Software Engineering 2: Project Teamwork

Final project report

A visual simulator for high-quality planning domain models

Mälardalen University

Group 2

January 16, 2019

Contents

List of Tables

List of Figures

1 Background

Ericsson is a Swedish multinational company dedicated to the development of projects related to the area of telecommunications, mobile technology, artificial intelligence applications and communications on the network, headquartered in Stockholm. At the moment, the company is taking part in a European Union funded project called SCOTT [1]. This project is collaborating with other big companies like Bosch or Sonic, and aims to take advantage of cross-domain integration in various industrial fields by strengthening Europe's position in technology and intelligent infrastructure processes.
The actual case that we have is part of the SCOTT project, in which our tool will be used in an automated warehouse where Artificial Intelligence planning is used to plan tasks for robots. To make this possible, PDDL (Planning Domain Definition Language) is used, which aim is to standardize domain planning and problem description languages in AI planning.
For the project in the course Software Engineering 2 at Mälardalen University, we contacted Swarup Kumar Mohalik, our contact person and representative for Ericsson located in Delhi, India. The client requested us to create a tool that would serve their company to be able to either import or edit a textual PDDL domain model and visually simulate it after. The tool is Windows PC-based, is developed in C# (.NET) and provides a visual and textual simulation GUI that presents the current state of the PDDL tree and all the actions that are possible. The visual simulation of the states provides the user with an overview of how any potential deadlocks stages have occurred, and when selecting one of the possible actions, the state of the tree updates. It also provides the functionality of returning to a previous state, and resume the visualization from there.

1.1 Terminology

PDDL: It is a language used to define domain models specification and planning problems, introduced to standardize AI planning languages. It consists of two parts: the Domain definition and the problem definition. [2]

- **Domain definition**: The domain definition contains the domain predicates and operators (called actions). It may also contain types, constants and static facts. Since the domain definition is critical for a correct plan generation, it is necessary for our client to have a tool that visualizes these domain models.
- **Problem Definition**: The Problem Definition contains the objects of the modelled world, the initial state of the model and the desired goal state.

Artificial Intelligence (AI): Combination of algorithms for creating machines able to mimic human intelligence.
State tree: A representation of states in a tree structure (map of possible actions to be carried out from a specific state).

Fluent: A feature introduced in version 2.1 of the PDDL language. In the most basic form of PDDL, objects are modelled in binary states. Fluents allows for modelling of numerical values in the environment, such as fuel-level for example.

2 Organization

The project was appointed to Group 2 of the Software Engineering course students. The group is a mix of nationalities and backgrounds, thus making possible a nice interrelation of experiences and qualifications. With the directions given by the examiners of the course, the group made the work possible by dividing the responsibilities and tasks. All the development and documentation was carried out by the group members, which got weekly feedback from the steering group (consisting of Jan Carlson and Robbert Jongeling).
The university course included, as well, constant communication with the company contact, which, unfortunately, was almost absent in all the meeting sessions via Skype and email. This resulted in the group working based on the project requirements and initial specifications, with the only proper meeting with the client being the one on December 14th.

2.1 Organization and Routine Changes

Throughout the duration of the project, many changes have been done to the distribution of work and responsibilities to the group members. Initially, all team members worked together to decide on the organization of the work and possible solutions to the problem presented to us by Ericsson.
During the design phase, work was split between all members in order for every task to be done efficiently. Later, after getting more information on the project, the group decided to split into smaller groups in order to divide the work. However, important aspects that the whole group should acknowledge were discussed and explained to all group members.
Further into the project, the group came to the conclusion focus should be on the following three points: the parsing of the PDDL, its implementation on the tool (Back-End part) and the development of the GUI of the tool. This lead to a division of the members and reallocation in several subtasks for more specific research and work done on every aspect of the tool.
Each sub-group worked on their assigned task according to all the plans and decisions made by the whole team. Periodically, meetings between two or all the groups were held where specific features or implementations to be done in collaboration were discussed. Lastly, some people from the groups separated to work in a documentation group. The documentation group were responsible for finishing the written deliverables needed for the course.

