The Mother of all Stakeholder Review #1's

Dwight Naylor

Daniel Zampini

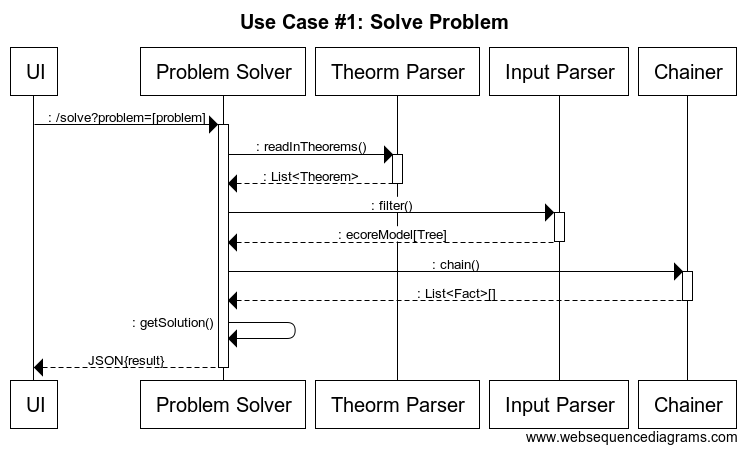
Zexin Wan

Wyler McAninch-Ruenzi

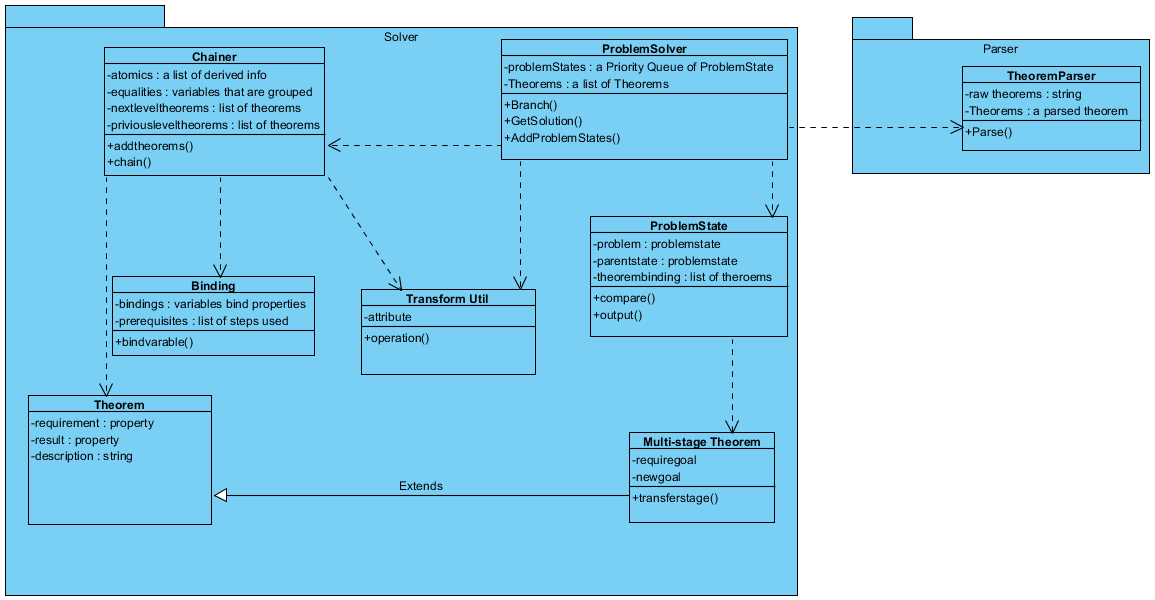
(We also updated our elaboration’s WBS)

Sequence Diagram

(We only have one use case)



Static Class Diagram



CRC Cards

We used Cohesion ratings from 1-7 and Coupling ratings from 1-5 based on the CRC Cards Powerpoint provided in class.

