SCHULICH
School of Engineering

UNIVERSITY OF CALGARY

Department of Electrical and Computer Engineering

SENG 696 – Agent-based Software Engineering

PROJECT REPORT

Submitted by:
Suchina Parihar

BUG TRACKING SYSTEM WITH AGENTS

Suchina Parihar
Dept. of Electrical and Computer Engineering
University of Calgary
Calgary, Canada
suchina.parihar2@ucalgary.ca

ABSTRACT

This is the world of information. The ever-growing field Information Technology has its many advanced notable features which made it what it is now today. In this world, the information has to be processed, clearly distributed and must be efficiently reachable to the end users intended for that. Otherwise we know it lead to disastrous situations. The other coin of the same phase is it is absolutely necessary to know any bugs that are faced by the end users. The first part of the project "Bug Tracking System" aims to provide the solution for that. The Bug Tracker can be made from any two types. The first one being the system side, the other being the services side. Our project deals with the second one [1]. The second part of the Project is to help improve the Bug Tracking system by using Agents. The Project Manager maintains the master details regarding to the bugs id , bugs type, bugs description, bugs status, user details. The Project Manager too has the authority to update the master details of severity level, status level, etc. Agents help in the assignment of bugs and also in the interaction between the developers and Project Managers. The Bug Tracker can be further analyzed and further relevant and quick decisions can be taken.

1. INTRODUCTION

This is the world of information. Bug and issue tracking systems are often implemented as a part of integrated project management system. This approach allows including bug tracking and fixing in a general product development process, fixing bugs in several product versions, automatic generation of a product knowledge base and release notes [1]. The use of bug tracking systems as a tool to organize maintenance activities is widespread. The systems serve as a central repository for monitoring the progress of bug reports, requesting additional information from reporters, and discussing potential solutions for fixing the bug. Developers use the information provided in bug reports to

identify the cause of the defect, and narrow down plausible files that need fixing. A survey conducted amongst developers from the APACHE, ECLIPSE, and MOZILLA projects found out which information items are considered useful to help resolve bugs [3].
Previous research has shown that Project Managers often omit the importance of interaction and expertise while assigning bugs to developers. Developers are then forced to actively solicit information from web and this may stall development. The effect of this delay is that bugs take longer to be fixed and more and more unresolved bugs accumulate in the project's bug tracking system. We believe that one reason for this problem is that current bug tracking systems are merely interfaces to relational databases that store the reported bugs [4, 5]. The on-going work is directed towards rectifying the situation by proposing ideas that can fundamentally transform bug tracking systems to enhance their usability such that relevant information is easier to gather and reported by a majority of users [2]. Further, a simulation of an interactive bug tracking system is presented, using agents, which helps in interaction and assignment.

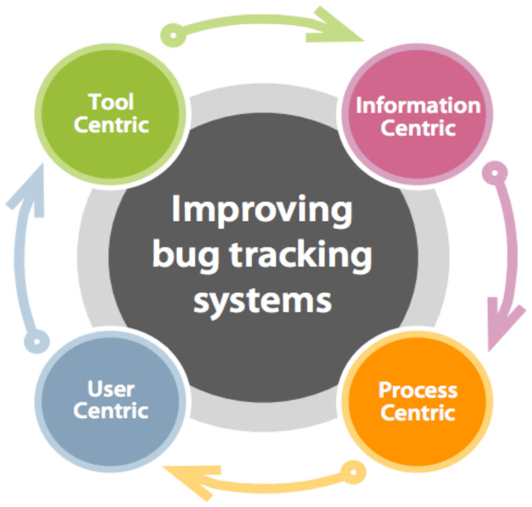


Fig.1. Improving Bug Tracking System

To improve the Bug Tracking System, I have used Agents to help interact with other employees in the system, thus making it easy and effective for the Project Managers to assign bugs to developers.

1.1 What is an Agent?

There is no strict definition about what an agent is. Literature offers a variety of definitions ranging from the simple to the lengthy and demanding ones. In [7], an agent is defined as follows,
"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors."

According to this definition an agent is any entity (physical or virtual one) that senses its environment and acting over it. Physical entities that could be considered as agents are, in the case of a power system, simple protection relay or any controller that controls directly particular power system component or part of the system. Virtual entity that can be considered as an agent is a piece of software that receives inputs from an environment and produces outputs that initiate acting over it. Often an agent is a combination of physical (computation architecture) and virtual one (a piece of software running on the computational architecture).
A more precise definition of the agent that is in fact extension of the definition mentioned above, is given in [8],

"Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are deigned."

The key extensions in this definition with respect to first one are words: computational, autonomy and goals. Word computational makes difference in agents that we are interested in engineering (computational agents) from biological agents (humans, animals, bacteria) since from first definition this distinction is not obvious. Autonomy means that computational agents operate without the direct intervention of some other entities and have some kind of control over their actions. Assigning the goals to the agent means that acting upon environment should be done in order to achieve some specified objective and that the agents expose a sort of rational (an agent that minimizes or maximizes its performance measure) behaviour in the environment. Behaviour here means the action that is performed after receiving sensory inputs [6].

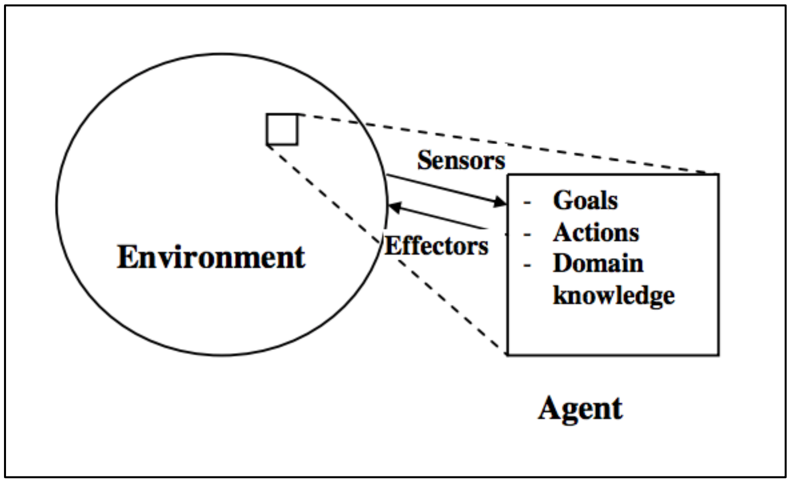


Fig.2. A general single-agent framework

1.2 MULTI AGENT SYSTEMS

A multi-agent system is a loosely coupled network of problem-solving entities (agents) that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity [9]. The fact that the agents within a MAS work together implies that a sort of cooperation among individual agents is to be involved. However, the concept of cooperation in MAS is at best unclear and at worst highly inconsistent, so that the terminology, possible classifications, etc., are even more problematic then in the case of agents what makes any attempt to present MAS a hard problem. A typology of cooperation from [10] seems the simplest and here we start with this typology as the basis for MAS classification. A MAS is independent if each individual agent pursues its own goals independently of the others. A MAS is discrete if it is independent, and if the goals of the agents bear no relation to one another. Discrete MAS involve no cooperation [4].

