonstration of SAGE functionalities directly on the online instance of SAGE. Part of this work involved the creation and maintenance of end-to-end flows of SAGE's UI/UX, through unified layout designs and cohesive brand identity for example through a consistently placed SAGE logo in the welcome page and throughout SAGE web pages that is deliberately color-coordinated across the web pages.

Figure 3: SAGE logo

Updated Data Model Diagrams

We also updated the SAGE data model that was first created last semester, to aid newly incoming researchers in visualizing how the different data models throughout SAGE modules are interconnected with each other. We also added an informational box showing which data models are no longer relevant with the current state of SAGE. We have received positive feedback from various other teams of the usefulness of this diagram because not all deprecated data models have been removed in the codebase, which prompted them to refer to this Data Model diagram for clarity during their development efforts.

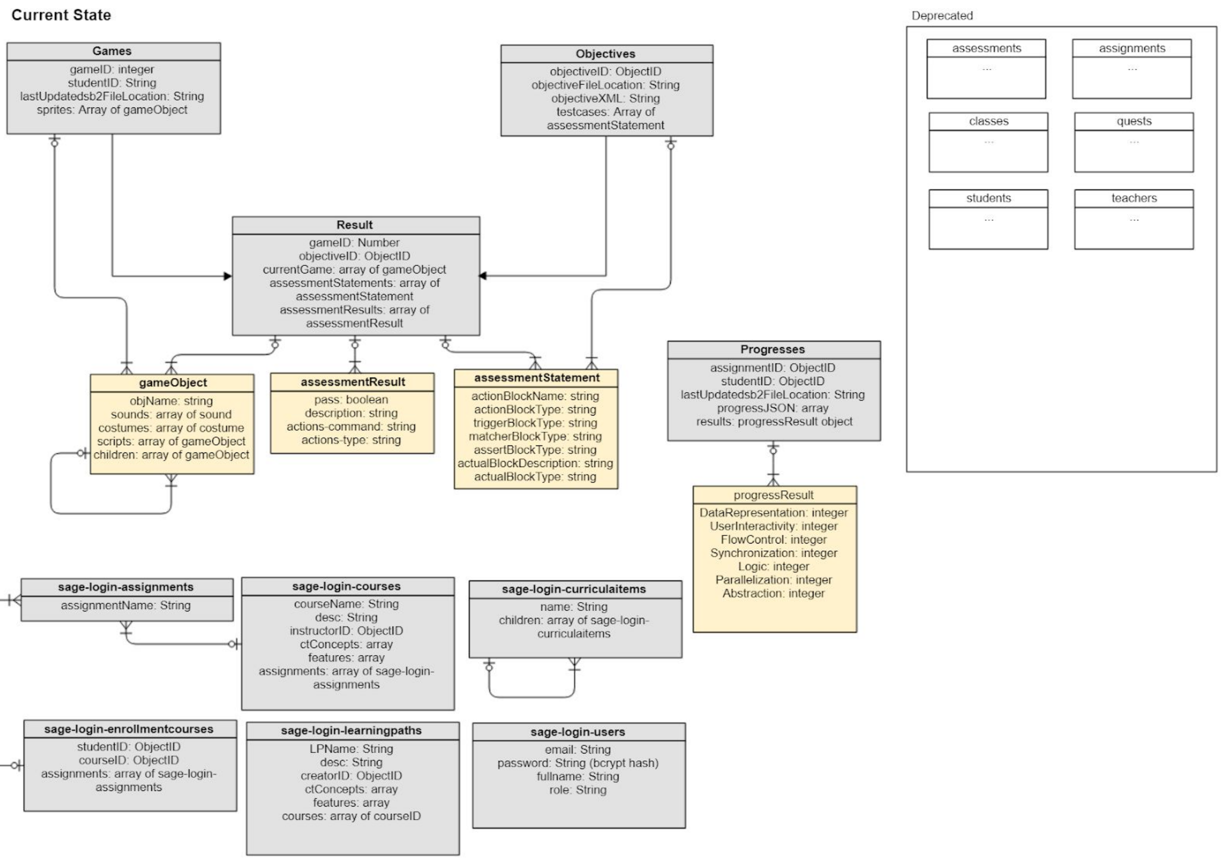


Figure 4: SAGE System-wide Data Model

Software-tier Reference Diagram

Similarly, we created a software-tier Reference Diagram (Figure 5) to comprehensively represent the different SAGE components, functionalities, and features that have been developed in the last four years. Present and future researchers, as well as readers of future SAGE publications, could utilize this systematic view as reference for their respective research, to be repurposed, replicated, and extended as desired.

