# DNA to Feature Models

## FINAL REPORT

**Team Number: sddec20-22**

**Client: Professor Myra Cohen**

**Advisor: Professor Myra Cohen**

**Team:**

**Abdul Rahman Moughrabi: Developer/Documentation Management**

**Ahmad Nazar: Team Leader/Developer**

**Ahmed Alketbi: Developer/Debugger**

**Hyegeun Gug: Developer/Web Management**

**Prathik Nair: Debugger/Developer**

**Team Email: sddec20-22@iastate.edu**

**Team Website: http://sddec20-22.sd.ece.iastate.edu/**

# Table of Contents

# 1 Introduction

## 1.1 Acknowledgement

Special thanks to Dr. Myra Cohen (Iowa State University), and Mikaela Cashman (Iowa State University) for providing the technical knowledge and guidance needed for success in this project. Special thanks also to the course supervisors, and everyone providing mentorship during the course of the project.

## 1.2 Problem and Project Statement

Software Product Lines are a set of software systems with the intrinsic value of features pertaining to the satisfaction of certain needs; a key aspect being a model presenting commonality and variability within a hierarchical model. A set of these SPLs are called families of SPLs. A subset of SPLs are Feature Models. Feature modeling is an organization tool that allows an engineer to represent features in a tree of hierarchies; a tool for software modeling to present family of software models. It is a unique and efficient way of modeling feature rich systems.
BioBricks, an iGEM repository of biological parts, provides a tool for biologists and users interested in DNA related-fields to analyze parts and models created on this website.. While this tool is useful, the repository does not implement the feature model organization method; revealing new ideologies about these DNA models that one could not see before.
Over the course of a year, creating an Eclipse plugin that creates Feature Models based on existing models found in an open-source repository called BioBricks is the goal of the project. A successful implementation of this plugin allows biologists and scientists to view various models from BioBricks in an organized hierarchy.

## 1.3 Operational Environment

The main operating environment for the plugin is windows, Mac, and linux. Eclipse is used to run the plugin and the database is run on SQL. Any OS that supports SQL and Eclipse will be able to run the plugin with full functionality.

## 1.4 Intended Users and Uses

The main users are scientists that build biological models of living organisms with specific desired properties. The goal of this project aims to be an aid for everyone interested in building DNA Feature Models without any restriction.

## 1.5 Assumptions and Limitations

During the course of the project, some assumptions and limitations needed to be noted. The assumptions and limitations are as follows:

Assumptions:

- Users with and without knowledge of feature models can build feature models of DNA.
- The end product provides access and can be used anywhere with internet access to the Biobrick repository.

Limitations:

- The Biobricks Repository is the main source of information and users need internet access anytime they want to use the plugin.
    - This limitation, however, is an introductory limitation; users running the plugin for the first time will need to update the local database with parts from the online database.
    - The intermediary steps after need not require internet access.

## 1.6 Expected End Product and Deliverables

The expected end product is a FeatureIDE plugin that uses parts extracted from the BioBricks Repository. The plugin includes up-to-date BioBrick parts classified within organized categories with informative description for each part. The organization allows users to construct models without the hassle of navigating BioBricks repository.
Delivery Date : November 20th 2020

# 2. Specifications and Analysis

## 2.1 Final Design

The final design of the project involves a hierarchical approach in the FrontEnd aspect such that each dropdown menu contains a child/subset of the selected option with similar attributes as the parent but contain distinct features. The final selection made by the user invokes the appearance of a pop-up window containing necessary information for the user to assess which part(s) are to be included in the Feature Model to be created.
The BackEnd aspect of the project was designed in a similar way to the FrontEnd but an abstract class was used to create the barebones of all parts. Classes that extend the aforementioned abstract class implement specifics that distinguish each subset and class of parts. An example of the following is Plasmids extending Part such that it has a name, description and length, and Plasmids include the attribute explaining a PCR insertion.

## 2.2 Design Analysis

The decisions made to design the FrontEnd and BackEnd aspects of the project relied on the stylings used in AGILE Software Development Practices. Creating an abstract class in the BackEnd

allowed for redundant code to be neglected where only one class is a parent class of all classes. Using SQL made the most sense since it is one of the most convenient methods in database creation available in the modern world. Everything in the backend is dependent upon controllers using CURL commands by ways of the FrontEnd.
For the FrontEnd, AGILE Software Practices were also used to develop and conclude design decisions. The FrontEnd relies on the aforementioned CURL commands to update data currently available related to Parts. The idea behind CURL commands and utilizing GET requests in the BackEnd controllers is to have consistent updates to the data and the constant availability of most recent data, specifically parts and models appearing on the BioBricks Repository.
These design choices were deemed to be most feasible based on the development style and the inherent functionality of the plugin.

# 3 Testing and Implementation

## 3.1 Interface Specifications

The project is entirely software development and coding. There are multiple ways in which a software project can be tested. This project was tested using:

- Functionality testing
    - Created test cases that asses all function implemented and ensures correct functionality
- Mockito Tests
    - Used to test part information parsing and database behavior
- Review of full code
    - Presented code to a professional for review and criticism
- CI/CD
    - Keeps the server running automatically and detects compiling issues on recent pushes to the Git
    - Immediately push latest updates to code.
- Code analysis
    - Testing the code with a software called PMD Java that reveals security vulnerabilities and concurrency issues
- JUnit Tests
    - Verify the wanted construction of feature models

# 3.2 Hardware and Software

## 3.2.1 Software To Be Used

The main software tools used for the plugin are Springboot Suite, Maven, Java 12+, Eclipse IDE and FeatureIDE.

Software used in the testing phase include:

- **PMD:** used to scan code written and shows some potential bugs and problems that may occur. This is useful because it can increase performance, complexity of code, and assists debuggers in removing all bugs.
- **Checkstyle:** used to improve code adheres and makes sure the style of each class written is similar to the other. This is useful because code is pushed to the database by 5 different people and each one will have a different coding style.
- **SecurityBugs:** used to show bugs that MAY occur in the future and helps write code to be able to avoid such bugs. This is useful because it helps save time by avoiding the occurrence of different bugs.

# 3.3 Functional Testing

As each aspect of the project was completed, we needed to test the functionality of the project thoroughly to ensure a satisfactory final product. When each individual module was completed, we performed unit testing on that module to verify the functionality.

