Douglas Swanson

Boeing  8/30/2022

Genesis: Simulation Configuration Management

Contents

[Summary 2](#_Toc113230303)

[Introduction 2](#_Toc113230304)

[Reason 2](#_Toc113230305)

[Objectives 2](#_Toc113230306)

[Literature 3](#_Toc113230307)

[Innovation and Uniqueness 3](#_Toc113230308)

[Scope 3](#_Toc113230309)

[Methodology 3](#_Toc113230310)

[Main Design Consideration 3](#_Toc113230311)

[Secondary and Tertiary Design Considerations 5](#_Toc113230312)

[Deliverables 5](#_Toc113230313)

[Schedule 5](#_Toc113230314)

[Sources 7](#_Toc113230315)

# Summary

The GENESIS application is a proposed solution to the configuration of a Space Simulation currently in use by Boeing’s Virtual Warfare Center (VWC), which is a part of Boeing’s PhantomWorks division. The current problem with the simulation is that it is very difficult to create and configure a new simulation scenario in a reasonable amount of time. This application would allow the team to not only increase the speed in which they can develop a new scenario but also validate its inputs. This validation would give the team more confidence in the simulation when they run it either during an Operator In The Loop (OITL) event or for internal research and development studies. The GENSIS project will take the form of a flexible and high-speed framework that would allow the team to generate a new scenario form any network within the VWC. It would also allow the team’s software engineers to update and modify various modules to increase the capabilities of the software. All of this can be done using web technologies which would allow for containerization and deployment on any kind server.

# Introduction

## Reason

The GENESIS simulation configuration management tool is a proposed solution to the slow turnaround and complex configuration of the Space Sim. The Space Sim is a simulation tool currently in use at the Boeing’s Virtual Warfare Center. This simulation allows the Space Team to run operator in the loop (OITL) training events and provides an environment to simulate and test future space architectures. The space team currently uses a Scenario Master Spreadsheet (SMS), along with a suite of accompanying software, to initialize scenarios and administer changes to objects in the simulation. The SMS is an Excel spreadsheet which currently gets the job done but, when more than one person starts editing a scenario it becomes an exercise in manually tracking changes.

## Objectives

In order to streamline the task of scenario generation, the GENESIS project should keep the following objectives in mind. Should be designed and built with modules in mind, like a modular framework. The inputs are key and should allow for validation. The framework should be well documented to allow for future developers to easily integrate new modules or fix current ones. The framework itself should have unit tests written. This framework should have some hooks written in to display the current scenario state to the user using proprietary tools like stellar. These tools would be rolled into the framework after it is delivered to the customer and would not be available until then. Should allow for the user to create a new scenario or load an existing one. Generate the appropriate schema from a configuration file. Populate the generated schema with data entered by the user. Display input forms so that a user can easily provide the data to be used during a scenario.

## Literature

Currently the literature reviewed was used to get a general basis of what configuration management (CM), Service-Oriented Architecture (SOA), and Services are. Configuration Management is defined, by the Department of Defense as a system that “provides and orderly way to facilitate change...” (3). There are currently a few different software suites that perform software configuration management. One of the most well-known solutions would be Docker, which “takes away repetitive, mundane configuration tasks and is used throughout the development lifecycle for fast, easy and portable application development” (6). Docker’s direct competitor, Kubernetes is another software configuration management tool that handles much of the same activities using the same containerization model (8). On the far end of the software configuration spectrum there is Ansible which advertises itself as a full-service automation platform allowing its users to bring automation to their containers, security, applications, infrastructure, and cloud (1). On the opposite side, there is a tool called Config Cat which allows for a quick method of toggling features on individual applications (4).

A Service is “something useful a provider does for a consumer” (7) which can be anything ranging from a product on a shelf or something less tangible such as insurance or providing access to the internet.

## Innovation and Uniqueness

GENESIS isn’t necessarily innovative, but it is a more focused configuration management tool. Most of the tools on the market seem to be aimed at any solution on any platform anywhere. This project is specifically designed to handle the setup and configuration of the Space Simulation for the VWC and that is it. It needs to be specific in order to provide it’s highly specialized users the flexibility and robustness they desire. It also needs to be flexible enough to change at the same speed as the simulation itself. So this tool is a super specialized configuration management tool designed to run in the Virtual Warfare Center’s private networks, sometimes without access to the greater internet.