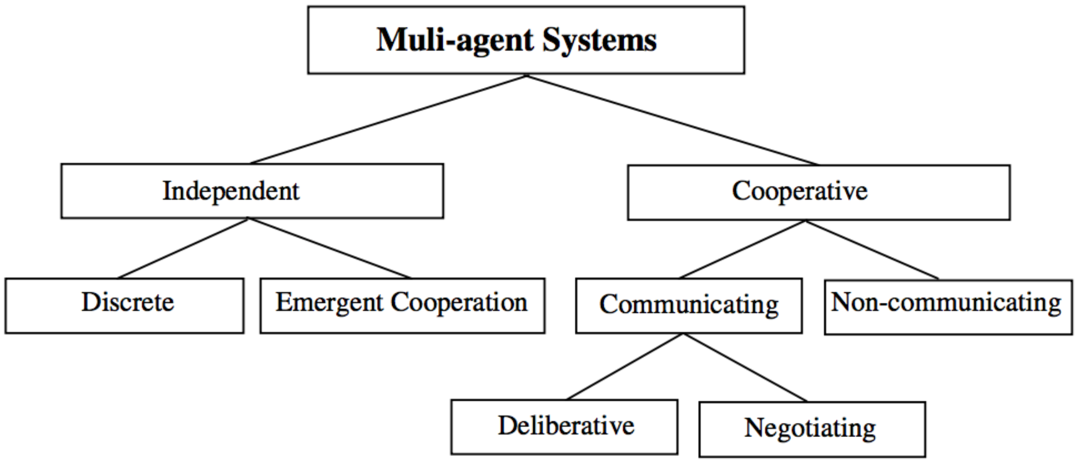


Fig.3. Cooperation typology

In Multi-Agent Systems (MAS), heterogeneous distributed services are represented as autonomous software agents which interact using an Agent Communication Language or ACL based on speech acts (Searle, 1969). There are several benefits to this particular approach: the ACL level of supporting computational interaction can be much “richer” and is at a higher level of abstraction (knowledge sharing level) than the classic representation of APIs in distributed object technology. This interaction can remove some of the common causes of communication errors in traditional systems which rely heavily on exact syntactical matching of service requests to service provider interfaces. Agents are designed as components, which are specifically designed to deal with unanticipated requests and they can spontaneously recruit the help of third parties when they need to. Agents are designed to operate autonomously, they can communicate using indirect asynchronous service requests rather than by direct, synchronous service invocation [6].

2. RELATED WORK

Many researchers have investigated what factors help in the quick resolution of bugs. Hooimeijer and Weimer [11] observed that bug reports with more comments get fixed sooner. They also noted that bug reports that are easier to read also have shorter life times. More recently, Aranda and Venolia have examined communication that takes place between developers outside the bug tracking system [12]. In [13], they discussed shortcomings of existing bug tracking systems, which led to the four areas of improvements presented in this paper.
In [15], they discussed that most agent platforms, including FIPA and non-FIPA platforms, naturally offer openness at the agent level in that although the platform itself may be fixed or closed, service and user agents can be dynamically added to the platform and agents themselves are naturally co-operative. This requires a common means of representing [encoding], understanding [ontology] and exchanging [protocol] service information. FIPA platforms defines a standard base protocol based on speech acts several standard ontologies: a core one for registering and querying de-registering services part of the FIPA management ontology] and various domain specific ones. No specific service encoding is mandatory. Non-FIPA platforms require the use of platform specific service ontology, encoding, and protocol combinations.

3. SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

The recommended minimum hardware specification for PC is:

- Pentium 166Mhz processor
- 64 MB Memory
- 4 MB disk space

3.2 SOFTWARE REQUIREMENTS

The recommended Software requirements for the Project are:

- JAVA 8
- JADE (Java Agent Development Framework) is a software Framework fully implemented in the Java language
- ECLIPSE IDE (Software Development Platform written in Java)
- SQLite Manager database

4. METHODOLOGY

4.1 FIPA

The Foundation for Intelligent Physical Agents, FIPA is a non-profit standards organization established in 1996 and registered in Geneva, Switzerland. Its purpose is to promote the development of specifications of generic agent technologies that maximize interoperability within and across agent based applications. Part of its function is to produce a specification for an agent enabling software framework. Contributors are free to produce their own implementations of this software framework as long as its construction and operation complies with the published FIPA specifications. In this way the individual software frameworks are interoperable [14].