# 3.4 Unit Testing

The team tests each method introduced to the server for correctness and performance. As more functionality is added to the backend, complex unit tests to validate stability and behavior will hence be added. These tests will be conducted on web scraping, data parsing, and database to validate our backend development.

## 3.4.1 Plugin

To individually test the plugin we emulated the output of the program. Then we send the test status to the script to check if the plugin is working correctly. In order to ensure that the data was correct, we cross-checked the result and compared it using Postman.

## 3.4.2 Plugin to Database

To ensure that the plugin could access the database, we had to check if the full data could be accessed from the plugin.

## 3.5 Integration Testing

Integration testing consists of putting the Plugin interface and the Plugin into the database. In order to ensure that the system performs as required, we need to make sure that we can send data from the program, see if the database was created  Once that is done, we can make sure that the program correctly catches the data.

## 3.6 Interface Testing

GUI is an important component of any software. The team's goal is to build a simple interface that lets users choose parts using a drop-down menu. Testing is done through simulation and on-click tests then users will be asked for feedback on implementation design to ensure a familiar and easy GUI design. Most of the interface testing follows manual testing since it provides better debugging results.

## 3.7 Database Testing

We performed a test ensuring if the database is initialized when the files are parsed through the backend. In order to test, we used a postman to check if the file could be parsed correctly before working with the files. After checking the outputs, we parsed the files and made sure the database has been initialized correctly.

## 3.8 System Testing

System testing will basically be a combination of the two integration tests with the addition of the server files. For a successful test, we need the same exact results as the integration tests.

## 3.9 Non-Functional Testing

Tests were conducted to establish stability, upgradability, usability, security, and performance. Mockito testing is used as database capacity increases to ensure the desired efficiency for the backend. The team strained the server to ensure its ability to handle many users at the same time. As for the frontend, on-click testing is used for various scenarios to ensure a smooth and satisfactory user experience.

## 3.10 Process

The team started off by a bottom-up approach to PMD and SecurityBugs. This involves adding code through PMD and SecurityBugs, which analyze code and the project for potential bugs or problems that can occur. For the testing process, team members went through the analyzed code and made changes to the component such as functions, class and interface. Due to working in a team project, code needs to be in a consistent style. By using Checkstyle, the team's code ended up with a similar style of code pushed to the repository and easier to understand the portion of what other members worked on.

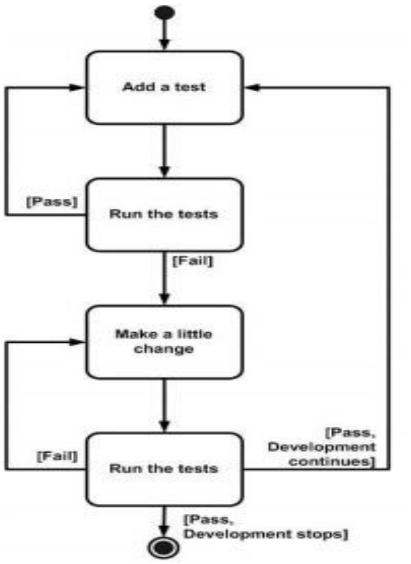



Figure 1: Diagram showing Basic testing flow. This diagram shows how the team's coding and testing has been processed involving PMD, SecurityBugs, and Checkstyle.

## 3.11 Results

The team faced several issues while working on the project. The first issue the team faced was "Error with web scraping." During the phase of web scraping, the team's final goal is to have consistent data from the Biobrick repository. However, the team struggled due to having unique characters throughout the output file from web scraping. This was resolved by having an assumption of having Operation System's base language setup other than English might cause the problem. By working on English based OS, the web scraping function got resolved and fully functional.

Teams are building file parsers to send output files to the database. The backend also has an automatic XML parser that gets the data from BioBricks' XML database. The XML parser was difficult to generalize between all parts categories. As a result, some part types had different implementations for the XML parser.

## 3.12 Implementation Challenges and Issues

The team ran though different challenges throughout the project such as:

1. Understanding the background of the project since the project deals with DNA and is more related to biologists.
2. XML parsing had a lot of edge cases since some part types had more than one format in the XML file structure.
3. FeatureIDE source code has minimal documentation which made it hard to modify or use.
4. COVID19 made team meetings more challenging and team members who work on the same stuff had some troubles with implementation and testing.

# 4 Operational manual

## 4.1 Necessary Items / First-time setup

### 4.1.1 Necessary Items:

- Eclipse compatible OS
- Java Environment
- Jar file(database)
- SQL to run the database
- FeatureIDE Plugin

### 4.1.2 First time setup:

Downloading the correct version of Java environment and eclipse is necessary to run the plugin correctly.
https://download.eclipse.org/eclipse/downloads/drops4/R-4.17-202009021800/
Next comes downloading the plugin from GitHub and installing it in eclipse

## 4.2 Setup

### 4.2.1 Setting up the database

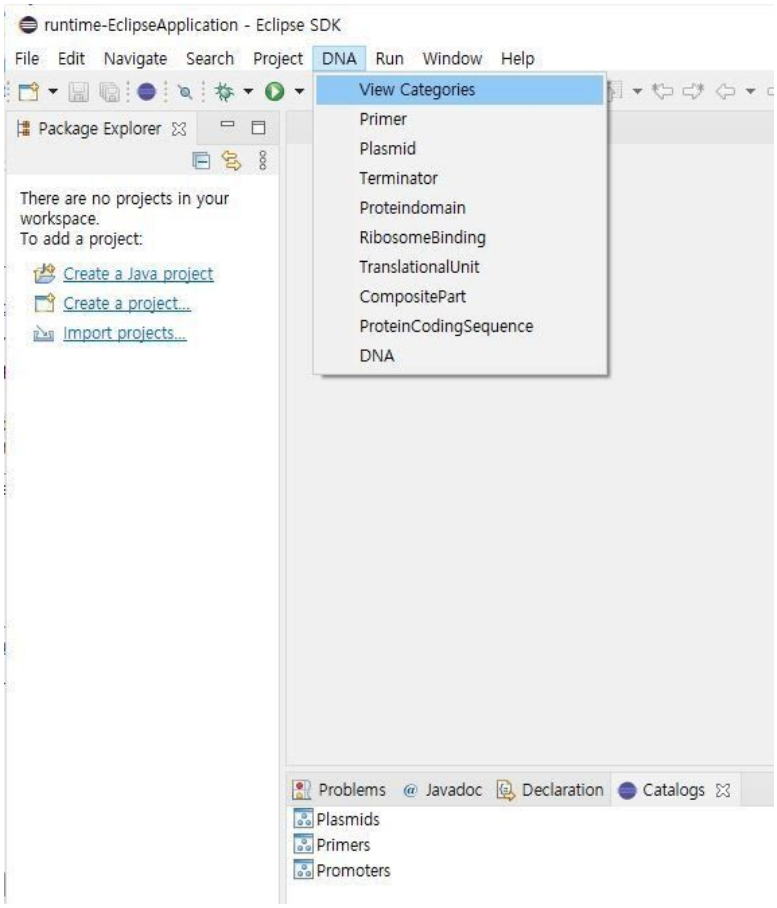
Database can be set up by:

- Opening and running the included JAR file

## 4.2.2 Setting up the plugin

Setting up the program comes in 2 steps:

- Downloading and installing DNAtoFeatureModels Plugin
- Running the plugin in eclipse to display the different categories




## 4.3 Running

Once the plugin and database have been set up, the operation should be simple. The user will select the category and sub-category needed, selecting the action import, view, or edit.

# 5 Conclusion

The plugin project incorporated ideas from an ongoing research project led by Dr. Myra Cohen and involved the application of concepts and knowledge gained by going through fundamental courses at Iowa State University. The plugin incorporated the use and application of critical analysis skills and the ability to think through multiple design options and choose the best route that benefits the project the most in terms of optimization and functionality. The plugin project has the barebones and foundations completed with utmost detail and dedication. The FrontEnd requires some work in terms of formulation and refinement of output to propagate data in order to automatically create parts from online data regarding parts of the database. The project was a fulfilling experience to be a part of during the team's last two semesters at Iowa State University and despite the unforeseen event of COVID-19 as well as the quick need to adapt to a virtual situation, the team found a way to persevere and deliver an efficient final product.

# 6 References

Previous works are referenced below. The team thanks all the information contributed by these sources and their availability.


- D. N. KTH, D. Nešić, J. Krüger, Ș. Stănciulescu, Ș. Stănciulescu, T. Berger, T. Berger, Kth, Jacob Krüger University of Magdeburg, University of Magdeburg, Abb, Abb, Chalmers University of Technology, Chalmers University of Technology, University of Tartu, Saarland University, and Imperial College, “Principles of feature modeling,” Principles of feature modeling | Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 01-Aug-2019. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3338906.3338974. [Accessed: 23-Jan-2020].
- M. Cashman, M. B. Cohen, M. B. Cohen, W. Niu, Mikaela Cashman Iowa State University, Iowa State University, Iowa State UniversityView Profile, Justin Firestone University of Nebraska-Lincoln, Justin Firestone, University of Nebraska-Lincoln, University of Nebraska-LincolnView Profile, Iowa State University, Suranaree University of Technology, Suranaree University of Technology, Wei Niu University of Nebraska-Lincoln, Chalmers | University of Gothenburg, University Lille, Danfoss Power Electronics A/S, University of Namur,Humboldt-Universit, University Paris, IK4-IKERLAN Research Center, Sorbonne University, and TU Braunschweig, “DNA as Features: Organic Software Product Lines,” DNA as Features | Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A, 01-Sep-2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3336294.3336298. [Accessed: 23-Jan-2020].
- T. Thum and J. Meinicke, “FeatureIDE,” FeatureIDE. [Online]. Available: https://featureide.github.io

# 7 Appendix I

- *Principles of Feature Modeling*- Damir Nesic, Jacob Kruger, Stefan Stanciulescu, Thorsten Berger
- *DNA as Features: Organic Software Product Lines*- Mikaela Cashman, Justin Firestone, Myra B.Cohen, Thammasak Thianniwet, Wei Niu
- FeatureIDE source code

**JSON EXAMPLE Object**

```
{

   "partFunction": "generic function",

   "partType": "Plasmid",

   "partName": "BB_123",

   "pcr": "ccaaggg"

}
```

Software Bugs encountered during the creation of the project include:

- **Exception handling**
    - Dealing with FileNotFound exceptions.
    - Handling NullPointer Exceptions
- **Error handling**
    - Using debuggers to find sources of bugs.
    - Using break statements.
- **Local database persistency**
- **Database setup without all required servers running such as Apache and MySQL server**

## 7.1 Appendix Ⅱ - Design Document

# DNA to Feature Models

DESIGN DOCUMENT

Senior Design Team 22
Professor Myra Cohen
Abdul Rahman Moughrabi: Developer/Documentation Management
Ahmad Nazar: Team Leader/Developer
Ahmed Alketbi: Developer/Debugger
Hyegeun Gug: Developer/Web Management
Prathik Nair: Debugger/Developer

Team Email
http://sddec20-22.sd.ece.iastate.edu/

Revised: 04.26.2020/Version3

# Executive Summary

## Development Standards & Practices Used

Following a set of standards ensures development of  a product that is safe and adheres to the consumer preferences and expectations; while also ensuring a reliable, and organized workflow for the engineers and the consumer. The standards used in this engineering standards used in this project follow the guidelines of:

- IEEE Engineering Standards
- IEEE Software Engineering Standards

## Summary of Requirements

- BioBricks repository
- Extending plugin to support bio bricks
- Web crawling
- Software product line engineering
- Translation of features to be compatible with Feature IDE
- Creating a system architecture

## Applicable Courses from Iowa State University Curriculum

- Com S 228: Introduction to Data Structures
- Com S 309: Software Development Practices
- Com S 311: Design and Analysis of Algorithms
- CPR E 308: Introduction to Operating Systems
- E E 230: Electronic Circuits and Systems

## New Skills/Knowledge acquired that was not taught in courses

- Background on BioBricks parts that are used in biological living cell building.
- Feature Modeling Concept and application
- FeatureIDE Eclipse Plugin
- Effective Team Coordination
- Effective Client Communication

# Table of Contents

# List of figures/tables/symbols/definitions

## Definitions

- **BioBricks Parts:** a standard for interchangeable parts, developed with a view to building biological systems in living cells. BioBricks Parts are referred to as Parts within the design document.
- **Software Product Line (SPL):** Software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production. This definition is referenced as **SPL** through the document.
- **Feature Model:** a compact representation of all the products of the SPL in terms of features.
- **FeatureIDE:** an Eclipse-based IDE that supports all phases of feature-oriented software development for the development of SPLs: domain analysis, domain design, domain implementation, requirements analysis, software generation, and quality assurance. Different SPL implementation techniques are integrated such as feature-oriented programming (FOP), aspect-oriented programming (AOP), preprocessors, and plug-ins.