3 Project Work

3.1 Worked Hours

The distribution of the worked hours per member is displayed in the table 1

Member	W45	W46	W47	W48	W49	W50	W51	W52	W1	W2	W3	Total
Allan	15	22	21	18	20	19	10	2	0	20	12	159
Ermal	8	20	19	18	10	18	11	6	10	21	12	153
Erik E	8	25	14	22	22	23	18	0	5	20	10	167
Erik P	8	19	14	19	19	19	10	0	10	19	8	145
Johannes	9	25	13	20	19	21	15	0	5	16	8	151
Laura	8	18	12	20	17	25	12	0	10	20	12	154
Marta	9	16	17	18	22	19	12	10	7	16	10	156

Table 1: Worked Hours

3.2 Total Project Effort

Table 2 shows the working hours in person days for each task. The row *"total project meetings"* consist of the hours spent on group and steering meetings. The row *"total project presentations"* include the hours spent on presentations during the steering meetings and during the three main presentations. The actual effort in person days is calculated with 4 hours counting as one day. The reason for that is that the course is running as a half-time course where a full workday is equal to 4 hours of work and not 8. To calculate the effort for each task, we added the hours each member has put for each task and divided those hours with 4 to obtain the total effort in person days.

Task	Actual effort (person days)
Total project meetings	68
Total project presentations	8
Research	21
Documentation	51
Testings	4
GUI development	47
Back-end development	73
Total	272

Table 2: Tasks' effort

4 Project results

4.1 Key features of the project

4.1.1 Changes from the initial design proposal

The biggest changes from the initial design are the use of graphviz for generating and drawing the graph. The benefit of this is that graphviz automatically creates the layout, instead of us having to calculate it ourselves.
Secondly, instead of evolving the graph after each taken action, we are now generating the full graph from the start.

4.1.2 Showcase of the tool

1) Starting the program.

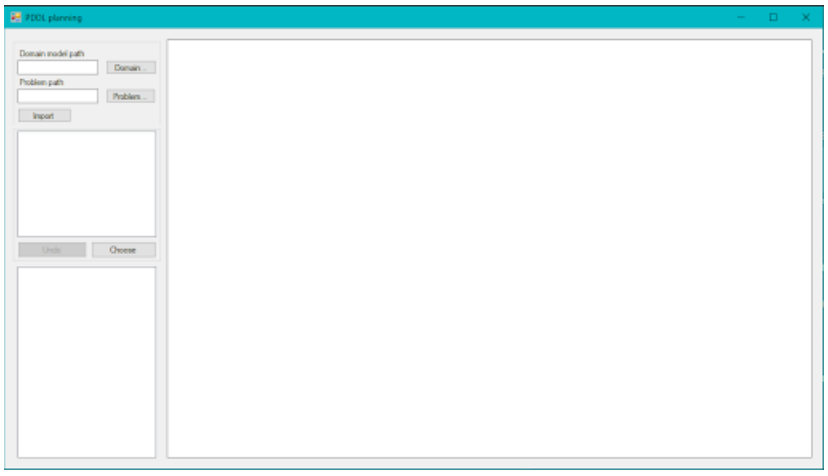

Figure 1: Screen Image 1

2) Choosing the files.

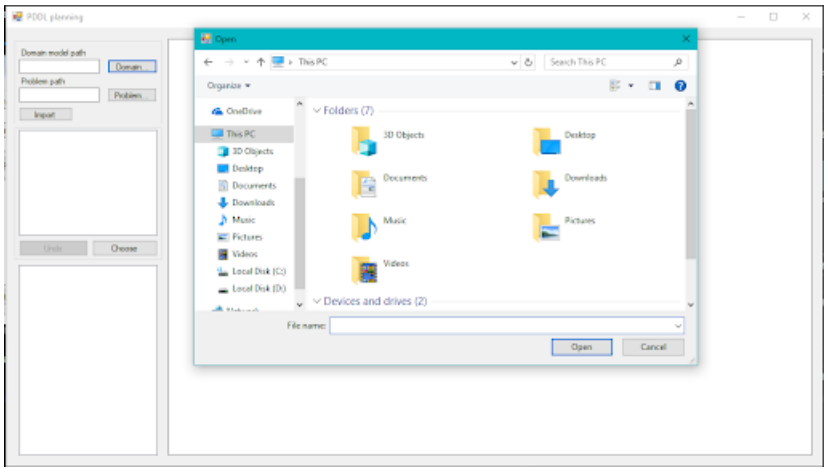

Figure 2: Screen Image 2

3) After clicking import, the state tree is generated.