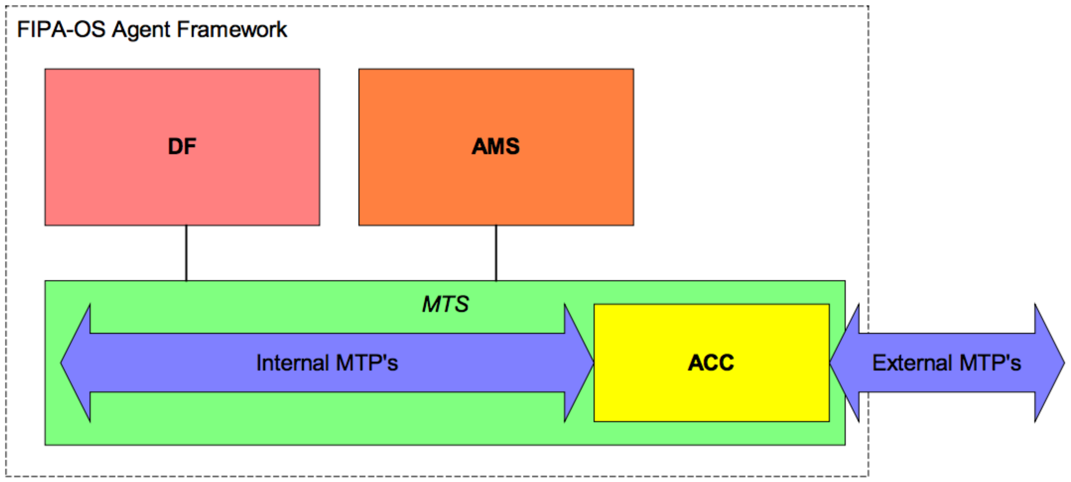


Fig.4. The FIPA agent reference model

The Directory Facilitator (DF), Agent Management System (AMS), Agent Communication Channel (ACC) and Internal Platform Message Transport (IPMT) form what are termed the Agent Platform (AP). The DF provides "yellow pages" services to other agents. The AMS provides white-page services and life-cycle management services for agents and the ACC supports inter-agent communication. The ACC supports interoperability both within and across different platforms. The Internal Platform Message Transport (IPMT) provides a message forwarding service for agents on a particular platform, which must be reliable. These are mandatory, normative components of the model. The ACC, AMS and DF are capability sets, they may be performed by one agent or by three different agents – this is left to the agent platform developer [14].

FIPA-OS is designed to support the FIPA agent standards. The FIPA reference model discussed earlier defines the core components of the FIPA-OS distribution: the Directory Facilitator (DF), Agent Management System (AMS), Agent Communication Channel (ACC) and the Internal Platform Message Transport (IPMT). In addition, there are different versions of the architecture available. In addition to the mandatory components of the FIPA Reference Model, the FIPA-OS distribution includes support for:

- Different types of Agent Shells for producing agents which can then communicate with each other using the FIPA-OS facilities;
- Multi-layered support for agent communication;
- Message and conversation management;

- Dynamic platform configuration to support multiple IPTMs, multiple types of persistence (enabling integration with legacy persistence software) and multiple encodings.
- Abstract interfaces and software design patterns
- Diagnostics and visualization tools

5. SYSTEM DESIGN

5.1 User

- Bug reporting is the main functionality of User where he is going to report the bugs to the System.
- Gets a notification when bug is fixed

5.2 Project Manager

- The Project Manager approve bugs that are submitted by the User
- The work of the Project Manager is to assign the specific bug to Developer.
- The Project Manager maintains Developer Information.
- Project Manager needs to Login into the System by entering Username and Password.
- Additional functionalities of Project Manager are:
  - Login
  - Browse Bug
  - Add Bug
  - Assign Bug
  - Remove Bug
  - Generate project report
  - Maintain Developer (add, remove or update)
  - Maintain Project (Add, remove or update)
  - Logout

5.2 Developer

- The Developer performs the different test methods on the given bugs and fix them.
- Developer reports a fixed bug
- Developer needs to Login to the system using Username And Password.
- Additional functionalities are:
  - Login
  - Browse Bug
  - Fix bug

- Report fixed Bug
- Generate bug report
- Logout

5.3 Discussion Agent

- The Discussion agent starts a discussion thread between Project Manager and Developers.
- Developer can ask for a specific bug according to their expertise
- Developer can ask questions from other developers
- Project Manager can assign bugs to developer through chat.

6. UML DIAGRAMS

6.1 STATE TRANSITION DIAGRAM OF A BUG

When a new bug is submitted the bug status will be NEW. Once the bug is created and stored in the data store the status will change to OPEN. The bug will be reviewed by the Project Manager to see if it needs to be tested by a developer. The bug status at this point will be TESTED. The Bug is assigned to a developer and the bug status now becomes ASSIGNED. The assigned developer will then mark the bug as FIXED after fixing the bug. After the developer reviews the fixed bug it is marked as CLOSED

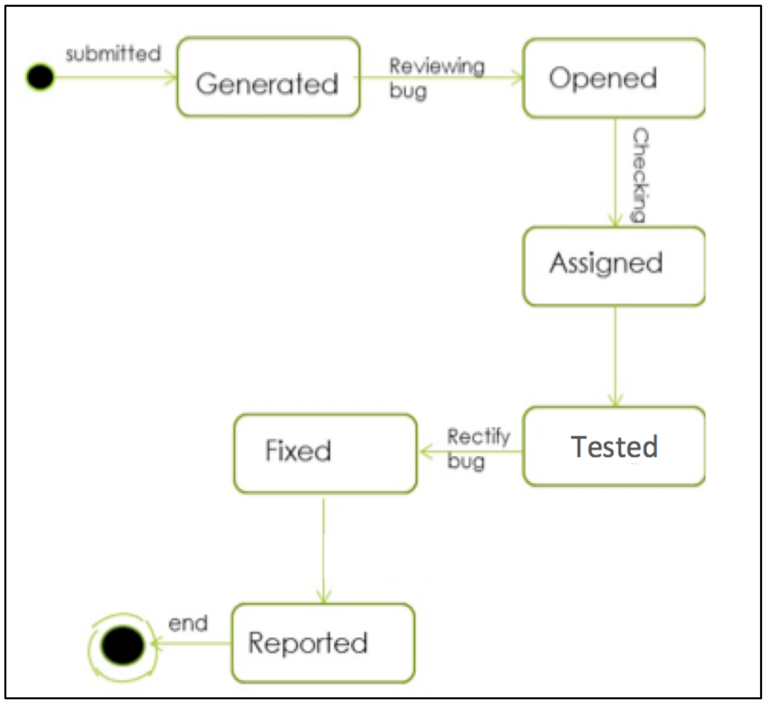


Fig.5. State Transition Diagram

6.2 CONTEXT DIAGRAM

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system, modelling its *process* aspects. A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored.