## Figures



## Tables

# 1 Introduction

## 1.1 Acknowledgement

Special thanks to Dr. Myra Cohen (Iowa State University), and Mikaela Cashman (Iowa State University) for providing the technical knowledge and guidance needed for success in this project. Special thanks also to the course supervisors, and everyone providing mentorship during the course of the project.

## 1.2 Problem and Project Statement

Software Product Lines are a set of software systems with the intrinsic value of features pertaining to the satisfaction of certain needs; a key aspect being a model presenting commonality and variability within a hierarchical model. A set of these SPLs are called families of SPLs. A subset of SPLs are Feature Models. Feature modeling is an organization tool that allows an engineer to represent features in a tree of hierarchies; a tool for software modeling to present family of software models. It is a unique and efficient way of modeling feature rich systems.

BioBricks, an iGEM repository of biological parts, provides a tool for biologists and users interested in DNA related-fields to analyze parts and models created on this website.. While this tool is useful, the repository does not implement the feature model organization method; revealing new ideologies about these DNA models that one could not see before.

Over the course of a year, creating an Eclipse plugin that creates Feature Models based on existing models found in an open-source repository called BioBricks is the goal of the project. A successful implementation of this plugin allows biologists and scientists to view various models from BioBricks in an organized hierarchy.

## 1.3 Operational Environment

The project is software-based. Java 8 and the FeatureIDE plugin for Eclipse are used for the project.

## 1.4 Requirements

The project is broken into functional and non-functional requirements.

- **Functional Requirements:**
    - Extract BioBricks part data using web scraper and XML extraction.
    - Automatically parse scraped part info into objects and populate the database.
    - Construct BioBricks Feature Models through FeatureIDE.
    - Create Software Product Lines from models using a simple GUI.
- **Non-functional Requirements:**
    - Ensure part database' capacity, security and accessibility to establish easy upgradability and data fetching.
    - Efficient and fast response time for web scraping, XML extraction and Feature Model construction.
    - Ability to handle many clients accessing the server without hindering performance.
- **Constraints**

- Ability to handle many clients accessing the server without compromising integrity and preventing a Denial of Service attack with overloaded traffic
- Working hand-in-hand with the current version of FeatureIDE and the need to update the plugin with consecutive FeatureIDE updates

### 1.5 Intended Users and Uses

The main users are scientists that build biological models of living organisms with specific desired properties. The goal of this project aims to be an aid for everyone interested in building DNA Feature Models without any restriction.

### 1.6 Assumptions and Limitations

During the course of the project, some assumptions and limitations needed to be noted. These assumptions and limitations are as follows:

**Assumptions:**

- Users with and without knowledge of feature models can build feature models of DNA.
- The end product provides access and can be used anywhere with internet access to the Biobrick repository.

**Limitations:**

- The Biobricks Repository is the main source of information and users need internet access anytime they want to use the plugin.
  - This limitation, however, is an introductory limitation; users running the plugin for the first time will need to update the local database with parts from the online database.
  - The intermediary steps after need not require internet access.

### 1.7 Expected End Product and Deliverables

An expected end product is a FeatureIDE plugin that uses parts extracted from the BioBricks Repository. The plugin includes up-to-date BioBrick parts classified within organized categories with informative description for each part. The organization allows users to construct models without the hassle of navigating BioBricks repository.

**Estimated Delivery Date:** December 1st 2020

## 2. Specifications and Analysis

### 2.1 Proposed Approach

The project can be tackled using various techniques and methods to solve the problem and deliver a high-quality product.  One approach is dividing the project into two sections: theoretical and practical section. For each section, assign two subsections: architecture and scope. Strengthening the understanding in the fundamental steps enables a solid composition of the scope of the theory. Gaining conceptual insight and obtaining all architectural designs helps ease the design of application and the practical section.

Another approach deemed vital and best is approaching this project as a project manager working on a software application for a company. This project consists entirely of coding and software design; this method proves unparalleled. Devising such a mechanism aids in the production of an ideal product. Utilizing this method commences several documents to aid in beginning the project: a business case, statement of requirements, a project timeline, risk assessment and mitigation, budget, and lastly, a communication plan. The last method of approach is an agile approach. This approach entails promptly coordinated, vigorous, and nimble adaptations to varying settings.

These methods of approach all follow IEEE standards with designing a software project and the standards regarding joint project work. Research and analysis of several papers concerning compilation of an architectural blueprint of the project and beginning development was completed as segways into the project. Various research papers handed out to us from our client: DNA as Features: Organic Software Product Lines and Principles of Feature Modeling were also studied. To start on the development of the plugin, a solid comprehension software product line engineering, the BioBricks repository, and Feature IDE (an Eclipse plugin) needed to be built. The first few weeks began with grasping the core concept of the project by identifying and exploring the various aspects of implementation (mentioned above). In the following weeks, project schemes and strategies were devised. Multiple tools make their use in the project. Those tools are web crawling, Java/XML programming, and working with Eclipse plugins.

Members of the team are tasked with roles best attributed to their ability and allows them to explore and learn while completing the tasks. Taking all these into consideration, the project presents milestones within a deadline to be achieved by following an Agile development scheme. Functionalities, hence, increase with project and team progression as judged by the team.

## 2.2 Design Analysis

As mentioned in the previous section, discussions on different tools necessary to begin the project and exactly how to use them commenced. Most tools and experiments worked well; most experiments lead to successes except for a single failure. Web crawling was a complicated task, but a program that scraped a simple, random website was created. Another success was understanding and editing the source code for an Eclipse plugin. After successfully scraping the data from the website and reading it, thoughts on an XML conversion of the data for later use came about.

