Final Report

GROUP: Gamygdala-Integration:
B.L.L. Kreynen
M. Spanoghe
R.A.N. Starre
Yannick Verhoog
J.H. Wooning

June 18, 2015

Contents

1 Abstract

2 Introduction

In this section you can find a brief introduction to the context problem and to the solutions applied by group 3.

2.1 Problem description

The research problem is stated by a company called Tygron. They provide local authorities with a game in which their city is fully simulated. In this game the council members can discuss difficult matters on where to build certain structures. By doing so they gain an understanding of everybody's needs and responsibilities. The main question that rises is the possibility of replacing one of those players with an Artificial Intelligence solution. Since the players of this game mostly interact in an emotional way, as real humans do, these bots should feel emotions too.

This topic is very interesting because it has many applications. Both further research and different industries can profit from this. For this context project the students work in a large group that is subdivided into different smaller groups each with specific tasks. In total there are 4 groups of 5 students that work together. Together the whole group will make a proof of concept by creating a game in the Tygron engine[4], creating an interface between the Tygron engine and GOAL and creating both a plug-in and an integration of the Gamygdala emotional engine in GOAL. The specific task of this subgroup is to provide the group that will create an agent with an integration of Gamygdala in GOAL. It is not our task to implement the engine itself, since it is already present, but it is our job to integrate it in GOAL in a way that programmers, like the other groups, can make use of this when creating virtual humans.

In this report there is given a overview of the user requirements first. Then a list of the implemented software product is explained in detail. After that, you can find a reflection of the product and process from a software engineering perspective. This is followed by a detailed list of implemented features. A section on the Human Computer Interaction will then explain how the product and users interact. Finally a conclusion and outlook will list all the important findings and future improvements.

2.2 User requirements

In this section you can find the specific tasks the product should fulfill. By using the MoSCoW method we can subdivide the high level specifications into the following. The subsections decrease in importance or priority. This means that for example the MUST-haves are way more important to implement than the COULD-haves. This method is very handy and more information can be found in the glossary or references.

Must haves

- An agent in GOAL must have an equivalent in Gamygdala.

- The agent's goals in GOAL should also be present in Gamygdala for that agent. This means that if you define a goal for an agent in GOAL that the Gamygdala engine must also know of this. New goals must be communicated to Gamygdala.

- It must be possible to define how good or bad a belief is for certain goals.

- It must be possible to define relations between agents in GOAL.

- It must be possible to have these relations also present in Gamygdala. Again, it must be the case that when a relation is created in GOAL, the Gamygdala engine knows about this relation. Otherwise no emotions between two agents can be calculated.

- It must be possible to retrieve the emotional state of an agent in the same way as his beliefs.

- It must be possible to setup an initial emotion of an agent.

Should haves

- It should be possible to set the gain to a specific value.

- It should be possible to set the decay to a specific function.

- The Gamygdala goals and its properties should be possible to change. This also includes the relations regarding beliefs and goals.

- It should be possible to display the emotional states in the Simple IDE[5] provided with GOAL for debugging.

Could haves

- It could be possible to set a custom decay function.

- It could be possible to change relations during runtime.

- It could be possible for an agent to reason about other agents emotional bases.

Will not haves

- Eclipse plug-in

3 Interaction Design

This section will describe how we tried to validate the interaction of our users with our product. First there will be a description of the method used to gain feedback then the results from this will be discussed after which there will be an explanation of how we processed this feedback. Finally there will be a short discussion on what could be done to gain more valuable feedback.

3.1 Used Method

The idea used to gain the feedback from users on our product was to ask the other group that is programming the virtual human in GOAL to complete a tutorial. This is a perfect group to do this with since they will have to work with our program to complete their own, they have experience with GOAL and know a little bit about gamygdala so they're a perfect example of our typical end user. They were asked to think out loud so that it was possible to identify any issues they were having. The recording of this session is attached to this report.

Sadly enough only three people of the other group were available, they worked on the tutorial in one team. This felt like the best set up because that way they talk with each other about what they're doing.

3.2 Results

From the feedback gained it became obvious that some of our documentation was not yet clear enough. There were three identifiable issues with the documentation:

- Sometimes small details about how something worked were plain missing.

- Sometimes information was not put in a central place. The users were for example not finding some details of how to use the emotion configuration, what they were looking for was mentioned somewhere in the documentation but due to it not being in one central place it was difficult to find.

- Some things were still plain wrong since they had been changed after the documentation was made.

