KBase Automation using Selenium

Design Document

SD-May21-26
Dr. Myra Cohen: Product Owner,
Jake Veatch: Developer,
Daniel Way: Developer,
Daulton Leach: Scrum Master,
Caleb Meyer: Developer,
Hunter Hall: Developer,
Sergey Gernega: Developer
sdmay21-26@iastate.edu
Team Website

Revised: 9-4-2020

Executive Summary

Development Standards & Practices Used

The project will follow the Agile development methodology. It will have an iterative development process where requirements and ideas emerge from collaboration among team members. The engineering standards that apply to this project will include: consistent syntax for filing naming and code styles, the utilization of automation where possible to reduce human error, the use of industry-standard languages and frameworks, Test Driven Development(TDD) that ensures work collaboration, and the use of S.O.L.I.D architecture principles. To insure the project is moving forward we will incorporate weekly advisor meetings, weekly team meetings, the use of slack, and the use of trello for our communication standards.

Summary of Requirements

- Configure Selenium to interface with/run KBase apps
- User set parameters to initiate automation using Selenium
- Intelligently sample the configuration space from set parameters
- Automate the testing of a certain sample configuration
- Use free and open source tools to reduce cost
- Keep the project itself as open source

Applicable Courses from Iowa State University Curriculum

Com S 227, Com S 228, Com S 309, Com S 327, Cpre 288, Com S 311, SE 319, SE 329, SE 339, ENGL 314, SP CM 212

New Skills/Knowledge acquired that was not taught in courses

- Configuration and use of Selenium testing software
- Use of KBase software
- Automating interactions with KBase software
- Automated JUnit tests through Github
- Maven Checkstyle tests to monitor code syntax

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1 Introduction

1.1 Acknowledgement

The team would like to take this opportunity to thank Dr. Myra Cohen for her significant contributions towards the project and our team's success. Dr. Cohen consistently aided the team's efforts and provided guidance on how to progress the project. Dr. Cohen is an expert in the field and without her vision and knowledge of the task at hand the team would not be where they are today. We want to take this time to thank Dr. Cohen for her efforts and contributions to make our team and project successful.

1.2 Problem and Project Statement

1.2.1 Problem Statement

The following quote from Dr. Cohen describes our project's problem statement precisely and readable by all users: “KBase is a software and data science platform designed to meet the grand challenge of systems biology: predicting and designing biological function. Many of the tools in this computational biology ecosystem are configurable, meaning that users can manipulate how the tool works by changing the various settings. While changing configuration options may improve a tool’s performance, it can also lead to program failures. The project will build a GUI testing framework using an automated testing framework such as Selenium, to (1) infer the configuration model; (2) intelligently sample the configuration space; and (3) automatically test a sample of configurations.” (Dr. Myra Cohen).

1.2.2 General Solution approach

The team prototypes a few options for approaching this problem. We landed on using a Selenium web driver to take requests from a GUI containing all configurable options for the Kbase tests. The web driver opens up a browser window of the users preferences then automatically fills in the variables on the Kbase website. Kbase has an exponential combination of potential variable inputs, giving us a fair amount of testing to run. Due to this the user can run our application once and fill parameters from a variety of methods for a variety of different tests. Methods include, reading from a file, randomized, or input by hand. Our program captures the output logs and other relevant information from failing tests and reports that to the Kbase developers. Doing this will show us the errors within the Kbase application and allow us to improve it via our testing. This is the purpose of our project. The number of combinations to test is exponential, so Kbase needs a way to test their software automatically and report the errors to their developers. The project is important to all Kbase users. The users we have spoken to have reported errors in the software. With our automated testing and output log collection we can define what exactly goes wrong to replicate errors to the Kbase team. Without the data on what went wrong the developers of Kbase are left in the dark and highly unlikely to replicate and solve the bugs. Leveraging Selenium and automated testing we can run as many tests as the Kbase system can handle at any given time.

1.2.3 Project Output

At the end of the project we hope to have a running standalone application that can be used and built off of to test and run FBA analyses on Kbase automatically. Since we only have a year we are keeping the project fully open source to allow the next team to pick up where we leave off to continue improving and propelling the automated testing process further and in turn creating a better Kbase application.

1.3 Operational Environment

The end product is expected to be run and available constantly to any Kbase users. The environment the software application runs in will be the user's own system. Our application will be downloaded by each Kbase user on their own machines. Our application is purely software therefore the environment is limited to the user's system. The only hazard that our project could encounter is no internet connection for the user. Our application needs an internet connection to operate. This environmental hazard is out primarily out of our control as developers.