There were some challenges translating it to the correct XML format. Throughout the testing and experimenting session, observations were made on modifications and tools needed to take advantage of during the course of the project. One observation was that data scraping does not result in XML code, therefore resorting to an SQL server deemed the next best option. Some recorded thoughts were learning XML aids during the product's final stages, changing from web scraping to an SQL database, and understanding how additions to plugins are made. more work spent on the approach of software design and the use of the software is required for an efficient gateway towards the end of the project.

## 2.3 Development Process

DNA to Feature Models follows the approach of Agile software development. Based on the nature of the project, Agile is most suitable due to project requirements and features evolving throughout the process of creation. The project has preliminary, required foundations but the building blocks and the materials built upon the foundations dynamically change with the project.

The team separates tasks based on individual skill; applying the best expertise to a given task. Team members knowledgeable in the backend aspect of the project work in that scope and those knowledgeable in the frontend gain the same workload within that location.

## 2.4 Conceptual Sketch

The conceptual sketch of the project is shown in Figure 1. The project involves utilizing Software Product Lines and Feature Models. To present Feature Models that make sense to a given user, a friendly user-interface is required. The user interface is provided through an Integrated Development Environment called FeatureIDE. This section is presented through the frontend aspect of the plugin. The frontend includes all formable relationships as defined by a feature model, and is built-upon Eclipse.

The next section talks about the backend aspect of the project. Parts from the BioBrick Repository will be extracted using a web-scraper and stored in a designated database. This database includes all information relevant parts used in a DNA model. Using the database, creation of models depends on an XML parser which organizes elements of a model according to a user and utilizes all properties of a subset of features with respect to a superset of features.

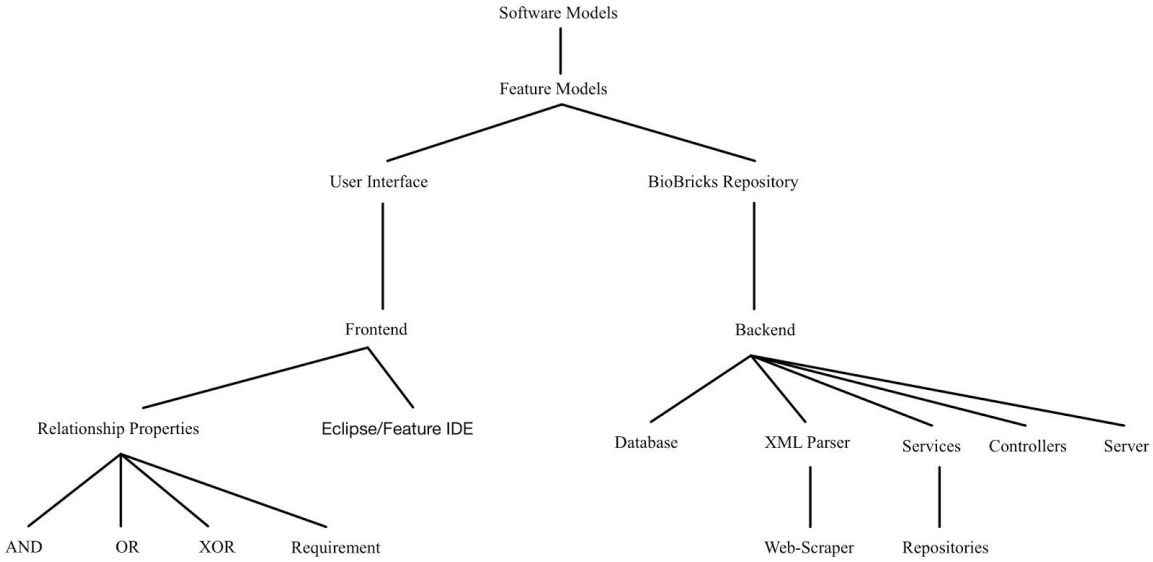



**Figure 1:** The flow of project requirements and dependencies presented in a hierarchy. The figure is modelled similarly to a feature model.

# 3. Statement of Work

## 3.1 Previous Work And Literature

- *Principles of Feature Modeling*- Damir Nesic, Jacob Kruger, Stefan Stanciulescu, Thorsten Berger
- *DNA as Features: Organic Software Product Lines*- Mikaela Cashman, Justin Firestone, Myra B.Cohen, Thammasak Thianniwet, Wei Niu
- FeatureIDE source code

The above reference documents provide various pieces of information that need to be brought together for the project to function fully. *Principles of Feature Modeling* dives deep into the concept of feature modeling, how it is used in the real world, and it's overall functionality. *The DNA as Features* document provides us the technical insight (from a biological viewpoint), and how biobricks and feature modeling come into play. With these two documents and the given FeatureIDE source code; all three pieces combined gives the project all its supplemental references. While there are many feature modeling products in the industry, none bring together the three documents outlined above. All references are cited below in section 6.2.

## 3.2 Technology Considerations

The project does not rely on technology that is behind it's time. In fact, anything that can be conceived (in terms of this project), can most likely be programmed in. While the technology needed for this project to be successful is available, things like efficiency, and data storage come to mind when improvements come to mind. The project relies on an individual server that stores all BioBricks data. Data is hard coded into the plugin.

A design tradeoff was made between using a live web scraper vs one updated on every interval. The decision was made to go with an interval-based update due to the fact that the BioBricks Repository is updated once a year towards the end of the year after the completion of a so-called competition: the iGEM Competition. This period implies that the update occurs after the completion of the competition and automatically rolls out as a CI/CD functionality.

## 3.3 Task Decomposition

The following tasks are derived from the project's requirements:

- Obtain parts' data from BioBricks Repo using web scraper/XML extraction.
- Construct parts objects corresponding to their types with the obtained data.
- Populate the database with parts data.
- Establish a connection between the server and FeatureIDE plugin.
- Construct feature modules within FeatureIDE using the XML Feature Builder Tool.
- Create a simple drop-down menu to assist users to choose parts inside a feature module.