Furthermore we identified one bug in the program when the users were working on the tutorial.

3.3 Reaction

The things that were still missing in the documentation will be added to it and there will be updates to make sure that there are central parts of the documentation to find all relevant information about certain aspects of the program without having to go look at other parts of the documentation. We

will also make sure that the wrong documentation is updated to be compliant with the current version of our pogram. The bug that was identified has since been fixed.

3.4 Commentary

The feedback we've gained is somewhat superficial, it was certainly useful feedback but sadly enough the other teams did not have enough time to use the program for an extended period of time and then have an interview with us. Pretty much all of the received feedback was feedback on the documentation, we feel like if they would have had more time to use our program that we would have received more valuable feedback on missing features and the usability of our implemented features.

Only being able to do it with one group also makes the feedback more limited the product should be tested on some more groups to gain more diverse feedback.

4 Outlook

This section gives a recommendation as to what possible improvements there still are for future expansion of the project.

4.1 Emotion configuration parser

First off we define the properties of the emotion configuration in a regular text file that is parsed by a simple parser. It's not a bad idea to improve upon this and to create a new file with the ANTLR framework that is used for the other files in GOAL as well. Furthermore while our parser does throw errors which mention which line numbers are still incorrect it would be nice if this could be statically checked and displayed in the SimpleIDE or the GOAL plug-in for eclipse.

4.2 Emotion configuration

Secondly there are still a few things that could be added to the emotion configuration. It is possible to define common goals and individual goals and it is possible to define that the individual goals only apply to certain agents. However it is not possible to define a notion of teams for the common goals, if a common goal is defined and two agents adopt this goal then it is assumed that they are working together on this goal in some instances it might be useful to have an optional parameter that allows you to define multiple teams for these common goals. However before adding something like this it should also be considered whether this does not complicate the emotion configuration too much for a feature that might not be all that widely used.

4.3 Aspects of GOAL that influence emotions

There are still some aspects in GOAL that are not being taken into account for the evaluation of emotions but which could potentially be interesting. For example bots can send messages to each other and bots can also try to reason about the goals and beliefs that other agents are holding. These parts of GOAL have no effect on emotions in the current implementation. An example of how this could affect emotions is that needing to drop a GOAL because of a message given by another agents is less bad than having to drop a GOAL because of your own observations (the idea being that you were notified beforehand and had to waste less time trying to achieve this goal before realizing you couldn't complete it anymore). While we're not sure how this should affect emotions exactly it would be interesting to take a look at what could be done in these areas. Although, again, adding definitions for these things might make the emotion configuration overly complicated for a feature that might not have that big of an effect, this should be considered when thinking of these features. Either a smart way of setting a standard for these messages (so that programmers just have to send the correct message and not worry about any other configuration) or a very easy way of defining them in the emotion configuration should be figured out.

4.4 Causal agents

At the moment determining the causal agent of dropping or achieving a goal is fairly simplistic, for individual goals the causal agent is always the agent itself and for common goals it is the agents that first achieved that goal. This does not always reflect the real world and it might be interesting to see what can be done to improve this. Again just like with the other sections it should be carefully thought out so that it does not complicate the use of gamygdala within GOAL too much. For this an potentially interesting solution would be to define a new drop and insert predicate that not only takes the goal/belief to drop/insert as input but that also allows the programmer to enter which agent caused this drop/insert.

4.5 Code quality

Finally, in terms of code quality there can always be improvements of course. Some parts of the code would benefit from being re-factored a bit. Most of the code would also benefit from better integration testing, but this is not only a problem in our own code but also in the code of GOAL. In terms of unit testing the code written by our group scores pretty high but throughout the project we noticed a few times that some changes created serious issue in GOAL but none of our tests notified us of this. This was caused by a shortage of integration testing, all the individual components still seemed to work perfectly fine but when combined in certain scenarios they failed and these scenarios were not always tested. As mentioned, this would also be a recommendation to GOAL

itself, at one time for example a modification to updating the goal base caused one of our two agents to not perform any actions anymore at all, this was not caught by integration nor unit tests of GOAL.

References

[1] GOAL programming language `http://ii.tudelft.nl/trac/goal`

[2] Gamygdala emotion engine `http://ii.tudelft.nl/~joostb/gamygdala/index.html`

[3] MoScoW method `http://en.wikipedia.org/wiki/MoSCoW_method`

[4] Tygron engine `http://www.tygron.com`

[5] Simple IDE `https://github.com/goalhub/simpleIDE`