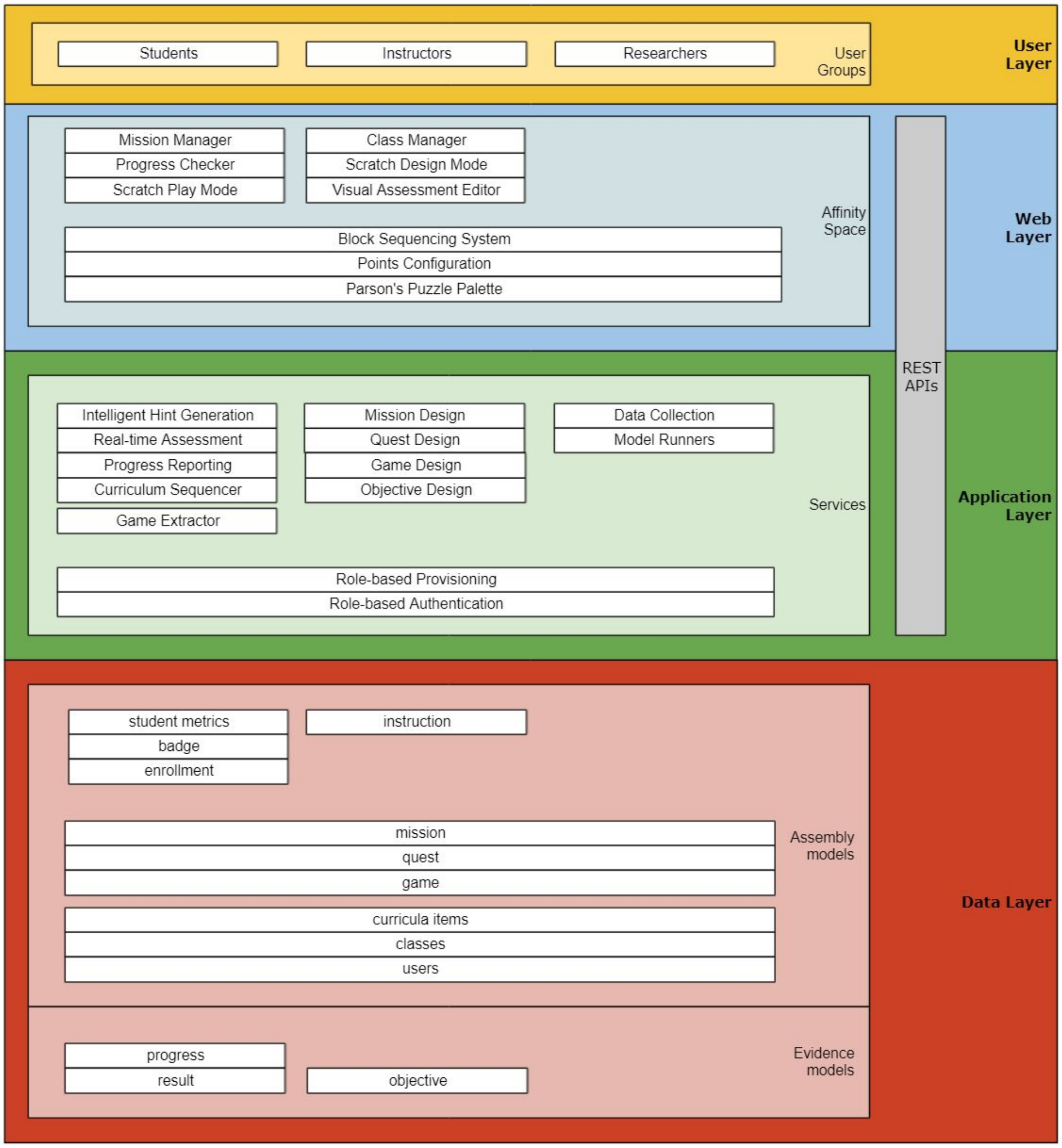


Figure 5: Software-Tier Reference Diagram

Activity Flow Reference Diagram

In Figure 6, we graphically represent another take of how to understand SAGE: from the activity flow perspective. In this diagram, the four different SAGE research areas (Gameful Affinity Space, Gameful Direct Instruction, Gameful Constructionism, and Gameful Intelligent Tutor) are represented separately and in relation to each other. This enhances readers' comprehension on how the main activities flow through different processes. We also added a legend to serve as visual aid indicating the areas impacted by the research of the nine Spring 2018 teams by looking up the team name and its designated color-coded star symbol (★).