1.4 Requirements

Functional

- Configure Selenium to interface with KBase apps
- User sets parameters to imitate automation using Selenium
- App will sample the configuration space from set parameters
- Automate the testing of certain sample configurations
- Will collect collect output data from KBase
- Users will be able to randomize inputs
- All inputs from the Kbase GUI will be available

Non-Functional

- Will not noticeable disrupt Kbase traffic flow
- App Gui will have similar layout as Kbase GUI
- App will have a dark mode

Environmental

- Will require a valid and active KBase account
- Will require a internet connection
- Must be run on device which supports Selenium

Economic

- Will be open source and use open source resources

1.5 Intended Users and Uses

The intended users of this application will be those who are already using the KBase GUI. The purpose of this application is to allow the automation of the KBase GUI, therefore it is expected that the users of this app will be family with the KBase system and how to use it. No experience beyond what is already required for KBase operation will be needed in order to operate the app.

The standard use of this app will be that once the user opens it they will enter in their selenium credentials. The app will then open a web driver and log into selenium on their behalf. Once logged in the user will be able to select a narrative they wish to automate. Once all inputs have been configured for the narrative the app will then run the Kbase narrative and collect output for each set of configurations of the user input.

1.6 Assumptions and Limitations

Assumptions

- A user will have a working Kbase account.
- A user will be familiar with Kbase.
- A user will have a working internet connection.

Limitations

- Will not store a user's Kbase credentials, in order to avoid security risks.
- The product will be open source and not cost users to use.
- The app will not store collected data in a database.

1.7 Expected End Product and Deliverables

There will be one deliverable. A stand-alone application that uses Selenium to interface with the KBase GUI. This app will have an interface GUI that will be similar in presentation as a KBase narrative GUI, but with the additional options of presetting multiple configurations to run including randomizing variables as well as auto configuring combinations. The app will run these configurations through the KBase system on the user's behalf by opening a web browser of the users choosing and configuring the KBase narrative based on the configurations input by the user. Any resultant data the a KBase Produces will be collected by the system and stored as a file.

2 Project Plan

2.1 Task Decomposition

KBase Interface Analysis

Analysis of the KBase web application to better-understand the user's workflow and how the webpages are built. This will inform how we might interact with and automate the interface. Particular deliverables from this task may include specification of workflows for interaction with KBase and identification of particular DOM selectors or risks we could encounter as we automate.

Automation Technology Analysis

Analysis of automation technologies like Selenium to understand which might best-fit our use-case. This will include considerations like the performance of the tool (can we parallel jobs?), support of the tool (does it accommodate modern browsers and devices?), and ease-of-use (will interaction with the framework require programming knowledge?). Deliverables should include identification of a best-fit automation technology and steps necessary to implement an MVP using the technology.

User Interface Input Implementation

This project will involve the implementation of a basic Java GUI to accept parameter specification from our users with validation and potentially advanced range specification. This may be a simple Java Swing interface with inputs/form-controls curated for the particular KBase application we're targeting, which is a Flux Balance Analysis (FBA).

KBase FBA Programming Automation

A Selenium routine must be developed to take the parameters specified through our Java Swing GUI and automation programming the KBase application (Flux Balance Analysis) with those parameters. Programming will involve a variety of challenges including accommodating timing of certain elements (dialogs with fade-outs, for instance) and interaction with a variety of primitive and rich inputs (checkbox vs. a multiple-selection from a set of media).

KBase FBA Execution and Management Automation

Once programmed, we must automate execution and monitoring of a KBase Flux Balance Analysis application. This includes verifying the parameter programming is complete, interacting with the "Run" button, and monitoring the status updates to ensure processing is proceeding successfully and indicating to the collection step once processing completes.

KBase FBA Collection and Review Automation

After processing completes successfully, we will need to collect results from the KBase Flux Balance Analysis by identifying and scraping particular elements from the webpage. These may include the objective value of the simulation and output logs from the job. These values should then be written to the output data structure of the job to be processed or visualized later.

Job Queue and Runner Implementation

The user may enter parameters with many permutations, so to automate execution we will develop a job queue to hold those parameter sets. This queue will be asynchronously linked to the GUI and runners, and will handle delegating jobs to runners and reporting the outcome of a job's processing to the GUI. The runners are responsible for executing the Selenium automation for a given job.

User Interface Results Display

Once a job has completed, we will need to report this back to the user. This may include a basic tabular view, as well as a status view for jobs which have been queued but may not have completed processing yet. This view may indicate aggregate values as well, with a later task.