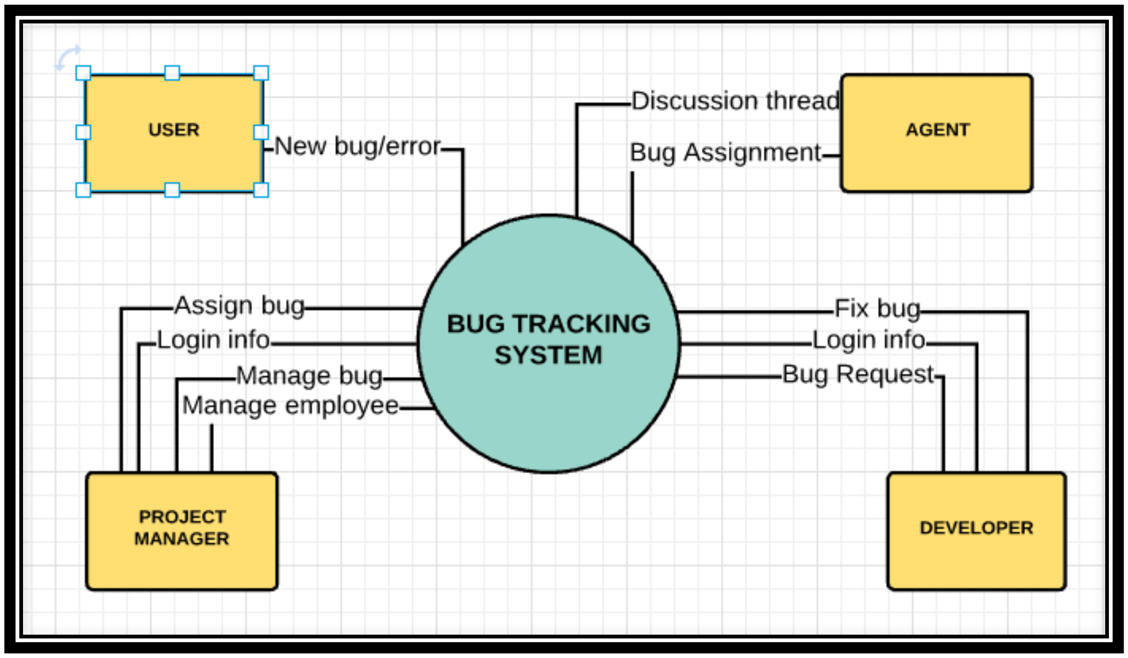


Fig.6. Level 0 DFD

6.3 USE CASE DIAGRAM

A use case model describes a system's functional requirements in terms of use cases. It is a model of the system's intended functions (use cases) and its environment (actors). It is used to communicate with the end users and Domain Experts. Use Case scenerios describe a few typical scenarios of how the system works. For example, what happens in the system while a user logs in, gives a short sequence of typical commands and logs out - what data flows from which module to where, what triggers which actions, etc. If the system is not very simple, this description can make the difference between utterly confused and fully comprehending readers.

The list of Use Cases and Actors used in this project are mentioned below:

Actors:

- Admin

- Project Manager
- Database
- User
- Developer
- BTS UI

Use cases:

- Report Bug
- Manage Employee
- Discussion
- Manage Bug
- Fix Bug
- Assign Bug
- Login