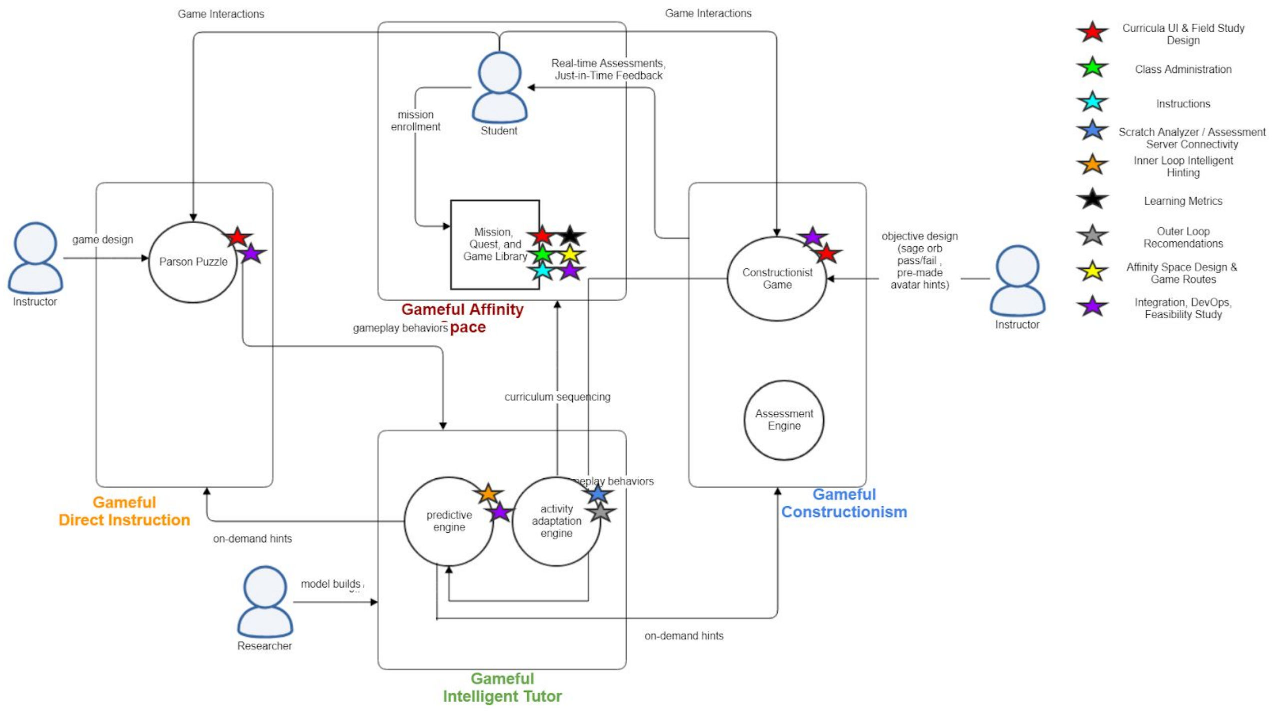

Game Interactions
Game Interactions
Real-time Assessments, Just-in-Time Feedback
mission enrollment
Student
Mission, Quest, and Game Library
Gameful Affinity Space
gameplay behaviors
Instructor
game design
Parson Puzzle
Gameful Direct Instruction
on-demand hints
curriculum sequencing
predictive engine
activity adaptation engine
gameplay behaviors
model builds
Researcher
on-demand hints
Gameful Intelligent Tutor
Constructionist Game
Assessment Engine
Gameful Constructionism
objective design (sage orb pass/fail, pre-made avatar hints)
Instructor
Curricula UI & Field Study Design
Class Administration
Instructions
Scratch Analyzer / Assessment Server Connectivity
Inner Loop Intelligent Hinting
Learning Metrics
Outer Loop Recomendations
Affinity Space Design & Game Routes
Integration, DevOps, Feasibility Study

