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: 03.29.2020/Version2

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:** 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

SPLs 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 and uses Java 8 and the FeatureIDE plugin for Eclipse .

1.4 Requirements

- Compatible machine with Eclipse and FeatureIDE plugin installed
- Access to the Biobrick repository with the extension plugin on Feature IDE
- Web crawling and scraping
    - Information gathered is used to translate information and make it compatible with Feature IDE
    - This provides users to build software product lines of DNA Feature Models.

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

Assumptions:

- Users with and without knowledge of feature models can build feature models of DNA.
- The end product provides access and can used anywhere with internet access to 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, so 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.

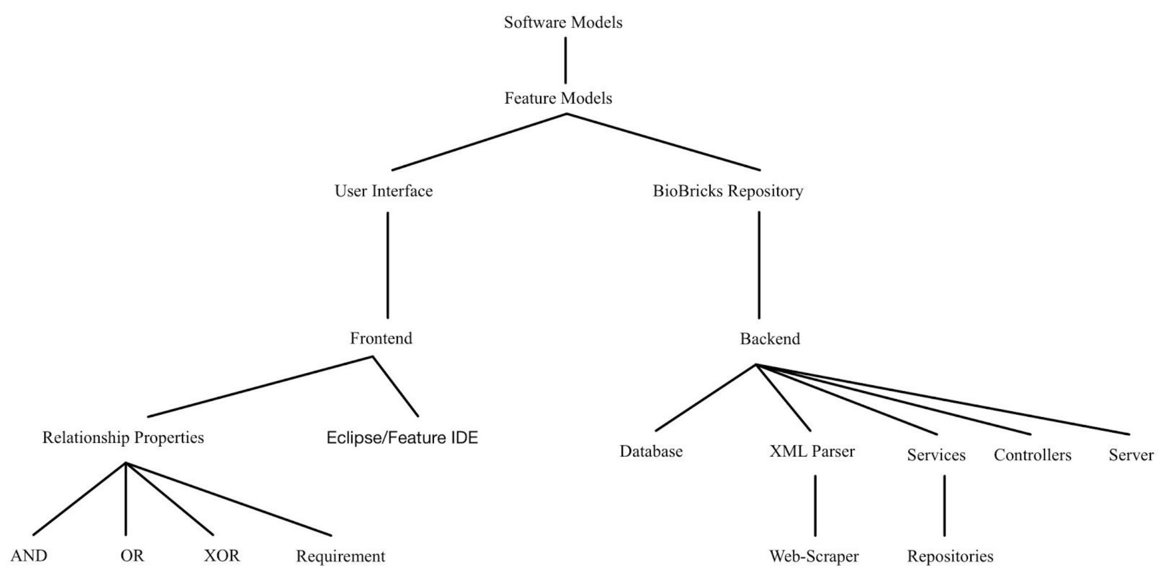


Figure 1: the flow of project requirements and dependencies presented in a hierarchy. The figure is modelled similarly to a feature model.

3. Statement of Work

3.1 Previous Work And Literature

1. *Principles of Feature Modeling*- Damir Nesic, Jacob Kruger, Stefan Stanciulescu, Thorsten Berger
2. *DNA as Features: Organic Software Product Lines*- Mikaela Cashman, Justin Firestone, Myra B.Cohen, Thammasak Thianniwet, Wei Niu
3. 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.

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 currently relies on an individual server that stores all BioBricks data. The data is hard coded into our program. While this is not a huge deal, there might be better ways to handle this.

There was a design tradeoff decision made between using a live web scraper and one updated once 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

Tasks are divided such that each member has a role best attributed to their abilities. Team members whose strengths depend on backend programming are exploited in the backend. Members with experience in the frontend are utilized in that area. Skills in experimentation and research are also utilized to provide optimal results for both frontend and backend programming. Some of these experiments include setting up the database and population with temporary objects as a test, and manipulation of the web scraping program to best suit the scope of the project and the eyes of the user when exporting hte parts from the database in the final product.

Team Members and Roles are Listed Below:

- Abdul Rahman El Moughrabi - Developer/Documentation Management
- Ahmad Nazar - Team Leader/Developer
- Ahmed Alketbi - Developer/Debugger
- Hyegeun Gug - Developer/Web Management
- Prathik Nair - Debugger/Developer

3.4 Possible Risks And Risk Management

While most of the project work is progressing as predicted; the pandemic crisis has split the team across the world, forcing remote interaction. While this is not a huge issue, 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; but this does remain to be seen.

Include any concerns or details that may slow or hinder your plan as it is now. These may include anything to do with costs, materials, equipment, knowledge of area, accuracy issues, etc.

3.5 Project Proposed Milestones and Evaluation Criteria

- Establish connection to Biobrick repository to gather bio-part information

- Store bio-part information to database
- Build Feature Models based on database
- Update database every year when changes are made
- Improve plugin functionality
- Enhance user interface

3.6 Project Tracking Procedures

- Gitlab
  - For the project, the team uses Gitlab 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 has gotten done for last 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 program 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. With expected results for the project, people interested in building DNA Feature Models can 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.