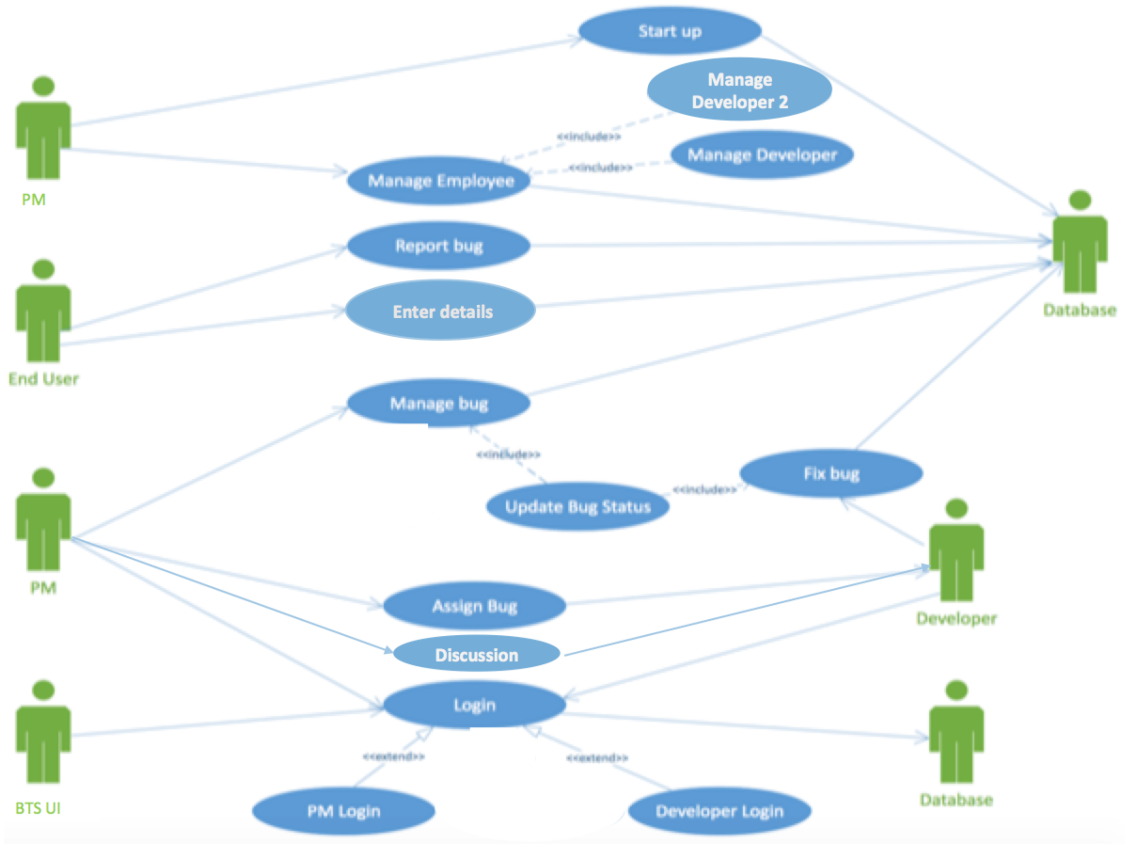


Fig.7. Use Case Diagram

6.4 CLASS DIAGRAM

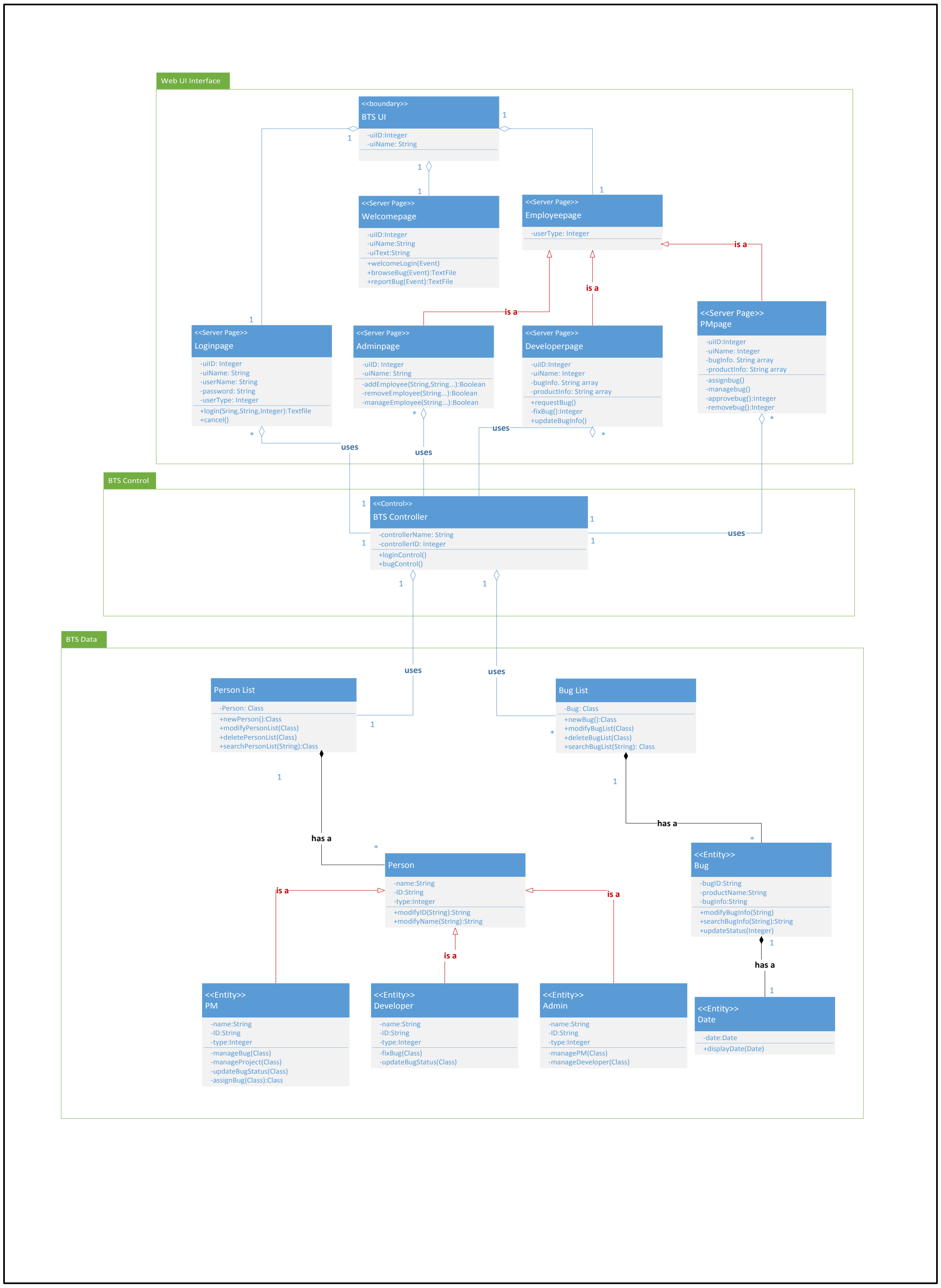
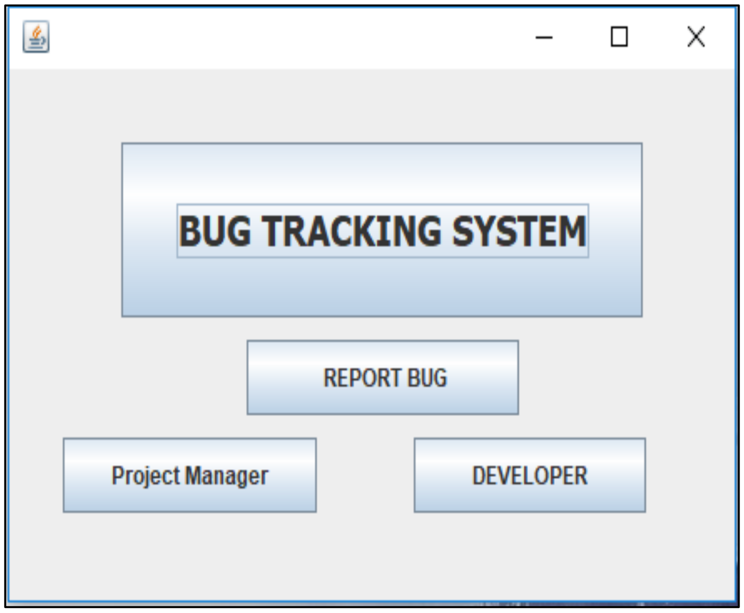