User Interface Export Functionality

Once a set of parameters has completed execution, the user should have opportunity to export that data for further evaluation in a tool like Excel. A set of jobs results should be exportable to the CSV file format from some GUI interaction, like an export dialog.

Multi-Format Parameter Specification

Parameters may be specified for the application in a variety of formats depending on the parameter's type. Explicit values like "true" or "1" maybe be specified, a set of explicit values may be specified like "1, 4, 10", a range may be specified like "10-100", or randomness may be applied either uniformly or with preference for extremes. THe permutations of these ranges/inputs will need to be processed into a discrete set of jobs and parameters which can then be queued and processed.

Result Aggregation and Identification

Once a set of jobs and parameters have completed processing, their results can be aggregated and outliers may be identifier for the user. These could be visualized through the UI or exported to some other format, like a CSV. We might attempt to programmatically identify outliers in the data and indicate them to the user.

2.2 Risks And Risk Management/Mitigation

TASK	RISK	RISK MITIGATION
KBase Interface Analysis	**0**	
Analysis of KBase interface behavior	**0**	
Automation Technology Analysis	**0.5 - There is a risk that the automation technology that we choose to use becomes deprecated and no longer supported and/or unstable.**	**We mitigate this by choosing a well supported and mature automation framework such as Selenium. We also utilize Maven to manage our dependencies to prevent auto-updating and creating potential breaking changes from updates.**
User Interface Input Implementation	**1 - There are many UI frameworks to choose from. Some may have limitations on the quality of the UI.**	**We chose to use Java Swing, a well-defined and documented library to build our UI and have designed our application in a way that we could create a new UI to collect requirements if necessary.**
KBase FBA Programming Automation	**1 - All software is inherently flawed. Particularly software written by Daniel Way and other members of this team.**	**We will ruthlessly unit and regression test the application to ensure the highest standard of code quality. We also have automated tests that run continuously during code contribution to ensure passing tests.**

KBase FBA Collection and Review Automation	**2 - Given that the project is automating the interaction of a GUI, there are high risks that the GUI (which we do not maintain) will change over time. This can break the automation if elements are not where they should be.**	**We have designed our interactions in a way that a broken element location can be changed with the swap of a single string, describing the location of that element. Reducing the amount of developer work needed to resolve the issue.**

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Take job input from a user in multiple forms such a custom GUI form, or text files such as CSV. These input forms should include various input types, such as boolean, float, or free-form strings. Be able to take two different formats of job input.
- Execute jobs from a user on KBase with multithreaded runners. This allows users to run multiple jobs at the same time. Be able to run at least five runners at one time.
- Collect output from the jobs and report them back to the user in a format of their choosing, such as a text/CSV file. Be able to report output to a user in two different formats.
- Iterate on the project by increasing the amount of KBase applications, inputs, and outputs that can be created by the project.
- Show runner feedback to the user through the GUI or CLI to display progress of jobs. Be able to show runner feedback in the console and through the GUI.

2.4 Project Timeline/Schedule

Our project development will follow an agile format, and the application's implementation will be performed in roughly three phases. The first phase will be "Analysis" and occur during the first-half of the Fall semester. From August through September, our team will perform analysis on the KBase system as well as technologies we may use to automate aspects of the system.

Following analysis, we will implement the basic proof-of-concept functionality for automation KBase through a testing framework, such as Selenium. This basic implementation will complete by the end of November, which will begin an 8 week hiatus before the Spring semester begins.

The Spring semester will include more advanced feature implementation that builds atop the basic functionalities implemented in the Fall semester. These more advanced functionalities include job queuing, multiple runners, multi-format parameter specification, and result aggregation and identification. This more advanced implementation continues through the Spring semester.

Implementation of tasks within these three phases will generally align with iterations in our Agile development process, where we produce a deliverable for our client/advisor and determine an appropriate deliverable for the next iteration's completion. This process allows the client/advisor to redirect our efforts in a continual feedback process, which should enable us to produce a solution which aligns well with the client's use-case. The Gantt chart above details exact dates for task/iteration implementation.

2.5 Project Tracking Procedures

We will use git as our version control by storing our code changes. Git will also be used to lay out issues and teammates can self assign issues according to their strengths. Trello will be used to efficiently map out which tasks need to be completed, are being worked on, and are completed to remove ambiguity. Slack will be used to communicate easily between team members any issues, code snippets, or general help that is needed. We will meet weekly as a team to iron out any issues

throughout the week and give updates on project progress. We also will meet weekly with our client to gain requirements, demonstrate progress, and ask any questions that would be beneficial to the growth of the project.