Figure 6: Activity Flow Diagram

End of Semester Researcher Survey

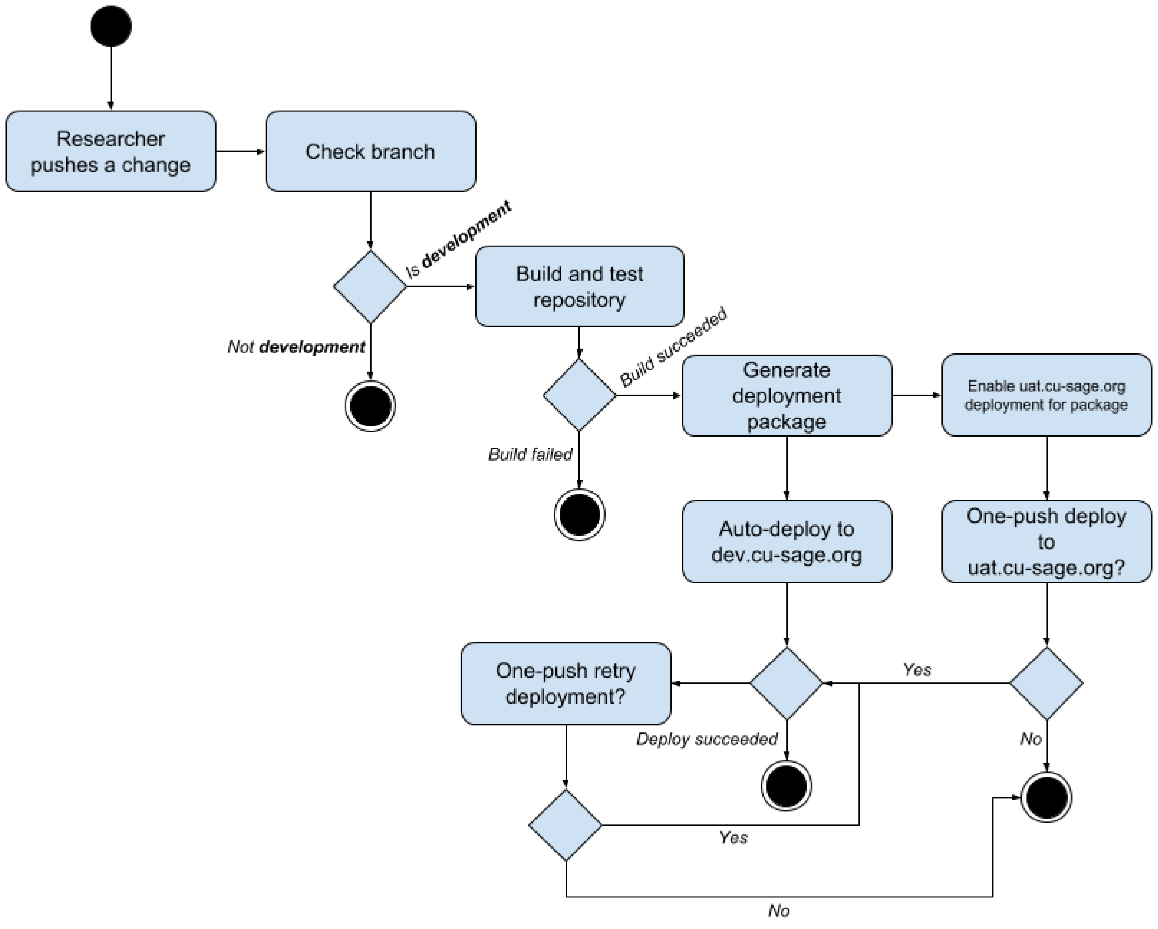
In the spirit of continuous learning and improvement, we have created and administered an end-of-semester Spring 2018 Research Survey to gather feedback from researchers on the effectiveness of SAGE labs. The survey will grow in usefulness if the end-of-semester administration becomes a recurring practice. The Spring 2018 survey is available here: https://docs.google.com/forms/d/1Ei1GAbWSW_11Tq74Ew4Tgv3bfOA3daf5HPSMiBLrNBk/prefill

SAGE Integration: DevOps MVP

Continuous Integration (CI) and Continuous Deployment (CD)

We introduced “Push on Commit” by implementing a multi-component CI and CD system with the following workflow:

1. When a researcher pushes a new commit (changeset on the codebase) to our centralized remote repository, that commit is detected by our CI system.
2. The CI system runs a customized set of actions per repository, which include “building” (or starting, in the case of Node.js repositories), running the entire test suite, and packaging a deployment artifact.
3. When a new deployment artifact is created, it is detected by our CD system.
4. The CD system performs environment cleanup, preparation (including stopping servers), deployment (including deployment of dependencies such as the latest version of a sage-scratch swf binary), and activation of the new artifact (starting IIS, restarting SMTP servers, etc.) in the shared development environment.