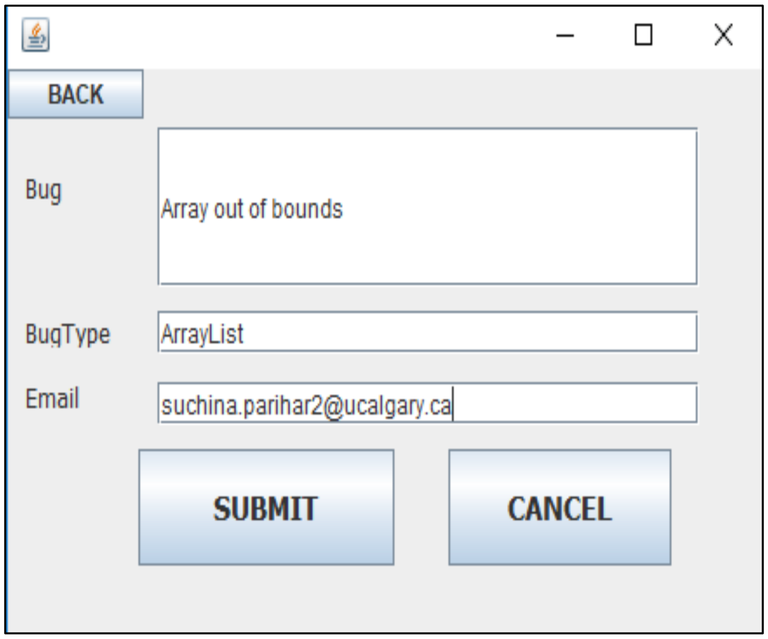
Fig. 8. Class Diagram

7. SCREENSHOTS

Step1: The GUI of the Project is given in the Screenshot 1. There are 3 different buttons for 3 different type of System users. When a **USER** wants to Report a bug, he/she will click the **REPORT BUG** button and will be directed to a page as shown in Screenshot 2.



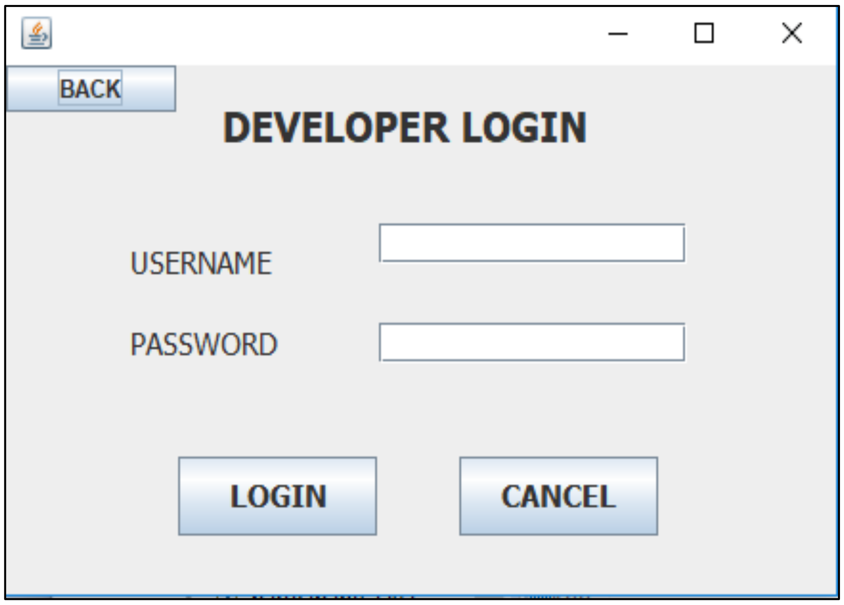
Screenshot 1



Bug: Array out of bounds
BugType: ArrayList
Email: suchina.parihar2@ucalgary.ca

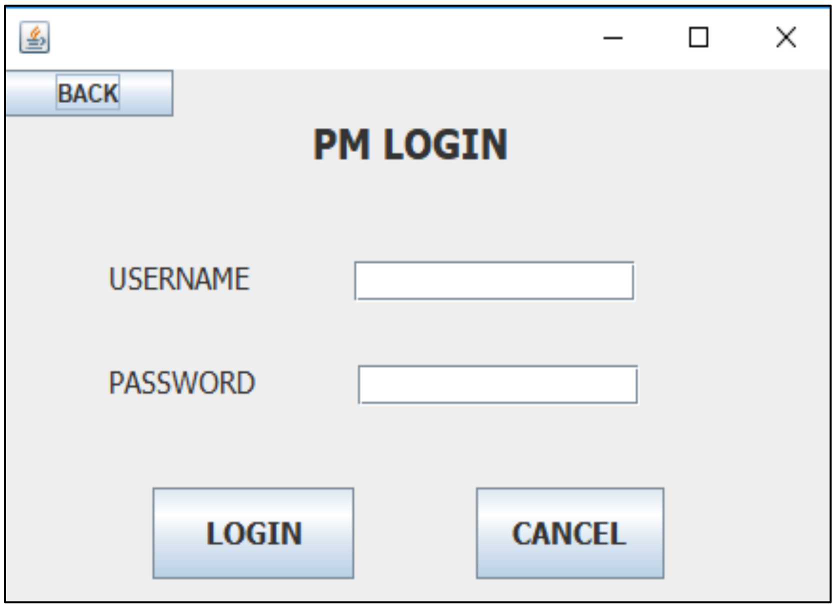
Screenshot 2

Step2: The Developers need to login to the system through login page as shown in Screenshot 3. After the login, they can view their assignments and can fix the bugs that are assigned to them.