**Theorem**

* Stores data for expanding a problem state
* Coupling Rating: 3- It is slightly coupled with the Theorem Chainer
* Cohesion Rating:  7- It only contains data on the theorem’s themselves
* Collaborators: Chainer, Problem State

**Web API**

* Pass user input to the solver and respond with result
* Coupling Rating: 4- Some reliance on problem solver
* Cohesion Rating: 7- Entirely internalized
* Collaborators: Problem Solver

**XText Parser**

* Reads string problem into Ecore model
* Coupling Rating: 5- Relies on no other classes to parse text
* Cohesion Rating: 7- Only parses input
* Collaborators: Input Utilities, Transform Utilities

**Input Utilities**

* All basic operations on the input model
* Coupling Rating: 3- Slightly coupled with Transform Utilities
* Cohesion Rating: 7- Contains only methods to perform basic operations on input
* Collaborators: Problem Solver, XText Parser, and Transform Utilities

**Transform Utilities**

* Provides higher-level transforms for the input model
* Coupling Rating: 3- Slightly coupled with Input Utilities
* Cohesion Rating: 5- Functions within are sometimes unrelated, as they are utilities
* Collaborators: Problem Solver, Input Utilities

**Problem Solver**

* Gets from a problem to a solved state
* Coupling Rating: 2- Requires interaction with problem state and chainer
* Cohesion Rating: 7- Contains only methods to solve problem
* Collaborators: Problem State, Chainer

**Problem State**

* Stores a problem and history of how we arrived at that state
* Coupling Rating: 3- Interacts heavily with theorems and eventually the problem solver.
* Cohesion Rating: 6- Only stores steps to get to the problem solution.
* Collaborators: Theorem, Problem Solver

**Chainer**

* Chains theorems to discover everything about a problem
* Coupling Rating: 3- It is coupled with the theorem class
* Cohesion Rating: 4- Can involve problem solutions
* Collaborators: Theorem, Problem Solver

Design Approach

Our design approach is intended to closely represent the internal structure of the program. The **Problem Solver** works by chaining **Problem States** together with **Multitheorems** until a solution is reached. Each of these **Problem states** is explored more thoroughly by a **Chainer**, which expands on the provided information using **Theorems** to derive all possible **Facts** about that part of the problem. When this is done, the program backtracks through the problem states, collecting pseudocode stubs from each one in order to output the final pseudocode.

We've captured a lot of ancillary functionality in two utility classes, **InputUtil** and **TransformUtil**, which perform simple basic operations and more advanced transforms on our object model, respectively.

The **Web-UI** is designed to be simple and user friendly. It is built in jsx with React.js utilizing its state model and using jQuery for ajax requests.

We use an xtext framework for our parser, which provides auto-generated libraries for our parsing and underlying object model for input. This framework generated a composite-based model for representing input. We are able to interact with this system via a facade which gives us access to a parse() method.

Contribution Summary and Status Report

Demo - all

Sequence Diagrams - Wyler

Static Class Diagrams - Zexin

CRC Cards - Daniel

Design Approach - Dwight

Contribution Summary - Wyler

Status Report - Dwight, Wyler

|  |  |  |
| --- | --- | --- |
| **Number** | **Requirement** | **Progress** |
| F1 | Program must be able to solve some algorithm problems | We've got a good handle on the problem range we want to solve, and we've started tackling |
| F2 | Step-by-Step explanation on output | Nothing yet |
| F3 | Help page | Have a page set up, but with no info on it. |
| F4 | Theorem Database | We have a basic database right now, nothing more |
| F5 | Basic runtime-cost analyser | We have a very crude runtime analyzer setup |
| 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 | Done! UI can take input and pass it through to a server, which then provides a response to the user. |
| R1 | The website and server must be up | Website is up and the server can be started on notice. |
| R2 | Predictable output | Can’t be tested until our input language is decided upon |
| P1 | Fast solution time | Things run very fast! (most of the time...) |
| 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. |