## Scope

The scope of this GENESIS project for this applied project course is as follows. The project should define the basic framework for the project. This framework should allow a user to generate a new scenario or load an existing scenario. Should allow the user to generate basic spacecraft data for this particular scenario through a webpage interface. Should generate the database schema and populate them with the provided data.

# Methodology

## Main Design Consideration

The modules (services) of the GENESIS project would handle individual business processes. Processes like designing a spacecraft or configuring the flight software commands that an operator can send during a simulation run. Ideally the services would be designed with the REST principals in mind. Also, the new services that would be written in a language that could be run in a Linux or Windows environment. To accomplish this, the services will be written in Node Typescript. This should allow the services to be easily containerized. The current services, I.E. Access Generation or Schedule Generation would need to be modified to run as headless and RESTful services versus their current UI implementations.

Kubernetes would be the first choice for the Service Broker because it would await requests from the users and then spin up the services required to accomplish the current requests. This will also allow for the modules to be ran on any machine within the VWC network without much configuration work on the side of the Software Engineering team.

The user facing UI applications would be written as REACT applications. The REACT library provides the ability to create generic components and basically add them to any webpage in the future. This would allow the team to spend time on the front end defining these components for a desired look and feel. This look and feel would than carry through all the applications designed in the future. The downside to this is the amount of work the team must do upfront but would allow for the team to create their own standard. REACT library also as the ability to automatically apply any bug fixes in real time without having to refresh a web page. It is a nice feature that would grant the ability to change the user pages during an event in the case of application stopping bugs. One last benefit of using REACT is that there are already standards in which the team can modify and build off instead of starting from scratch. These REACT applications would be run out of a central Node server; this server would be the user’s connection into the GENESIS environment.

Next up is the unit tests. Since both the UI and services are using JavaScript the JEST testing framework would be used to both write and execute the tests. JEST was chosen because, like REACT, it has a large community that has been using it for quite some time and thus it can be accredited by the community. It will also allow test to be written for both the Node services and the REACT components.

GENESIS will use MySQL and Redis as its databases. The MySQL should store the starting state of each scenario defined through GENESIS. Redis will be used to load each scenario state into its NoSQL key value storage for use during the simulation run. It could also handle logging and state change for the GENESIS application, but this is to be determined after further research into how REACT applications work. GENSIS may need to interpret the starting state stored in MySQL and translate it into the proper key value stores in Redis for a simulation run.

To provide the flexibility required by the VWC customer, the GENESIS tool itself needs to be configurable in order to know what services and UI pages a specific scenario requires. I believe this can be done through configuration flat files that the developers of the tool would change as the tool itself updates. These configurations would change what services Kubernetes spins up when a specific scenario is selected. Also, it would change which components that REACT will display to the end user.

All of this should be able to be written through the use of Microsoft’s (semi) open-sourced IDE Vs Code. It was specifically designed to write JavaScript for the web environment and has a handful of extensions that would work with Kubernetes and JEST, the testing software.

## Secondary and Tertiary Design Considerations

GENESIS is a framework made of many different parts all using different design methods. So here are some backup design options for a few of those parts. The largest unknown at this point is using Kubernetes as the service broker for the service modules. It is an unknown because it is a completely new concept for me as a developer and would require some playing around to figure out if it will work within this framework. So, a few backup options would be using a different service, like docker or ansible, or going through the process of writing the broker myself, which would be very interesting to say the least. If it came down to writing it myself, it would be written in Node JavaScript.

The service backup, should writing them as a RESTful node service, would be to write them in a language I understand better, c#. These services would look a lot like the services written in Dr. Chen’s service-oriented computing textbook (2). If I fail to get these working properly than

# Deliverables

* The GENESIS framework working with at least a single module.
  + The module would allow a user to configure satellites within a scenario.
* The test cases used to validate that the GENESIS framework is working as intended.
* The user interface for any module delivered with the Framework.
* Documentation of the GENESIS framework and any modules delivered.
* All data properly stored within the MySQL database. Schema generated properly automatically.
* A method of creating a new scenario and loading an existing one.