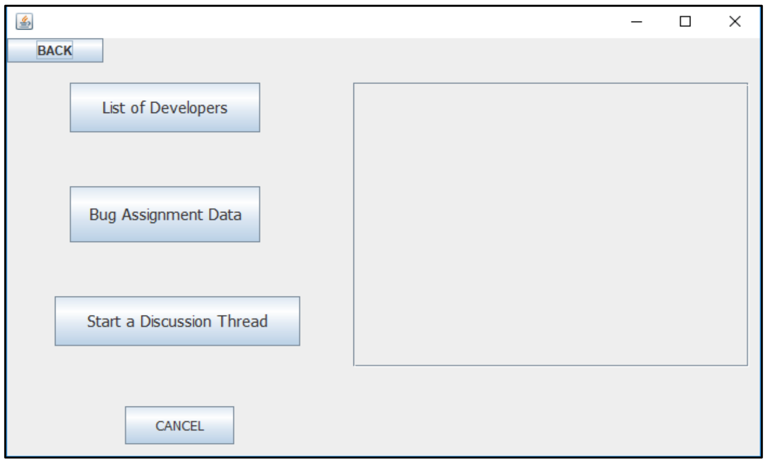


Screenshot 3

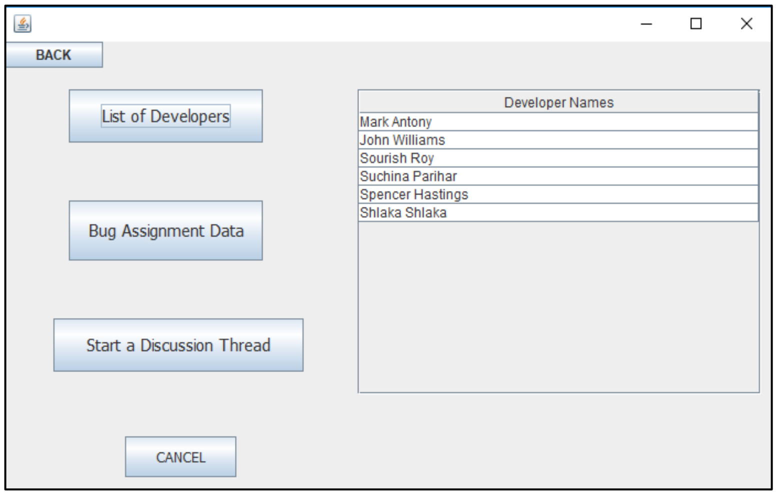
Step3: The Project manager can login into the system through **PM LOGIN** page as shown in Screenshot 4. PM is directed to a page where he/she can administer list of developers and the assigned bug data. The PM can also start a Discussion with other employees through discussion threads run by Agents



Screenshot 4

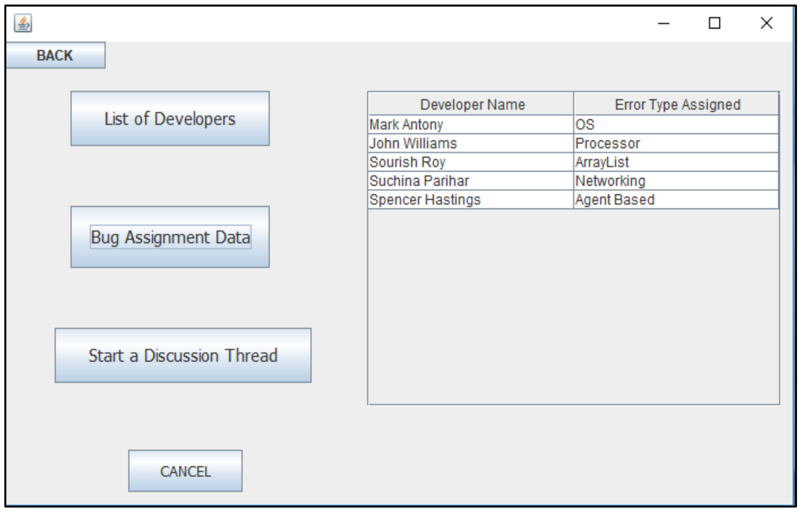


Screenshot 5



Developer Names
Mark Antony
John Williams
Sourish Roy
Suchina Parihar
Spencer Hastings
Shlaka Shlaka

Screenshot 6

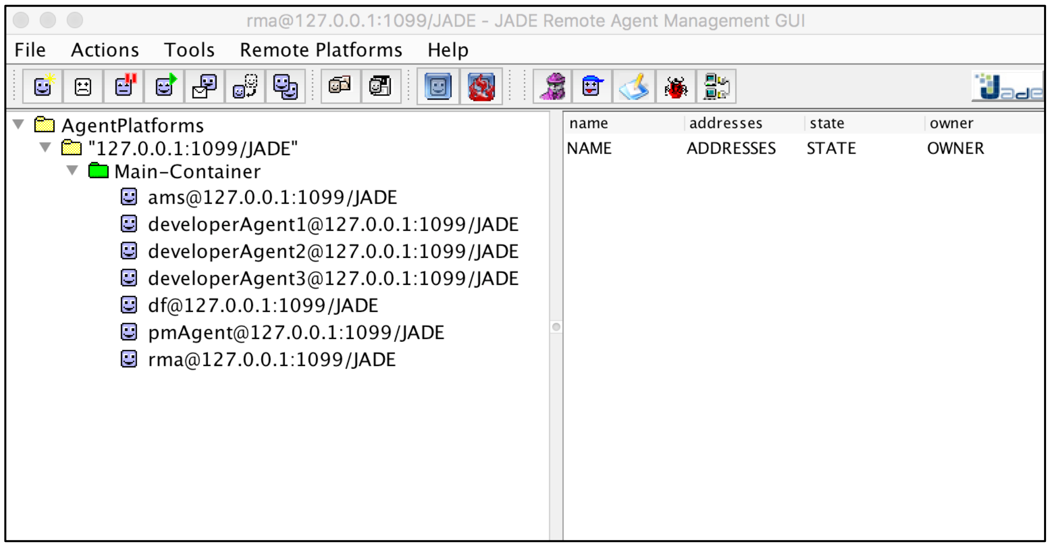


Developer Name	Error Type Assigned
Mark Antony	OS
John Williams	Processor
Sourish Roy	ArrayList
Suchina Parihar	Networking
Spencer Hastings	Agent Based

Screenshot 7

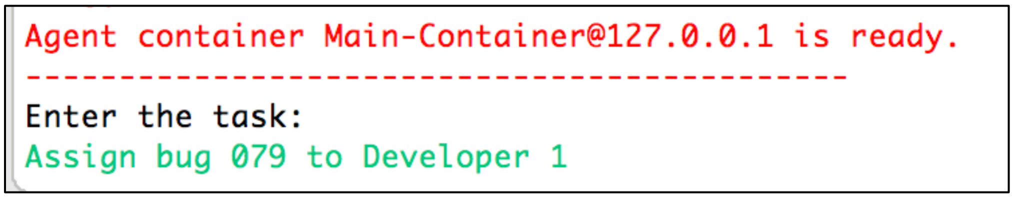
Step 4: Then comes the Agent module. In screenshot 8, the JADE platform is shown where there is a main container and 4 agents (PM agent, Developer 1, Developer 2, Developer 3). The message passing is represented in Screenshot10 where PM Agent assigns tasks to Developer Agents

.