## 3.4 Possible Risks And Risk Management

With every project comes a risk. The pandemic crisis has split the team across the world, forcing remote interaction. Face to face interactions often led to a greater understanding of what needed to be done on the project. The software aspect of this project makes it easy to collaborate on remotely; there are many communication channels set up for us to reduce the risk of miscommunication. Risks associated with the project are the following:

- Communication Issues
    - Giving each team member tasks to be completed by a deadline and a friendly environment where communication is encouraged is provided. Team leader also checks in periodically.
- Discontinuation of the BioBricks and iGEM Repository
    - Find a new source to update parts, and keep available parts within the database.
- Plugin Developmental Error
    - Break up plugin development into parts.
- Loss of Updated Code
    - Use of GitHub and creating branches to keep most updated work on remote and most successful on the main branch.

### 3.5 Project Proposed Milestones and Evaluation Criteria

- **Establish connection to Biobrick Repository to gather Biological Part information**
    - In order to get data for Parts to build Feature Models, the team needed to make sure all Biological Part data on Biobricks repository is scraped.
- **Store Biological Part information to database**
    - Build a database with the output from web scraping.
- **Build Feature Models based on database**
    - On initialization of the database, the team integrates a mean to convert data to part objects for the Feature Model in FeatureIDE.
- **Update database every year when changes are made**
    - Due to iGEM opening annually, parts in the Biobrick Repository will be updated annually. The database is hence updated annually.
- **Improve plugin functionality**
    - The final format of the project is to build a system with plugins usable from Eclipse.
- **Enhance user interface**
    - The team updated the graphical user interface to provide easier but effective tools for users.

In order to confirm that team's milestones are fulfilled, the team came up with following ways to evaluate:

- Confirm the correct data has been scraped with the online BioBricks repository.
- Run and check if output of web scraping is parsed to the team's database with all contents.
- Extract database and run it through FeatureIDE to ensure the parts are usable and stored.
- Check annually for updates to the database from Biobrick Repository to ensure the team's database is updated.
- Make sure plugin's functions are usable.
- Test user interface using different methods to ensure it's ease of use.

### 3.6 Project Tracking Procedures

- **Gitlab**
    - Gitlab is used  to track code developed and give easier access to all team members.
    - Members track changes and revert to different versions of the project.
- **Weekly Meetings**
    - Every Tuesday, the team has a meeting with Professor Myra discussing what was completed in the previous week.
    - On Sunday, team members gather and discuss the task for the future.
- **Trello**
    - Team members use Trello boards to keep track of task status.
- **Slack**
    - The team uses Slack to discuss issues or details about tasks.

### 3.7 Expected Results and Validation

**Expected Results**
A desired outcome is to have a working plugin that assists scientists build biological models of living organisms with specific desired properties. Users view each biological part containing information of part name, type, number of uses, validation of stock. An expected final product for people interested in building DNA Feature Models to work on the task without restriction.

**Validation**
To validate the program, the team conducts through acceptance testing. Gathering review and feedback on what needs to be adjusted. Unit testing is another option to check if a project program is working properly as team's intention throughout the developing stage and finalizing.

## 4. Project Timeline, Estimated Resources, and Challenges

### 4.1 Project Timeline

The project timeline divides itself into four phases. Each phase consists of eight work weeks. These eight work weeks are subdivided according to team-agreed deadlines for completing tasks amounting to total project progress.

The timeline is presented in Figure 2, where a hierarchy of tasks are presented similar to a feature model. By judging current team progress, the feature model of the project timeline represents an achievable goal within two semesters given that each task in subphases are completed by members best attributed to the task. A Gantt Chart is presented in Figure 3 where milestones and tasks are broken down into their estimated time needed.

Plugin implementation has exclusive control by phases 3 and 4 due to the heaviness of the task compared to the fundamental plugin build completed in phases 1 and 2; earlier phases involve composition of solid foundations before creating the frontend aspect to ensure a reliable product.

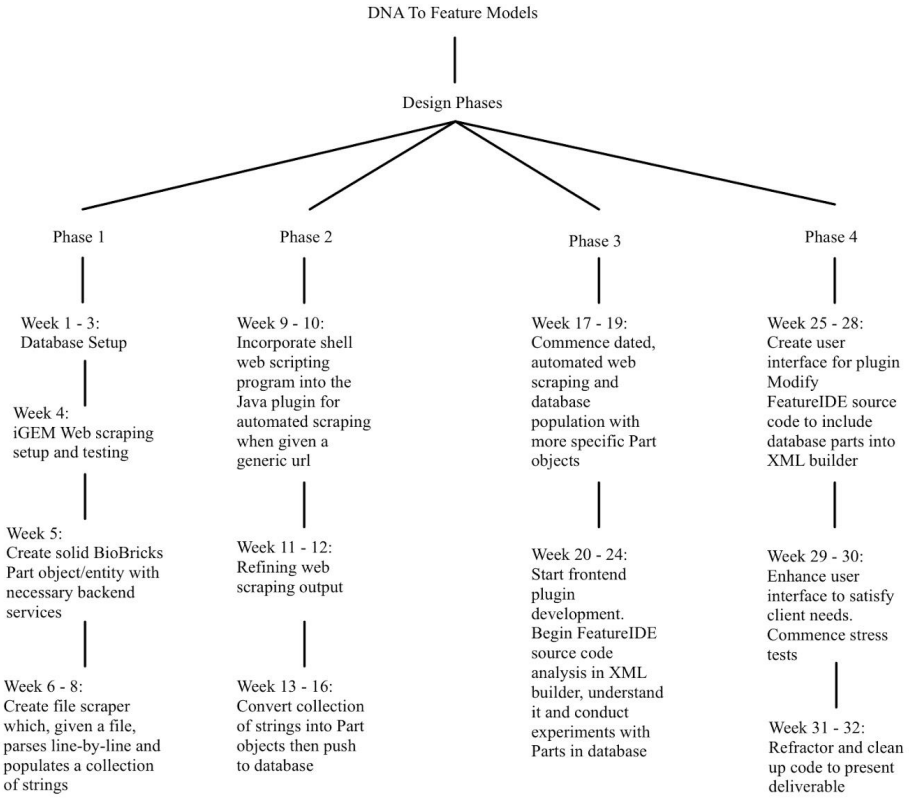



**Figure 2:** Timeline of the project presented as a hierarchy similar to a feature model. The project is divided into four design phases consisting of eight work weeks. Each phase breaks down a set of tasks to be completed by the expected work week deadline