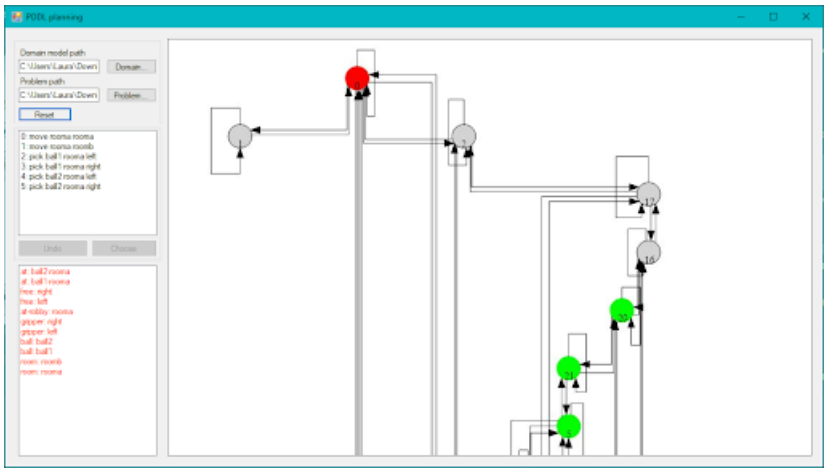

Figure 3: Screen Image 3

4) Choosing an action leads us to a new state, indicated with the red color.

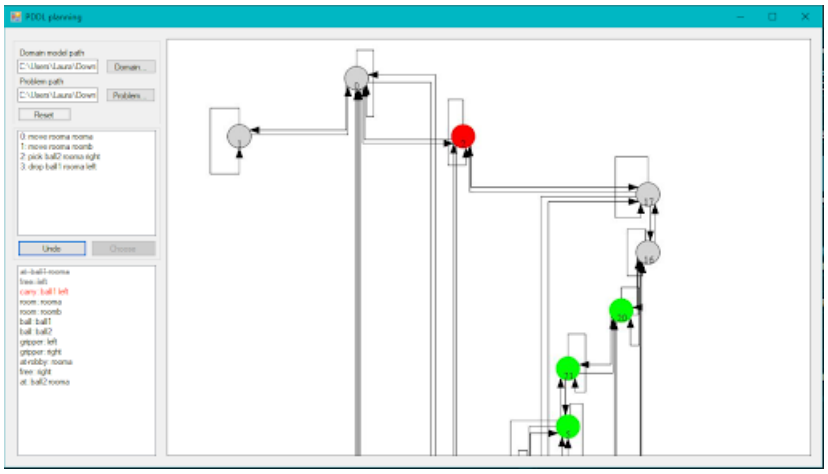

Figure 4: Screen Image 4

5) When clicking on an action, it colours the arrow in blue, showing us to which state the action will take us.

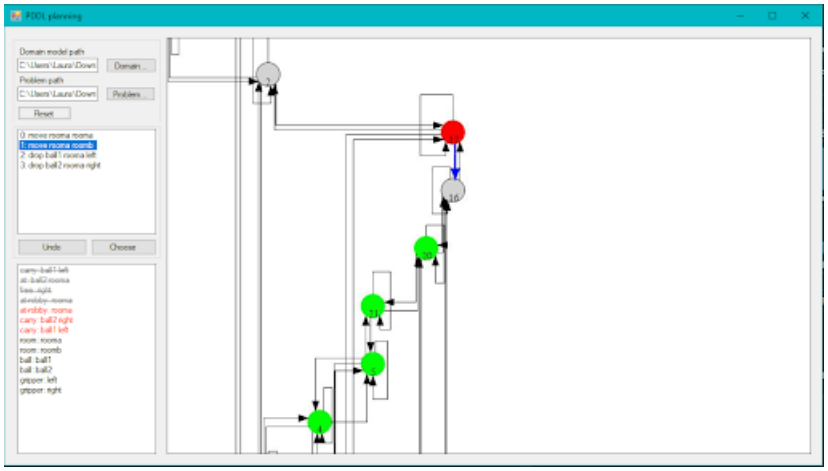

Figure 5: Screen Image 5

6) Selecting an action that will take us to the same state.

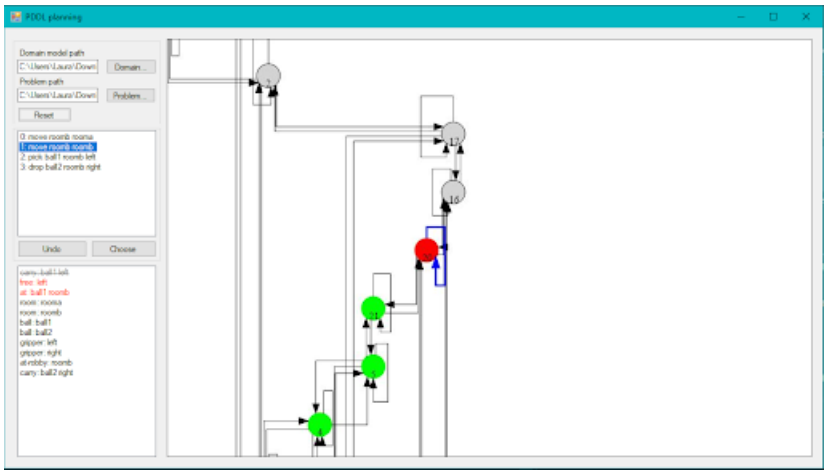

Figure 6: Screen Image 6

7) There are no new predicates listed (red ones) because the state remained, and so the predicates, remained the same.