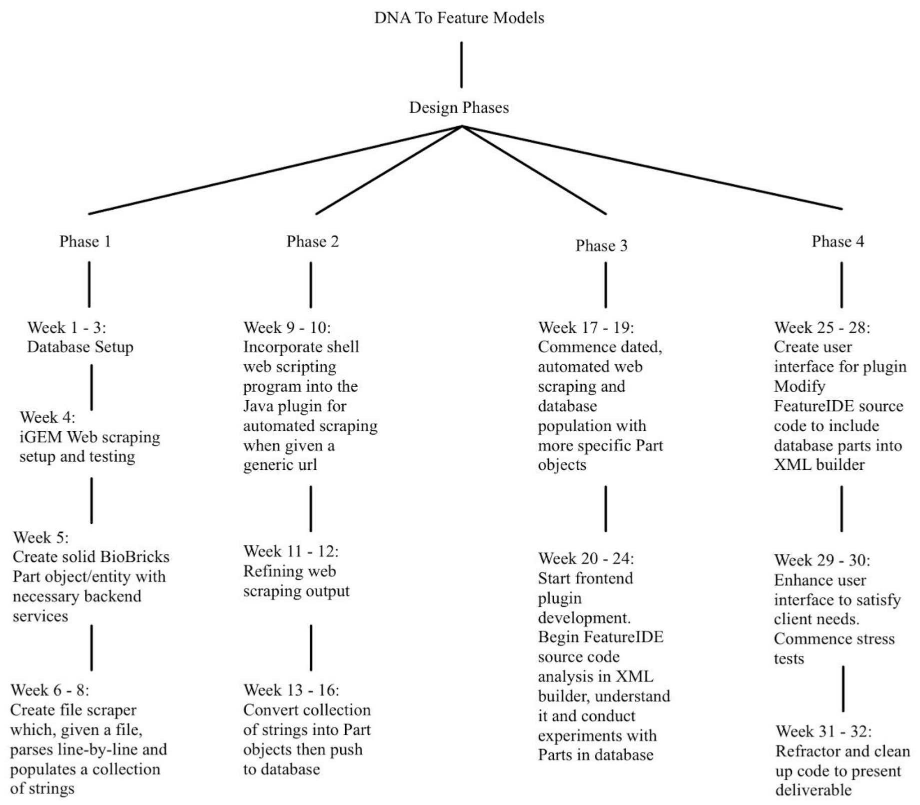
4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

The project timeline divides itself into four phases such that each phase consists of eight work weeks. These eight work weeks are subdivided according to team-agreed deadlines for completing certain 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.

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.



DNA To Feature Models
Design Phases
Phase 1
Phase 2
Phase 3
Phase 4
Week 1 - 3: Database Setup
Week 4: iGEM Web scraping setup and testing
Week 5: Create solid BioBricks Part object/entity with necessary backend services
Week 6 - 8: Create file scraper which, given a file, parses line-by-line and populates a collection of strings
Week 9 - 10: Incorporate shell web scripting program into the Java plugin for automated scraping when given a generic url
Week 11 - 12: Refining web scraping output
Week 13 - 16: Convert collection of strings into Part objects then push to database
Week 17 - 19: Commence dated, automated web scraping and database population with more specific Part objects
Week 20 - 24: Start frontend plugin development. Begin FeatureIDE source code analysis in XML builder, understand it and conduct experiments with Parts in database
Week 25 - 28: Create user interface for plugin Modify FeatureIDE source code to include database parts into XML builder
Week 29 - 30: Enhance user interface to satisfy client needs. Commence stress tests
Week 31 - 32: Refractor and clean up code to present deliverable

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

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

Table 1: table showing tasks with low projected effort

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	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 Scraper	low	5 hours for 2 weeks	Created the file scraper that parces line by line and populates a collection of strings

Table 2: table showing tasks with medium 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 3: table showing tasks with high projected effort

Shell Web Scripting	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.

Tables 1 - 3: tables depicting a task-by-task deduction for the project.

4.4 Other Resource Requirements

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

4.5 Financial Requirements

- Server hosting for database and backend support (provided by ISU).

5. Testing and Implementation

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

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study for functional and non-functional requirements)
2. Define the individual items 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've determined.

5.1 Interface Specifications

– Discuss any hardware/software interfacing that you are working on for testing your project

- JUnit Tests are used to verify the wanted construction of feature models.
- Mockito Tests are used to test part information parsing and database behavior.
- CI/CD to keep the server running automatically and detect compiling issues.

5.2 Hardware and software

– Indicate any hardware and/or software used in the testing phase

– Provide brief, simple introductions for each to explain the usefulness of each

5.3 Functional Testing

Examples include unit, integration, system, acceptance testing

5.4 Non-Functional Testing

Testing for performance, security, usability, compatibility

5.5 Process

– Explain how each method indicated in Section 2 was tested

– Flow diagram of the process if applicable (should be for most projects)

5.6 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 it as you progress with your project
- – If you are including figures, please include captions and cite it in the text

• This part will likely need to be refined in your 492 semester where the majority of the implementation and testing work will take place

-**Modeling and Simulation**: This could be logic analyzation, waveform outputs, block testing. 3D model renders, modeling graphs.

-List the **implementation Issues and Challenges**.

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

This will likely be different than in project plan, since these will be technical references versus 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 that does not directly pertain to the problem but helps support it, include that 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.