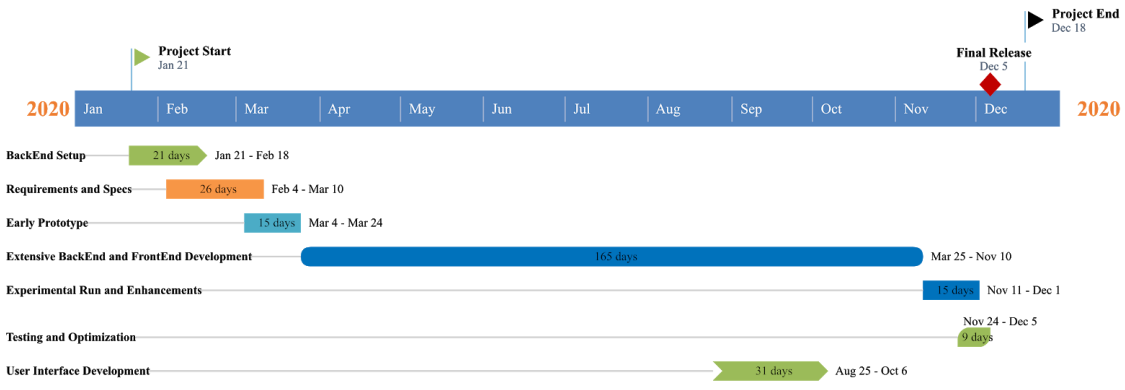



**Figure 3:** Gantt Chart presenting task and milestone breakdown with estimated times. This Gantt Chart is used as a progress tracker to make sure the team is in the expected design phase.

### 4.2 Feasibility Assessment

The project helps scientists view various models from BioBricks in an organized hierarchy. It is also an aid for everyone interested in building DNA Feature Models without any restriction. Foreseen challenges came in two aspects: poor change management and no long-term thinking. For the project to succeed, a big picture view was needed to complete the project. What this project does need is constant change throughout implementation; keeping track of the different changing curves, and commitment to those changes.

### 4.3 Personnel Effort Requirements

Tables were created to present task deduction and dedicated time allocated for each task as well as an explanation. These are shown below in tables 1 - 3.

| Task | Projected effort | Hours/Weeks | Explanation |
|---|---|---|---|
| Database Setup | low | 5 hours for 3 weeks | Building and creating the database needed for the project. Contains relevant information for software setup and data needed for the plugin. |
| iGEM web scraping setup and testing | low | 10 hours for 1 week | Focused on web scraping different data to be able to test, setup, and understand how web scraping works. |
| Create solid BioBricks Part Entity | low | 5 hours for 1 week | Worked on creating the solid BioBricks part object/entity with all the necessary backend services for full function. |
| File Parser | low | 5 hours for 2 weeks | Created the file parser that parses line-by-line and populates a collection of strings later to be used when creating part entities. |

**Table 1:** table showing tasks with low projected effort

| | | | |
|---|---|---|---|
| Frontend Development | Medium | 12 hours for 4 weeks | Started working on the frontend portion of the project to create the plugin and connect back end with front end |
| User Interface | Medium | 6 hours for 3 weeks | Creating user interface for plugin and creating how user accesses data in the plugin |
| Enhance UI | Medium | 8 hours for 2 weeks | Final touches for the UI and satisfying user needs |
| Testing | Medium | 4 hours for 1 week | Stress testing everything needed to move on |
| Refactoring | Medium | 6 hours for 2 weeks | Refactoring and cleaning up code to present the final deliverable. |

**Table 2:** table showing tasks with medium projected effort

| | | | |
|---|---|---|---|
| Shell Web Scraping | High | 6 hours for 2 weeks | This task was incorporating shell web scripting program into the Java plugin for automated scraping when given a generic URL |
| Refining Web Scraping Output | High | 10 hours for 2 weeks | Taking the web scraped output and translating it into XML |
| Converting Strings into Objects | High | 6 hours for 3 weeks | This trask was converting a collection of strings into part objects to be pushed to the database and for future usage |
| Commenced Automated Web scraping | High | 5 hours for 2 weeks | Commenced, dated automated web scraping, and database population with more specific part objects |
| Frontend Development | High | 12 hours for 4 weeks | Started working on the front end portion of the project to create the plugin and connect the backend with the frontend. |

**Table 3:** table showing tasks with high projected effort

### 4.4 Other Resource Requirements

- EclipseIDE
- Java Runtime Environment
- FeatureIDE Plugin for Eclipse
- MySQL Database
- Server Hosting

### 4.5 Financial Requirements

A breakdown of financial requirements for the project is shown in the below table. Decisions were made with cost efficiency and quality in mind.

| **Resource** | **Cost** |
|---|---|
| Iowa State Provided Server | Free |
| Git Provided by Iowa State | Free |
| Equipment | $300 |
| Personal Computers | Free |
| Online lessons for Plugin Development | Free |
| **Total** | **$300** |

**Table 4:** cost breakdown for the project

# 5. Testing and Implementation

## 5.1 Interface Specifications

The project is entirely software development and coding. There are multiple ways in which a software project can be tested. This project is tested using:

- Functionality testing
  - Trying out every function implemented and making sure functionality is correct.
- Mockito Tests
  - Used to test part information parsing and database behavior.
- Review of full code
  - Presenting code to a professional to allow them to discover any problems that will cause bugs in the future
- CI/CD
  - Keep the server running automatically and detect compiling issues.
  - Immediately push latest updates to code.
- Code analysis:
  - Testing the code with a software called PMD Java that reveals security vulnerabilities and concurrency issues.
  - Encoded in Eclipse to automatically check for the aforementioned issues.
- JUnit Tests
  - Verify the wanted construction of feature models.

## 5.2 Hardware and software

Software used in the testing phase include:

- **PMD:** used to scan code written and shows some potential bugs and problems that may occur. This is useful because it can increase performance, complexity of code, and assists debuggers in removing all bugs.
- **Checkstyle:** used to improve code adheres and makes sure the style of each class written is similar to the other. This is useful because code is pushed to the database by 5 different people and each one will have a different coding style.
- **SecurityBugs:** used to show bugs that MAY occur in the future and helps write code to be able to avoid such bugs. This is useful because it helps save time by avoiding the occurrence of different bugs.