Researcher pushes a change
Check branch
Is development
Not development
Build and test repository
Build succeeded
Build failed
Generate deployment package
Enable uat.cu-sage.org deployment for package
Auto-deploy to dev.cu-sage.org
One-push deploy to uat.cu-sage.org?
One-push retry deployment?
Deploy succeeded
Yes
No
Yes
No

Fig. 7: DevOps workflow in SAGE

Dependencies between repositories, such as changes to sage-frontend that affect execution of the sage-scratch swf, are managed by deploying the latest version of all related deployment artifacts at the same time. In practice, this means that if one of a set of related repositories is deployed (e.g. sage-scratch and sage-frontend), then the latest artifacts built from each of the other repositories in that set is also deployed. This means that a change in sage-frontend that breaks a feature in sage-scratch wouldn't be deployed alone; as long as dependent changes are also pushed to sage-scratch, the shared environment would not be broken.

Because our test codebase is not mature enough to warrant failing builds on test failures, we currently execute all tests, but only display a warning for broken tests, and let the build succeed. However, the system as deployed could be configured with a single change in TFS that would result in failed builds if any test case fails, or, optionally if test failures exceed a configurable proportional threshold.

To facilitate CI and CD, we deployed a managed web server (IIS) to the development and User Acceptance Testing (UAT) environments. CI and CD were implemented with a combination of Team Foundation Server (TFS) tasks, build automation via Gulp, Gradle, and NPM, test automation via Mocha and Chai, and introduction of environment specific configurations in each repository; i.e. config option values for the binary in “development” may differ from “local” and “uat,” without requiring separate builds for each environment.

Integration of CI, CD, Version Control System (VCS), and work management in TFS

At the start of this semester, VCS was hosted by GitHub, and work was managed in TFS; CI and CD had not yet been implemented. We decided to implement all of these components within TFS by migrating our existing repositories to TFS, and setting up TFS-based CI and CD. There a multiple advantages to this approach:

1. Commits can be associated with TFS work items by mentioning the work item ID in the commit message.
2. Commits can be automatically built on push, without requiring polling of an external system (Github).
3. Deployment can be automatically triggered by a successful build.
4. The configuration of all of these deployments can be exported in a single format, i.e. as TFS configurations, allowing for rapid deployment to a replacement TFS instance for disaster recovery.

Quality control

We implemented testing in the Node.js (sage-node, sage-frontend) and Java (scratch-analyzer) repositories, using Mocha, Chai and JUnit. SAGE’s approaches to using these test frameworks were documented in the wiki (see Documentation section), and the test suites were included in our Gulp and Gradle based build automation system.

This implementation allowed us to integrate test execution into our CI system. Currently, we have not chosen to fail builds based on failed tests or poor test coverage. The potential for introducing build failures on test / coverage regressions is outlined in Future Work.

Development Velocity

We added build automation to all repositories, using Gulp and Gradle, in order to allow researchers to build and test changes locally before pushing them to the central repository, and thus the shared development environment. Our Gulp and Gradle implementations include test tasks, allowing researchers to run the entire test suite for each repository locally before pushing changes. We are also using Gulp and Gradle (as well as NPM) for dependency management for each repository; dependencies are automatically downloaded when a build is executed, greatly simplifying the initial development environment configuration for a new researcher.

Our migration from GitHub to TFS also included merging multiple long-standing feature and team branches back into the upstream development branch. By reducing the number of unnecessary divergent branches, we avoided large merge conflicts later in the development cycle, and were able to informally implement GitFlow as a branching and merging strategy. Currently, all repositories contain only a central development branch, a deployable “master” branch, and feature / experiment branches that should be easily merged into development at the end of the semester.

Environment Stability and Isolation

We created a separate User Acceptance Testing (UAT) serving environment and database, to ensure a stable environment was available for field studies. A separate mongoDB mLab environment was created by cloning the development database, and a separate virtual server was set up to serve the UAT web components. While releases were pushed to the development environment on each successful automatic build, UAT deployments had to be manually triggered. The deployment itself was automated through TFS’ release automation, but required a project administrator to manually trigger a deployment within TFS. This allowed the UAT environment to remain stable and unchanged, while our repository continued accepting and building commits during field studies.

System Capabilities

We implemented an SMTP server in our development environment, allowing for SAGE notifications via the Class Administration feature of the Gameful Affinity Space, and enabling future research teams to implement notification features. This capability could also be leveraged by our automation systems to push build breakage notifications, test results, and other continuous integration artifacts and reports.

We replaced the native Node.js servers running in user processes in our shared environments with IIS websites, running Node.js servers via IIS node (Tjanczuk, 2016). This allowed our TFS-based deployment systems to automatically deploy SAGE using common components available in Microsoft's TFS marketplace. It also allowed for simpler implementations of environment-based configurations, allowing us to deploy a single build package to multiple environments by embedding the environment specific configurations for all environments in the build package, and detecting the current environment via iisnode.