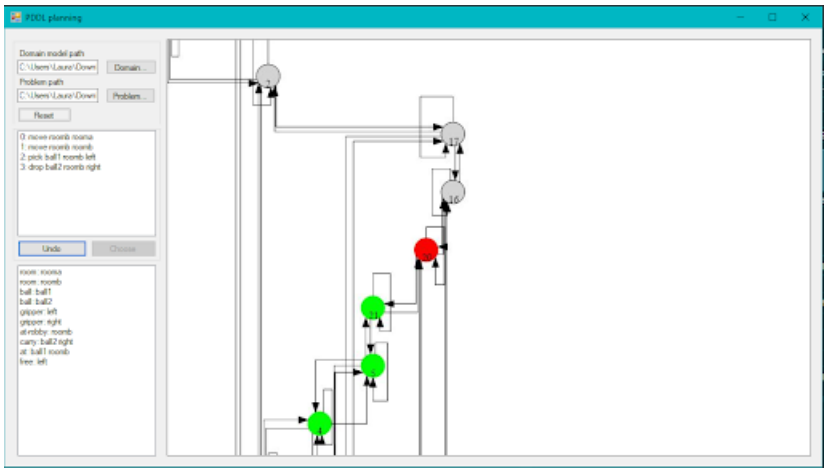

Figure 7: Screen Image 7

4.2 Requirement Compliance

The requirement compliance table is shown in the Table 3

Priority	Description	% Completed
High	**Importing PDDL** The user shall be able to import a textual PDDL domain- and problem description into the application.	100%
High	**Parsing domain- and problem definition** The system shall synthesize (and simulate) the two PDDL files and parse them for debugging purposes.	100%
High	**Visualizing the domain model** The user should be able to visualize the extracted data in textual and graphical form.	100%
Medium	**Simulating the domain model manually** The user can choose an action and the current state and the state tree shall be updated.	100%
Medium	**Returning to the previous state** The user can choose to go back to any previously visited state and continue from there.	100%
Medium	**Simulating the domain model automatically** The system can select arbitrary actions and simulate the domain model.	100%
Low	**Visualizing the states' tree** The user should be able to visualization of the whole tree of reached states.	100%

Table 3: Requirements

4.3 Acceptance testing

The acceptance test was conducted over a Skype meeting with the client. A visual demonstration of the tool was carried out during the meeting session. During the interactions with the client, various features of the software were tested and demonstrated in real-time with the client. In this way, the client could examine and give feedback on the progress and things that could be improved. This was done in order to make sure that the tool fulfils the minimal requirements as set by the client.
At the acceptance testing, the following feedback was collected from the client, as desired functionalities of the program.

- **Fluents**: Due to the scope of our project, numeric fluents introduced in PDDL 2.1 could not be included in the tool development. A reason for that is because we had to develop our own parser that covers the core functionalities of the PDDL language. The time was too limited for an extension of our own parser to cover numeric fluents. This could be solved using the VAL-tool which supports the numeric fluents and other features of the newer versions of the PDDL language.

- **Val-tool**: We prioritized an integration of the VAL-tool parser with our program. However, the lack of documentation of the VAL-tool made it difficult for us to continue with the integration. The information generated by the VAL-tool when parsing the PDDL files could not be extracted or interpreted by us, hence we started the development of our own parser.

- **Syntax checker and PDDL editor**: A syntax checker and editor was decided early in our planning to be left out of our development scope due to time limitation.

- **Negative preconditions**: General support for negative-preconditions in PDDL language is currently missing in the program. The need for negative preconditions in our software was identified late in the development, and as a result, it was not included due to time limitation.

Overall the client expressed satisfaction with the results of the development of the tool considering all the factors. The client moreover mentioned that the tool might be used in production after some modifications.

4.4 Functionality improvement areas

There is much that can be done to further improve our tool. One of the major points to extend is the parsing since our current solution only supports the most basic functionalities of PDDL. To do this we can either, broaden our own currently used parser or resume our research on the VAL-tool in attempt to integrate that parser with our program. The second option would be preferred since it allegedly already contains support for a large portion of the PDDL language as well as syntax checking.