# Schedule

The schedule for this project has been broken down into the following sprints:

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint # | Dates | Reports | Project Goals |
| 1 | Sept 5 – 11 |  | Data Modelling  Scenario start / load Service |
| 2 | Sept 12 – 18 | Bi-weekly 1 | Spacecraft Service |
| 3 | Sept 19 – 25 |  | Start Service Broker |
| 4 | Sept 26 – Oct 2 | Bi-weekly 2 | Complete Service Broker |
| 5 | Oct 3 – 9 |  | REACT UI App (Start and Load) |
| 6 | Oct 10 – 16 | Mid-Term | REACT UI App (Spacecraft Service) |
| 7 | Oct 17 – 23 |  | Complete REACT UI Apps |
| 8 | Oct 24 – 30 | Bi-weekly 3 | Start Configuration of GENESIS |
| 9 | Oct 31 – Nov 6 |  | Complete Configuration of GENESIS |
| 10 | Nov 7 – 13 | Bi-weekly 4 | Final cleanup |
| 11 | Nov 14 – 20 |  | Documentation |
| 12 | Nov 21 – 27 | Final Report |  |
| 13 | Nov 28 - 4 | Final Demo |  |

First milestone is to model the spacecraft data as it currently stands in the space simulation. This is to understand what data the service is required to receive from the user. Also, need to write a service that handles initialization and loading of scenarios when a user selects them.

Second milestone is to write the spacecraft service which will accept inputs from the user, validate them, store them in the database, and respond to the user.

The third milestone is to configure and run the service broker with the two services created in the past two milestones. This is a two week milestone to allow for plenty of time should the service broker not work out as intended.

Fourth is to start writing the REACT UI applications for both the starting and load of scenarios as well as the Spacecraft Service Data form. These will be the user’s insight into the GENESIS application and will collect data and trigger service calls based on the user’s selections. This is a three week milestone which should cover learning REACT as well as putting the spit and polish on the interfaces.

Fifth milestone is to bring all the pieces of the GENESIS application together. As part of this a method of configuration will be devised to allow for different types of scenarios and the services they require.

The sixth milestone is the final cleanup and documentation of the application as a whole. This is done to bring the project together and get it ready for presentation and delivery.

The last milestone will be generating the reports and presentation for this course.

# Sources

1. Ansible, Red Hat. “Ansible Is Simple It Automation.” Ansible Is Simple IT Automation, Red Hat, <https://www.ansible.com/>.
2. CHEN, YINONG. Service-Oriented Computing and System Integration: Software Iot Big Data and AI as Services. KENDALL HUNT, 2017.
3. “CM Overview.” Chief Information Officer U.S. Department of Defense, United States Department of Defense, <https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_cm_overview/>.
4. ConfigCat. “Configcat Feature Flag Service.” ConfigCat, ConfigCat, 1 Mar. 2022, <https://configcat.com/#product>.
5. Dikmans, Lonneke, and Ronald Van Luttikhuizen. *SOA Made Simple*. Packt Publishing, 2012.
6. “Home.” Docker, 24 Aug. 2022, <https://www.docker.com/>.
7. Jiang, Peishi, et al. “A Service-Oriented Architecture for Coupling Web Service Models Using the Basic Model Interface (BMI).” Environmental Modelling and Software, vol. 92, June 2017, pp. 107–118., <https://doi.org/https://doi.org/10.1016/j.envsoft.2017.01.021>.
8. “Production-Grade Container Orchestration.” Kubernetes, The Linux Foundation, <https://kubernetes.io/>.
9. Sen, Kaushik. “Top 10 Configuration Management Tools You Need to Know about: Upguard.” UpGuard, UpGuard, Inc, 12 May 2022, <https://www.upguard.com/blog/configuration-management-tools#toc-2>.
10. Whyte, Jennifer, et al. “Managing Change in the Delivery of Complex Projects: Configuration Management, Asset Information and ‘Big Data.’” International Journal of Project Management, vol. 34, no. 2, Feb. 2016, pp. 339–351., <https://doi.org/https://doi.org/10.1016/j.ijproman.2015.02.006>.