SAGE Feasibility Study

In addition to creating a persistent, online SAGE environment to support the Feasibility Studies, we also documented summarization of an extensive academic articles (including some Field Study-related articles), and made them available as reference for our researcher community at the wiki's Theoretical Foundation section. We also completed the training (TC0087 - Human Subjects Protection (HSP) Training) necessary for us to participate in the field studies.

As we start receiving studies and survey materials, we contribute in the organization and analysis of these data (see Spring 2018 'Curricula Analysis and Field Study Design and Publication - Final Report'). Overall, we have created six Survey Analysis Dashboards using Tableau: 1) Demographics, 2) Computational Thinking, 3) Parson's Puzzles, 4) Intelligent Tutoring Systems, 5) Constructionist Video Games, and 6) Direct Instruction.

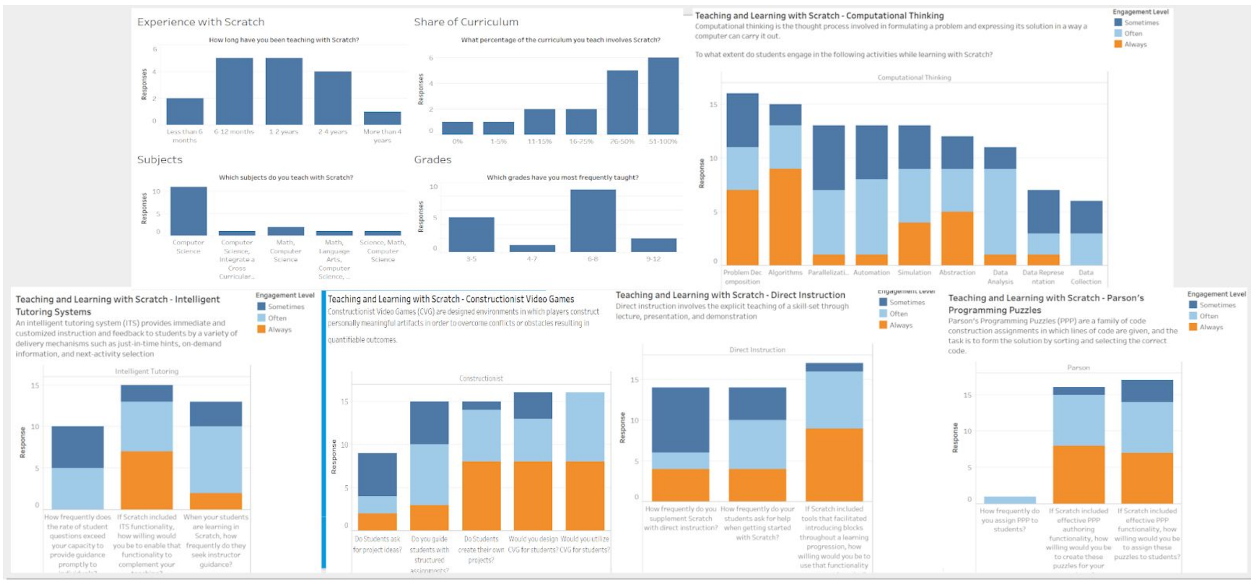


Figure 8: SAGE Spring 2018 Survey Dashboards generated from Tableau

We offered our analysis of these dashboards, including suggestions on how to better enhance the survey going forward, as part of the 'Curricula Analysis and Field Study Design and Publication - Final Report'. Because we anticipate a lot more respondents (up to 200 respondents) in the next several months and the introduction of future versions of the survey to continue being used to draw survey data, we suggest future researchers to reuse our data processing and data analysis process to efficiently obtain, transform, and present the data using a pre-generated Tableau workbook specifically created for the Spring 2018 SAGE survey (see *Survey Data Extract, Transform, and Report Generation using Tableau* in the Documentation section below).

