The Mother of all Elaborations

MOAP

Dwight Naylor

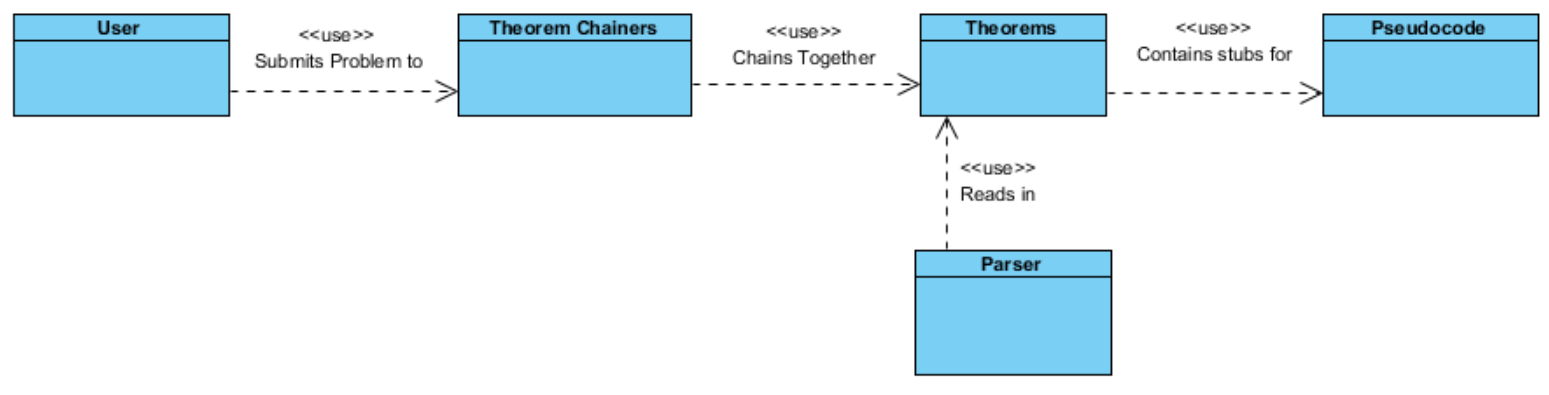
Wyler McAninch-Ruenzi

Zexin Wan

Daniel Zampini

Our group has submitted a fixed-up inception

Domain Model Diagram



Our high-level domain is very simple. The **User** inputs a problem, which enters into the **Theorem Chainer**, or **Problem Solver**. The solver uses existing Theorems, which were read in by the **Parser**, to solve the problem. The Solver then backtracks through these theorems, outputting the **Pseudocode** stubs attached to each one.

Supplemental Specification

|  |  |  |  |
| --- | --- | --- | --- |
| *Functionality* | | | |
| **Number** | **Requirement** | **Description** | **Priority** |
| F1 | Program must be able to solve some algorithm problems | While we don't hope to be able to solve every programming problem we may be given, the program should be able to solve a fair number of them. | Must |
| F2 | Step-by-Step explanation on output | After the problem is solved by our system and outputs the algorithm, it will also output a step-by-step list detailing how the meta-algorithm came to the solution. | Must |
| F3 | Help page | There should be a help page which tells the user how to use the input language, as well as providing a few simple examples | Must |
| F4 | Theorem Database | Database must include a wide range of theorems, including information on:   * Basic data structures like arrays, sets, lists, strings, etc * Basic math and logic operations | Must |
| F5 | Basic runtime-cost analyser | Able to do a basic approximation of runtime-cost of an algorithm when constructed, so that the best solution can be chosen. The distinction between this and a more advanced solution is that this solution is only required to distinguish approximately the best algorithm, and is not expected to return optimal results. | Must |
| F6 | Syntax errors and warnings | The program must alert the user to incorrect / inadvisable syntax in their problem expression. This must happen before the query is processed by the problem solver. | Should |
| F7 | Detailed runtime-cost analyser | This would be an extensive implementation of a runtime analyser, able to give a more precise estimation of runtime so that the meta-algorithm can better choose a solution. This requires a lot of research and right now we believe a basic implementation is good enough, but this is a long-term goal | Won't |