## 5.3 Functional Testing

The project follows a simple testing scheme. The testing schemes are defined as follows:

- **Unit Testing:**
  The team tests each method introduced to the server for correctness and performance. As more functionality is added to the backend, complex unit tests to validate stability and behavior will hence be added. These tests will be conducted on web scraping, data parsing and database to validate our backend development.

- **Interface Testing:**
  GUI is an important component of any software. The team's goal is to build a simple interface that lets users choose parts using a drop-down menu. Testing is done through simulation and on-click tests then users will be asked for feedback on implementation design to ensure a familiar and easy GUI design. Most of the interface testing follows manual testing since it provides better debugging results.

- **Integration Testing:**
  Since frontend and backend dependence is essential, integration is a crucial part of the project that brings all components together. Testing is conducted for basic authentication and communication between the server and frontend to establish stable ground. Features will be tested as they are added to ensure stability, performance, and security standards until all desired features are implemented.

## 5.4 Non-Functional Testing

Tests are conducted to establish stability, upgradability, usability, security, and performance. Mockito testing is used as database capacity increases to ensure the desired efficiency for the backend. The team will strain the server to ensure its ability to handle many users at the same time. As for the frontend, on-click testing is used for various scenarios to ensure a smooth and satisfactory user experience.

## 5.5 Process

The team will start off by a bottom-up approach to PMD and SecurityBugs. This will involve adding codes that need to be checked and scanned through PMD and SecurityBugs, which will analyze codes and project the potential bugs or problems that could occur in future. For the testing process, team members go through the analyzed code and make changes to the component such as functions, class and interface. Due to working in a team project, code needs to be in a consistent style. By using Checkstyle, the team's code ended up with a similar style of code pushed to the repository and easier to understand the portion of what other members worked on.

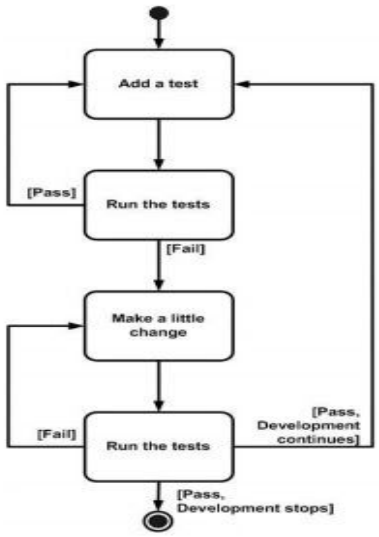



**Figure 4:** Diagram showing Basic testing flow. This diagram shows how the team's coding and testing has been processed involving PMD, SecurityBugs, and Checkstyle.

## 5.6 Results

The team faced several issues while working on the project. The first issue the team faced was "Error with web scraping." During the phase of web scraping, the team's final goal is to have consistent data from the Biobrick repository. However, the team struggled due to having unique characters throughout the output file from web scraping. This was resolved by having an assumption of having Operation System's base language setup other than English might cause the problem. By working on English based OS, the web scraping function got resolved and fully functional.

Teams are building file parsers to send output files to the database. In future, the program will consist of a plugin that we automatically parse files to the database.

# 6. Closing Material

## 6.1 Conclusion

DNA to Feature Models has been an interesting journey for the team. Exploring families of SPLs, SPLs and Feature Models lead us to a better understanding of representing the project. Experiments conducted thus far have been a success; a database for a catalog of parts based on the BioBricks Repository has been established. Parts can be extracted using controllers with respect to certain criteria.

Testing has been successful as interactions with the backend through a designated request service was successful. The project's achievements during the first two phases exceeded our expectations; the last phases of the project appear to be full of excitement. Successful project planning led the team to this outcome. With the unfortunate COVID-19 outbreak and pandemic, the team created a contingency plan; following the plan ensured project progress remains efficient and team members safety a priority.

## 6.2 References

Previous works are referenced below. The team thanks all the information contributed by these sources and their availability.


- D. N. KTH, D. Nešić, J. Krüger, Ș. Stănciulescu, Ș. Stănciulescu, T. Berger, T. Berger, Kth, Jacob Krüger University of Magdeburg, University of Magdeburg, Abb, Abb, Chalmers University of Technology, Chalmers University of Technology, University of Tartu, Saarland University, and Imperial College, "Principles of feature modeling," Principles of feature modeling | Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 01-Aug-2019. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3338906.3338974. [Accessed: 23-Jan-2020].
- M. Cashman, M. B. Cohen, M. B. Cohen, W. Niu, Mikaela Cashman Iowa State University, Iowa State University, Iowa State UniversityView Profile, Justin Firestone University of Nebraska-Lincoln, Justin Firestone, University of Nebraska-Lincoln, University of Nebraska-LincolnView Profile, Iowa State University, Suranaree University of Technology, Suranaree University of Technology, Wei Niu University of Nebraska-Lincoln, Chalmers | University of Gothenburg, University Lille, Danfoss Power Electronics A/S, University of Namur,Humboldt-Universit, University Paris, IK4-IKERLAN Research Center, Sorbonne University, and TU Braunschweig, "DNA as Features: Organic Software Product Lines," DNA as Features | Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A, 01-Sep-2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3336294.3336298. [Accessed: 23-Jan-2020].
- T. Thum and J. Meinicke, "FeatureIDE," FeatureIDE. [Online]. Available: https://featureide.github.io


## 6.3 Appendices

- *Principles of Feature Modeling*- Damir Nesic, Jacob Kruger, Stefan Stanciulescu, Thorsten Berger
- *DNA as Features: Organic Software Product Lines*- Mikaela Cashman, Justin Firestone, Myra B.Cohen, Thammasak Thianniwet, Wei Niu
- FeatureIDE source code

**JSON EXAMPLE Object**

```
{

   "partFunction": "generic function",

   "partType": "Plasmid",

   "partName": "BB_123",

   "pcr": "ccaaggg"

}
```


Software Bugs encountered during the creation of the project include:

- **Exception handling**
    - Dealing with FileNotFound exceptions.
    - Handling NullPointer Exceptions
- **Error handling**
    - Using debuggers to find sources of bugs.
    - Using break statements.
- **Local database persistency**
- **Database setup without all required servers running such as Apache and MySQL serve**