2.6 Personnel Effort Requirements

This table is a breakdown of the effort requirements for each team member for a standard two week sprint

Task	Number of Hours	Explanation
Research	1-2	Topics may require research before designing a solution
Software/project design	1-2	Solutions should be well thought out before implementation to promote optimal solutions
Software development	5-10	Creating software solutions
Software Testing	1-3	Testing software for bugs and to make sure it performs required task/s
Reviewing team pull requests	1-2	Reviewing team members pull requests to ensure their code meets team standards
Team meetings	1-2	Meetings with clients to demo product and obtain future project requirements and team meetings to assign project requirements and plan sprints

2.7 Other Resource Requirements

There are various outside resources that must be used in the completion of our project. Resources are used to either enable functionality in the project, or aid in the development process of the project.

We have open-sourced our project on GitHub with an MIT license to open the development to anyone upon completion of the project. This also serves as a remote repo and collaboration tool for the team in the development of the project. Developers on the team utilize IntelliJ or Eclipse IDEs

(depending on user preference) to work on the project. The team also utilizes GitHub Actions to automatically build the project and run automated unit tests and check-styles on code formatting. The rest of the resource requirements are open-sourced software, available through Maven to utilize in adding functionality to the application. Dependencies such as Java Swing, and Maven-Checkstyle.

2.8 Financial Requirements

Total financial resources required to complete the project will equal to $0.00 . The reason the project development will be free is KBase is an open source project, the technologies and licenses required to develop are free and or have been provided through being a student of Iowa State. Selenium, IDE's we choose, frameworks, version control, and other technologies are free to use and require no financial resources.

3 Design

3.1 Previous Work And Literature

Include relevant background/literature review for the project

– If similar products exist in the market, describe what has already been done

– If you are following previous work, cite that and discuss the **advantages/shortcomings**

– Note that while you are not expected to "compete" with other existing products / research groups, you should be able to differentiate your project from what is available

Detail any similar products or research done on this topic previously. Please cite your sources and include them in your references. All figures must be captioned and referenced in your text.

3.2 Design Thinking

Detail any design thinking driven design "define" aspects that shape your design. Enumerate some of the other design choices that came up in your design thinking "ideate" phase.

3.3 Proposed Design

Include any/all possible methods of approach to solving the problem:

- Discuss what you have done so far – what have you tried/implemented/tested?
- Some discussion of how this design satisfies the **functional and non-functional requirements** of the project.
- If any **standards** are relevant to your project (e.g. IEEE standards, NIST standards) discuss the applicability of those standards here

- This design description should be in **sufficient detail** that another team of engineers can look through it and implement it.

3.4 Technology Considerations

Highlight the strengths, weakness, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

3.5 Design Analysis

– Did your proposed design from 3.3 work? Why or why not?

– What are your observations, thoughts, and ideas to modify or iterate over the design?

3.6 Development Process

Discuss what development process you are following with a rationale for it – Waterfall, TDD, Agile. Note that this is not necessarily only for software projects. Development processes are applicable for all design projects.

3.7 Design Plan

Describe a design plan with respect to use-cases within the context of requirements, modules in your design (dependency/concurrency of modules through a module diagram, interfaces, architectural overview), module constraints tied to requirements.

4 Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or software.

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study or acceptance testing for functional and non-functional requirements).
2. Define/identify the individual items/units and interfaces to be tested.
3. Define, design, and develop the actual test cases.
4. Determine the anticipated test results for each test case
5. Perform the actual tests.
6. Evaluate the actual test results.
7. Make the necessary changes to the product being tested
8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you have determined.

4.1 Unit Testing

– Discuss any hardware/software units being tested in isolation

4.2 Interface Testing

– Discuss how the composition of two or more units (interfaces) are to be tested. Enumerate all the relevant interfaces in your design.

4.3 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

4.4 Results

– List and explain any and all results obtained so far during the testing phase

- Include failures and successes
- Explain what you learned and how you are planning to change the design iteratively as you progress with your project
- If you are including figures, please include captions and cite it in the text

5 Implementation

Describe any (preliminary) implementation plan for the next semester for your proposed design in 3.3.

6 Closing Material

6.1 Conclusion

Summarize the work you have done so far. Briefly re-iterate your goals. Then, re-iterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

6.2 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE).

6.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar data that does not directly pertain to the problem but helps support it, include it here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc,. PCB testing issues etc., Software bugs etc.