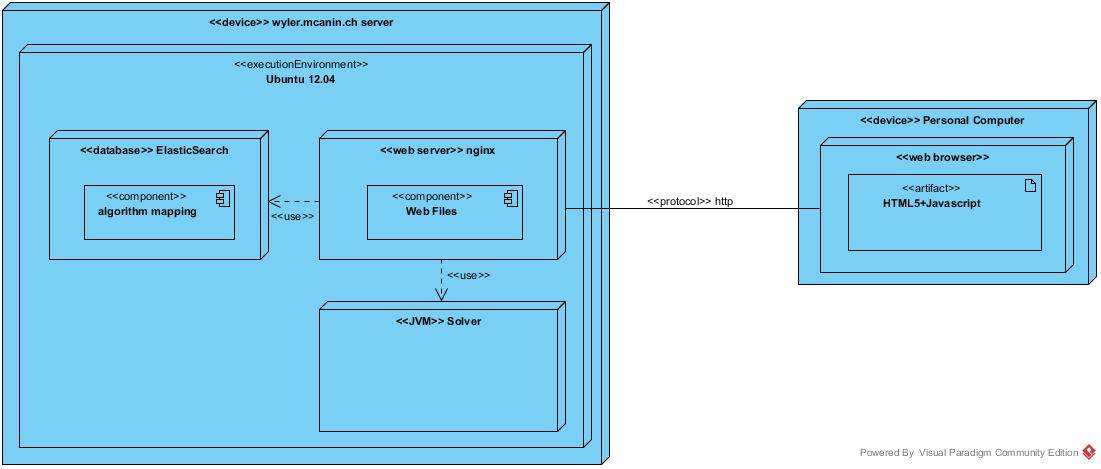
|  |  |  |  |
| --- | --- | --- | --- |
| *Usability* | | | |
| **Number** | **Requirement** | **Description** | **Priority** |
| U1 | The Input language must be able to express all imaginable problems | With some exceptions, we expect the input language to be capable of correctly expressing all algorithm problems we come across. A tentative list of these problems is under construction. | Must |
| U2 | Simple Web-UI | We only need a textbox for input, a button to request a solution and a link to our help page on the landing page of the website. | Should |

|  |  |  |  |
| --- | --- | --- | --- |
| *Reliability* | | | |
| **Number** | **Requirement** | **Description** | **Priority** |
| R1 | The website and server must be up | While it is not the core part of the project, the availability of a web server and website are important for this prototype. | Must |
| R2 | Predictable output | Once someone has an understanding of how the meta-algorithm works, it must produce predictable results. This means that the system will reliably give similar answers to similar problems | Must |

|  |  |  |  |
| --- | --- | --- | --- |
| *Performance* | | | |
| **Number** | **Requirement** | **Description** | **Priority** |
| P1 | Fast solution time | Due to the nature of the input (usually very small computer problems) The program should be able to deliver algorithm solutions relatively quickly. We don't anticipate any problems in this regard. | Must |
| P2 | Quick website loading | The website should load within a few seconds, as the starting page is very minimal. | Should |

|  |  |  |  |
| --- | --- | --- | --- |
| *Supportability* | | | |
| **Number** | **Requirement** | **Description** | **Priority** |
| S1 | JUnit testing | There should be a JUnit testing project to which we can add problems and input continuously to avoid regression and ensure the program works correctly. This is up and running. | Must |
| S2 | Expandable theorem and algorithm databases | We want the databases to be easily expandable to add new entries. New entries into the system should not interfere with the previous solutions, except where they directly improve the solution.. | Must |