Regarding the UI parts of the program some improvements could be made, the following are some examples:

- **Zoom functionality**:The graph can get quite big for complex domain- and problem definitions, therefore the ability to zoom in and out would help the user to get a better overview of the graph.
- **Clickable graph nodes**: If the user would like to see information about a specific node in the graph or perhaps move to that node using the shortest path, a more interactive graph where the user can click on the nodes is desired.

One thing that we would like to investigate further is to give the user the ability to remove states from the graph. This could be useful for cases where deadlock-states are occurring due to one or more faulty actions. When the user specifies which state to remove, the system would then show the user which other states will be affected by the change. By removing the state, the PDDL domain- and problem definitions should also be changed by removing the action causing the deadlock.
In addition, further improvements would be implementing optional functionalities such us a syntax checker and a PDDL editor.

4.5 Work products and Deliverables

Table 4 shows a general overview of the deliverables of the project, which are uploaded in GitHub. Three reports have been created to document the progress of the project, containing all the information required, which are the project plan, the detailed description and the final report. Three Powerpoint presentations have been presented in front of the other project groups of the Software Engineering course; which were related to the project plan, design and as well a final presentation for the completed work. Weekly Powerpoint presentations have been presented to the steering group for the updated development of the project. The final product is a tool which is going to be used in the future production processes of Ericsson.

Name	**Date planned**	**Date finished**
Project plan	Nov 22 2018	Nov 21 2018
Project plan (improved)		Nov 28 2018
1st Project presentation	Nov 28 2018	Nov 27 2018
Design Description (1st version)	Dec 6 2018	6 Dec 2018
Product(1st version)	Dec 6 2018	6 Dec 2018
2nd Project presentation	Dec 12 2018	Dec 11 2018
3rd Project presentation	Jan 16 2019	Jan 15 2019
Project plan (final latex version)		Jan 11 2019
Design Description (last version)	Jan 17 2019	Jan 11 2019
Product (last version)	Jan 17 2019	Jan 16 2019
Final project report	Jan 17 2019	Jan 16 2019

Table 4: Delivered files

5 Project Experiences

5.1 Experiences

Throughout all the work done for the project, each group member had a very positive experience and benefited from working with each other on a topic unknown for all of them. We got to learn what PDDL is and had a close-up approach with AI.
Working with a group of people that you didn't know before and managing to coordinate work and make compromises with each other were challenges that we all faced and overcame successfully.
After familiarizing with each other and presenting ourselves to all the group members, we divided the tasks and responsibilities for the project. Considering the fact that we still did not know each members' experience and background, the task was quite challenging. In the end, we can say that the way we compromised about dividing tasks in the start of the course was the most ideal one. As well as learning about AI and tools such as the VAL tool, we also learned about the methodology of working in the Software Engineering field using a platform like Trello, which proved to be very useful in organizing work in a detailed way and setting deadlines. In addition to this, each member learned other skills in the computer science field, in different areas of programming. We acquired knowledge about skills such as C# and LateX, a document preparation system for high-quality typesetting.
Something that we will all remember as the biggest challenge of this course was the difficulties with the client. As a crucial factor of a project like this, it proved to be the number one necessity that can, with its absence, lead to a slowdown of the development process due to a lot of questions not getting an answer.

5.2 Possibilities of improvement for future projects

Coming to the end of the project, we have gained a lot of experiences from the work. Most importantly, we will take with us possible improvements that each of us can benefit from when applying them in future teamwork projects.
The summarized collected suggestions from our team members were the following:

- **Communication**: Over the holidays the members of the group had difficulties in conducting meetings over the online platform "Discord". Response times were not ideal so showing each other the individual work done on the implementation of the tool was quite challenging. In the future, more fixed and mandatory meeting times can be set in anticipation, so that the meetings are attended by everyone and in time.

- **Ways of work**: The group members could have been more independent at the start by not relying too much on the client to respond to questions or issues that arose during the early development phase. In future work,

in these kinds of cases, an improvement would be to keep working with the development without dependency from the client.

- **Presentation skills**: The steering group meetings could have been used more efficiently by everyone to practice their public speaking skills, so to have more efficient and dynamic presentations.

References

[1] General overview of the scott project (secure connected trustable things): `https://scottproject.eu/general-overview/`.

[2] Mcdermott, Ghallab, Howe, Knoblock, Ram, Veloso, Weld, and Wilkins. *PDDL - the planning domain definition language. Technical report.* Yale Center for Computational Vision and Control, 1998.