Documentation

The Spring 2018 Semester SAGE Research Team has continued to add new knowledge and content onto the the SAGE Wiki that was originally created in Fall 2017. An export of the wiki into PDF now contains 135 pages, an increase of 52 new pages from the previous semester. This semester the team also added a separate SAGE DEV Wiki to house development and operation-related content for the project. For the purposes of this report, new documentation on DevOps workflows and infrastructure, shared environments, workstream integration, and survey, field study, and publication preparation and execution are available in the SAGE wiki at the following locations:

1. *Survey Data Extract, Transform, and Report Generation using Tableau;* https://gudangdaya.atlassian.net/wiki/spaces/SAG/pages/397606917/Survey+Data+Extract+Transform+and+Report+Generation+using+Tableau
2. *Sending Email from SAGE via dev SMTP server*; https://gudangdaya.atlassian.net/wiki/spaces/SAGE/pages/332988422/Sending+Email+from+SAGE+via+dev+SMTP+server
3. *Node.js Testing*; https://gudangdaya.atlassian.net/wiki/spaces/SAGE/pages/303857665/Node.js+Testing
4. *Java Testing*; https://gudangdaya.atlassian.net/wiki/spaces/SAGE/pages/323059880/Java+Testing
5. *Shared Environment Configurations*; https://gudangdaya.atlassian.net/wiki/spaces/SAGE/pages/233373697/Shared+Environment+Configurations
6. *Researcher Forum;* https://gudangdaya.atlassian.net/wiki/spaces/SAG/pages/208306177/Researcher+Forum
7. *System-Level Data Model*; https://gudangdaya.atlassian.net/wiki/spaces/SAG/pages/236224552/System-Level+Data+Model
8. *Semi-structured Interview Protocol*; https://gudangdaya.atlassian.net/wiki/spaces/SAG/pages/225673234/Semi-structured+Interview+Protocol
9. *SAGE Feasibility Draft*; https://gudangdaya.atlassian.net/wiki/spaces/SAG/pages/231899166/SAGE+Feasibility+Draft

10. *Migrating repositories from GitHub to TFS*;
https://gudangdaya.atlassian.net/wiki/spaces/SAGE/blog/2018/03/05/254476289/Migrating+repositories+from+GitHub+to+TFS

11. *Getting Started (SAGE Reference Architecture Diagram, SAGE Activities Diagram)*;
https://gudangdaya.atlassian.net/wiki/spaces/SAGE/pages/60293127/Getting+Started

12. *SAGE Reference Architecture Diagram*;
https://gudangdaya.atlassian.net/wiki/plugins/servlet/ac/com.gliffy.integration.confluence/gliffy-confluence-fullscreen-viewer-launcher?content.plugin=ac%3Acom.gliffy.integration.confluence%3Agliffy-diagram&space.key=SAGE&content.id=365428747&content.version=1&space.id=46473989&content.type=custom

13. *SAGE Student Guide*;
https://gudangdaya.atlassian.net/wiki/spaces/SAGE/pages/141819905/SAGE+Student+Guide

14. *User Guide*;
https://gudangdaya.atlassian.net/wiki/spaces/SAGE/pages/141721601/User+Guide

15. *SAGE Papers*;
https://gudangdaya.atlassian.net/wiki/spaces/SAGE/pages/209911834/SAGE+Papers

16. SAGE Theoretical Foundation;
https://gudangdaya.atlassian.net/wiki/spaces/SAGE/pages/209911834/SAGE+Papers

Future Work

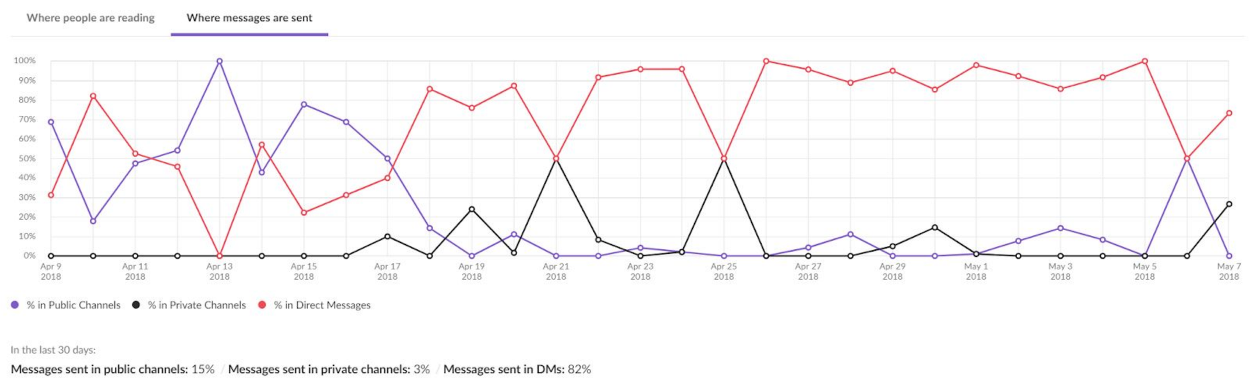
Our work this semester has surfaced a large amount of opportunities for additional work within DevOps, workstream integration, and publication / study preparation and execution in future semesters. These include:

1. **Intelligent Hinting - Chatbot POC** - Use our trained model as a decider in an n-gram analyzer to create a proof-of-concept chatbot, that would provide intelligent hinting to students on-demand through a natural language chat interface.
2. **Configuration management** - Automated setup for shared and local dev environments, performed through Ansible, Puppet, or Chef to accelerate future research teams and allow for repeatable environment builds, and configuration-as-code for shared environments (Dev, UAT, Prod).
3. **Interactive Intelligent Hinting Analysis and Framework**
4. **AsUnit testing**
5. **Newman (Postman Collection Runner) integration**
6. **Linting and pre/post-commit hooks**
7. **Test coverage and "publish-on-green"**
8. **SAGE code standards**
9. **Code review integration into the CI pipeline**
10. **Production environment**

    A production environment is not necessary until SAGE is released to end users or research subjects. However, creating infrastructure and implementation plans for a cloud-provider agnostic production environment would improve SAGE's disaster recovery time, and may uncover inefficiencies or undocumented dependencies on the current development and UAT environments in Azure.
11. **Scheduled UAT builds**

    UAT deployments are currently manually triggered. Finding an optimal UAT build schedule and pushing "N-day'ly" builds would reduce the risk that valid builds are accidentally skipped in UAT, and would reduce the overhead on project administrators, who are currently required to manually trigger deployments.

12. Communication optimization



In the last 30 days:
Messages sent in public channels: 15% / **Messages sent in private channels:** 3% / **Messages sent in DMs:** 82%

The Slack **#general** channel, to which all researchers are subscribed, had much lower utilization (15-18% of overall messages, according to Slack Analytics) than direct messages (82%) this semester. More utilization of the **#general** channel, in particular to announce teams' progress and feature updates, would improve the visibility of new work across the research team. To further enhance cross-pollination and exposure of new features across teams, messages to **#general** could be generated automatically with every TFS build. Increased shared demos, and ad-hoc demos (via WebEx) of new features, would also improve information sharing within the team.

13. Project management and WBS improvements:

Commits are not currently tied to work items. We should update the researcher workflow to include work item IDs in commits. This is supported by TFS, and would have the added benefit of allowing researchers to close (or otherwise change state) on a work item via a commit message, instead of requiring them to also log into and interact with the TFS work interface.

14. Use Cases and Personas

Some SAGE use cases aren't documented. It would benefit future researchers if a use case / user persona inventory is created to guide future work.

15. Project management methodology

Our project management methodology is not fully formalized or documented. Building an agile project schedule for research in future semesters, with regular deliverables, would improve visibility of work across research teams. Documenting our project management methodology would reduce the time to productivity for new researchers, and the increased understanding across the team should improve productivity and reduce project management overhead, such as closing WBS tasks, in future semesters.

Conclusion

Our work this semester focused on researchers productivity, automation, process, and preparation for SAGE's public introduction in the form of surveys, field studies, and research publication. We believe our work significantly accelerated the progress of the research team, facilitated the successful completion of our surveys and field study this semester, laid the foundation for publication in 2018, and set up future semesters' researchers for decreased time to productivity, and increased overall productivity.

Our work also identified multiple opportunities for future work, and provided a conceptual framework for managing SAGE's multiple workstreams via process management, automation, and tooling. We believe that this will be increasingly beneficial to future semesters' research teams as the project continues to progress. We thoroughly enjoyed the research variety and work effort of our fellow Spring 2018 researchers, and would like to thank all of them for an enjoyable and educational semester, with special mentions to Jeff Bender, Harsimran Bath, and Lalitha Madduri.

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]

Driessen, Vincent. 2010. “A Successful Git Branching Model.” Nvie.com, nvie.com/posts/a-successful-git-branching-model/

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

Holton, J. A. (2004). *Building trust and collaboration in a virtual team*. National Library of Canada = Bibliothèque nationale du Canada.

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.

Mannila, L., Dagiene, V.,Demo,B., Grgurina, N., Mirolo, C., Rolandsson, L., and Settle, A.. 2014. Computational Thinking in K-9 Education. In Proceedings of the Working Group Reports of the

2014 on Innovation & Technology in Computer Science Education Conference (ITiCSE-WGR '14), Alison Clear Clear and Raymond Lister (Eds.). ACM, New York, NY, USA, 1-29.

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].

Brotherton, S. A., Fried, R. T., & Norman, E. S. (2008). Applying the work breakdown structure to the project management lifecycle. Paper presented at PMI® Global Congress 2008—North America, Denver, CO. Newtown Square, PA: Project Management Institute.

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).