Deployment Diagram



Use Case(s)

|  |  |
| --- | --- |
| **Use Case** | Solve Problem |
| **Identifier** | UC1 |
| **Description** | The Solve Problem use case models a user entering an algorithm problem and getting a solution. |
| **Actors** | User, Problem Solver |
| **Preconditions** | User knows syntax, is on the website |
| **Flow of Events** | 1. User enters computer algorithm problem in formal logic    1. Failure case: Computer shows any syntax errors to the user, and the user fixes the displayed syntax errors and resubmits the problem    2. Failure case: The user is unfamiliar with the program syntax, and has to look it up on the help page before entering their problem 2. Solver outputs an algorithm in pseudocode that will solve the user’s problem    1. Failure case: User is unhappy with solution, re-enters problem |
| **PostConditions** | User has a solution to their problem |

Work Breakdown Structure

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Assignee** |
| Theorem database | The database used by the program to derive all possible information about a given problem state | Daniel |
| Core Solver | The solver which transitions the problem between problem states to reach a solution | Dwight |
| Website setup | The website, UI and server arrangement that makes the website usable to query the program | Wyler |
| Expression simplifier | A simplifier for converting all possible representations of the same problem into one form | Zexin |
| Algorithm output system | A system for backtracking when a solution is reached to output the pseudocode for the algorithm | Dwight |
| Runtime analyzer | The cost analyzer that | all |
| Project executive manager | Makes sure all project deadlines are met and everything is handed in on time | Daniel |

Updated Project Schedule

|  |  |
| --- | --- |
| **Week** | **Goals** |
| 10/18 | Hook up working website with UI, Get solver working on larger group of sample problems including basic algorithms problems, and finish the prototype expression simplifier |
| 10/25 | Build on theorem databases and meta-algorithms to solve even more problems, expanding on some simple data structures like stack and queue |
| 11/1 | Improve website UI to achieve better user experience. Improve output format for increased readability and clarity |
| 11/8 | Create help page for syntax and begin to create complex meta-algorithms, such as dynamic programming and bisection algorithms |
| 11/15 | Begin integration of more complex data structures (hashtables, special trees, graphs, etc) |
| 11/22 | Build a list of problems the program can solve and test and debug to assure that it works on everything in this list |
| 11/29 | Prepare for demo. |

Contribution Summary

Domain Model – Dwight

Supplemental Specifications – Daniel (and some Dwight)

Deployment Diagram – Wyler

Use Case – All

WBS – Zexin, Dwight

Updated Schedule – Zexin

Contribution Summary – Dwight, the coolest team member

Status Report – Dwight

Assembly and touchup – Dwight

Status Report

Our status report is included as a list of our supplemental specifications, with progress along each task.

|  |  |  |
| --- | --- | --- |
| **Number** | **Requirement** | **Progress** |
| F1 | Program must be able to solve some algorithm problems | The program is able to solve several small sample problems, but is still far from being able to solve the problem range we eventually hope for. |
| F2 | Step-by-Step explanation on output | Nothing yet |
| F3 | Help page | Nothing yet |
| F4 | Theorem Database | We have a basic database right now, nothing more |
| F5 | Basic runtime-cost analyser | We have a very crude runtime analyzer set up |
| F6 | Syntax errors and warnings | This is done automagically with xtext, we just have to hook it up to the web interface |
| F7 | Detailed runtime-cost analyser | Not Happening. |
| U1 | The Input language must be able to express all imaginable problems | This is under heavy research at the moment. It’s our hope that we will reach a language capable of doing this. We can currently tackle a fair variety of problems, but we hope to get many more. |
| U2 | Simple Web-UI | Pretty much done, just needs to be hooked up! |
| R1 | The website and server must be up | Can’t be tested until we set up a real website | |
| R2 | Predictable output | Can’t be tested until our input language is decided upon | |
| P1 | Fast solution time | Things run very fast! | |
| P2 | Quick website loading | This just should not be an issue. | |
| S1 | JUnit testing | We have a Junit project up and running, we’ll add more to it as needed. |
| S2 | Expandable theorem and algorithm databases | The theorem database is still under design. We won’t know how well expansion works until we’ve settled on a design for it. |