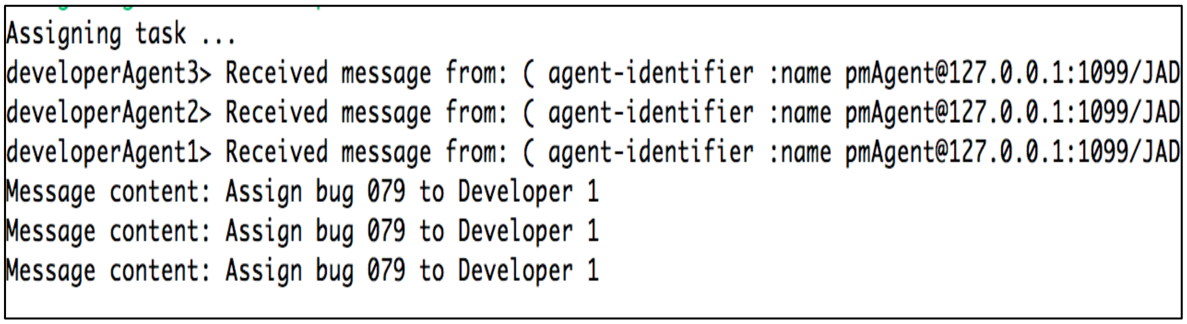
```
▼ AgentPlatforms
  ▼ "127.0.0.1:1099/JADE"
    ▼ Main-Container
        ams@127.0.0.1:1099/JADE
        developerAgent1@127.0.0.1:1099/JADE
        developerAgent2@127.0.0.1:1099/JADE
        developerAgent3@127.0.0.1:1099/JADE
        df@127.0.0.1:1099/JADE
        pmAgent@127.0.0.1:1099/JADE
        rma@127.0.0.1:1099/JADE
```

Screenshot 8



```
Agent container Main-Container@127.0.0.1 is ready.
--------------------------------------------------
Enter the task:
Assign bug 079 to Developer 1
```

Screenshot 9



```
Assigning task ...
developerAgent3> Received message from: ( agent-identifier :name pmAgent@127.0.0.1:1099/JAD
developerAgent2> Received message from: ( agent-identifier :name pmAgent@127.0.0.1:1099/JAD
developerAgent1> Received message from: ( agent-identifier :name pmAgent@127.0.0.1:1099/JAD
Message content: Assign bug 079 to Developer 1
Message content: Assign bug 079 to Developer 1
Message content: Assign bug 079 to Developer 1
```

Screenshot 10

8. FUTURE WORK

Further work is oriented along supporting future FIPA specifications, supporting new user and application requirements and enhancing user support [15]. There are many potential useful enhancements which could be made to the Project. Some of these in progress include supporting additional transport protocols, adding hooks for and providing user and management tools and supplying a richer suite of facilitators and sample agents. Large number of agents can help increase the scope of the Project. Moreover, I have implemented only the interaction done by different agents. In future, the assignment of bugs can be implemented with the help of Agents. Sniffer agent can also be used to capture all the interactions between the developers for security reasons as well. The use of agents differently, can divert the scope of the project in many different ways.

9. CONCLUSION

Recently, there has been a particular proliferation in the development of agent-based frameworks or software toolkits. Many of the agent toolkits provide support for multi-threading, communication and other basic interfaces such that the agent system designer can concentrate more on the modeling of the particular application domain. Therefore, these toolkits enable software developers to create agent based systems whilst removing some of the more tedious, complex tasks [15]. FIPA-OS represents an agent framework, which has been developed for use to construct heterogeneous, multi-agent platforms, agents, and services which adhere to the FIPA agent standards. In this Project, agents have helped in the interaction between different employees in a system. Agents are really helpful in passing messages to different employees. With the help of Jade, these messages can be easily sent to different agents in the system without any need to connect to server or local hosts in the network. Many different agent behaviors gives a wide domain of tasks that agents can perform.

10. REFERENCES

[1] Fiaz, A. S., N. Devi, and S. Aarthi. "Bug Tracking and Reporting System." *arXiv preprint arXiv:1309.1232* (2013).

[2] Zimmermann, Thomas, et al. "Improving bug tracking systems." *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*. IEEE, 2009.

[3] N. Bettenburg, S. Just, A. Schroter, C. Weiss, R. Premraj, and ¨ T. Zimmermann. What makes a good bug report? In FSE'08: Proceedings of the 16th International Symposium on Foundations of Software Engineering, pages 308–318, November 2008.

[4] N. Bettenburg, S. Just, A. Schroter, C. Weiss, R. Premraj, and ¨ T. Zimmermann. What makes a good bug report? In FSE'08: Proceedings of the 16th International Symposium on Foundations of Software Engineering, pages 308–318, November 2008

[5] S. Breu, J. Sillito, R. Premraj, and T. Zimmermann. Frequently asked questions in bug reports. Technical report, University of Calgary, March 2009.

[6] Glavic, Mevludin. "Agents and multi-agent systems: a short introduction for power engineers." (2006).

[7] S. Russel, P. Norvig, "Artificial intelligence – A modern approach", Prentice Hall, 1995

[8] P. Maess, "Artificial life meets entertainment: Life like autonomous agents", Communications of the ACM, vol. 38, no. 11, pp. 108-114, 1995.

[9] P. Stone, M. Veloso, " Multiagent systems: A survey from a machine learning perspective", Autonomous Robots, vol. 8, no. 3, pp. 345–383, 2000.

[10] J. Doran, S. Franklin, N. R. Jenkins, T. J. Norman, " On cooperation in multi-agent systems", In UK Workshop on Foundations of Multi-agent Systems, Warwick, 1996.

[11] P. Hooimeijer and W. Weimer. Modeling bug report quality. In ASE'07: Proceedings of the 22nd International Conference on Automated Software Engineering, pages 34–43, 2007.

[12] J. Aranda and G. Venolia. The secret life of bugs: Going past the errors and omissions in software repositories. In ICSE'09: Proceedings of the 31st International Conference on Software Engineering (to appear), 2009.

[13] S. Just, R. Premraj, and T. Zimmermann. Towards the next generation of bug tracking systems. In VL/HCC'08: Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing, pages 82–85, September 2008.

[14] O'Brien, Paul D., and Richard C. Nicol. "FIPA—towards a standard for software agents." BT Technology Journal 16.3 (1998): 51-59.

[15] Poslad, Stefan, Phil Buckle, and Rob Hadingham. "The FIPA-OS agent platform: Open source for open standards." *proceedings of the 5th international conference and exhibition on the practical application of intelligent agents and multi-agents*. Vol. 355. 2000.