Rafael H. Bordini, Louise A. Dennis and Yves Lespérance (eds)
Pre-proceedings of the 7th International Workshop on Engineering Multi-Agent Systems (EMAS 2019)
13th-14th May 2019, Montreal, Canada
`cgi.csc.liv.ac.uk/~lad/emas2019/accepted/EMAS2019_paper_8.pdf`

Jacamo-web is on the fly: an interactive Multi-Agent System IDE*

Cleber Jorge Amaral[1,2] and
Jomi Fred Hübner[1]

Federal University of Santa Catarina (UFSC), Florianópolis, SC, Brazil
`jomi.hubner@ufsc.br`
Federal Institute of Santa Catarina (IFSC), São José, SC, Brazil
`cleber.amaral@ifsc.edu.br`

Abstract. This paper presents *jacamo-web*, an interactive programming IDE to develop Multi-Agent Systems. The standard programming method usually follows the sequence of compile, link and execute the application. Alternatively, so-called interactive programming provides a way to modify a system while it is running. Besides saving developing time, it maintains the system's availability since the application is kept running while it is being changed. To illustrate *jacamo-web* interactive functions, we have developed a MAS for the financial market. It checks data of companies and applies known formulae to suggest whether to buy assets or not.

Keywords: Interactive programming · Just-in-time programming · Multi-Agent Oriented Programming · On-the-fly programming.

1 Introduction

Interactive programming is a form to develop a program while it is running, without stopping or restarting, acting directly over its instance [6]. It allows rapid prototyping, debugging and learning, as well as facilities for incremental development. On the interactive approach, the programmer can enter a program or a fragment directly into an already running system, reducing system development time since the usual compile-link-execute process is done in a single step [5]. This method is also useful in cases where there is no clear specification of the problem at design phase and adaptations are required at run-time. This feature is particularly useful for systems that perform critical missions needing high availability.

For long term running systems, such as some open systems, interactive programming gives tools to improve partly or the whole system at run-time. For example, imagine stock market autonomous agents buying and selling all sorts of assets. It is usual that one needs to enhance some functions of the agents, for instance, their prediction models and decision-making rules. Interactive programming allows to instantly apply such changes.

* Supported by Petrobras project AG-BR, IFSC and UFSC.

This paper presents *jacamo-web* an Integrated Development Environment (IDE)[1] which uses the concept of interactive programming for development of Multi-Agent Systems. It extends JaCaMo platform [3] adding facilities to create, modify and destroy agents, artifacts, and organisations at running time. This IDE is showed by demonstrating an application of financial market consultants.

2 Jacamo-web

JaCaMo is a Multi-Agent Oriented Programming (MAOP) platform that splits programming concerns of a MAS by the parts responsible for autonomous decisions: (i) the agents which are developed in Jason; (ii) their shared environment, programmed in CArtAgO, a java-based framework; and, (iii) the coordination of global behaviour, which is developed in MOISE, that uses artifacts to represent organisational entities [3]. *Jacamo-web* adds a web interface allowing *users* to create, modify, interact with, and destroy agents, artifacts, and organisations. Although we have MAS IDEs where the agents themselves can modify the running system (by dynamically adding plans, changing beliefs of others, changing its organisation and environment), *jacamo-web* brings this feature for the (developer) user using a web interface.

Jacamo-web provides interactive functions of Read-Eval-Print Loop (REPL). The acronym REPL refers to: Read user insertions, Evaluates them, Print the result for the user, all of this, repeatedly in a Loop [7]. This technology allows the user to send commands to agents, to insert new instructions or full blocks of code. Jason's API is equipped with REPL functions which are processed by Jason's internal interpreter.

In case of environmental artifacts, *jacamo-web* brings a built-in Java compiler. It allows the development of new artifacts by coding java files which are compiled automatically. These new or changed artifacts can be used in the running system.

In the case of the organisation, *jacamo-web* allows the user to create new organisations and change those that are already running. For instance, the user can create, modify and remove roles, shared goals, coordination schemes, and norms.

3 Demonstration

The facilities provided by *jacamo-web* for developers are demonstrated by the implementation of a MAS for the financial market. The organisation of the MAS has two roles: consultants, which read assets data to apply a particular formulae to suggest whether to buy it or not; and, assistants, which receive users requests, asking to consultants their opinion, compiling a final suggestion and replying to the user. The interface with final users is implemented using the Telegram cell phone application, which is being integrated through Apache Camel framework [1]. Fig. 1 shows the architecture of the application and *jacamo-web*.

[1] A demo application is running at http://191.36.8.42:8080/

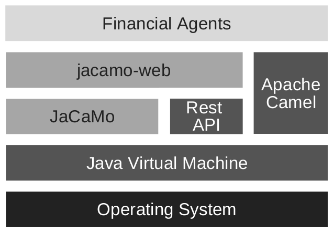


Fig. 1. Financial agents and *jacamo-web* architecture

In the financial market, there are some known investors that have shared the way they decide to buy an asset or not. For this demo, we adapt *Benjamin Grahan*'s, *Decio Bazin*'s and *Joel Greenblatt*'s formulas [2, 4][2]. Each of these decision rules is coded into agents with the same name as the original authors' formulas. These agents are connected to an artifact that gets financial data of assets from an external web-site. The assistant agent sends to the user consultant opinions as well as a summarised recommendation. The final recommendation is to *buy* the asset if at least two of the consultants are suggesting to buy.

Fig. 2 shows a diagram generated by *jacamo-web* according to the current system's state. The agents are represented by round shapes. The roles they are playing are represented by connections with the organisation *finantialteam*. The missions they are committed to are represented by connections with the organisational scheme *finantialsch*. The *consultants* are also connected with the artifact *fundamentus* which contains a parser for consulting asset data.

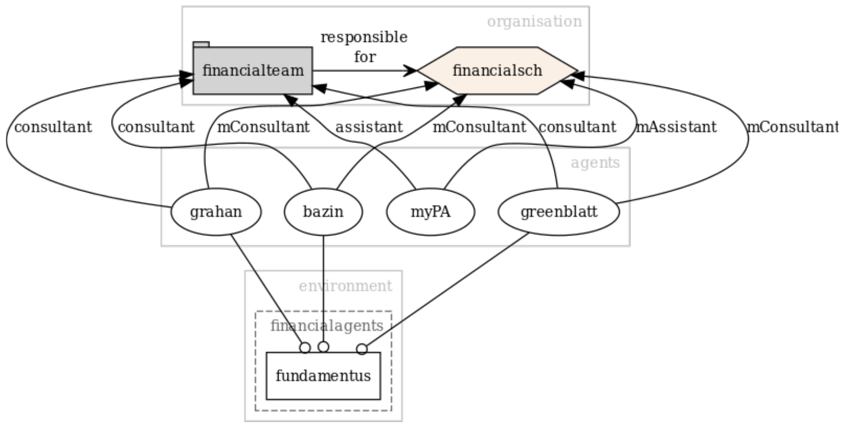
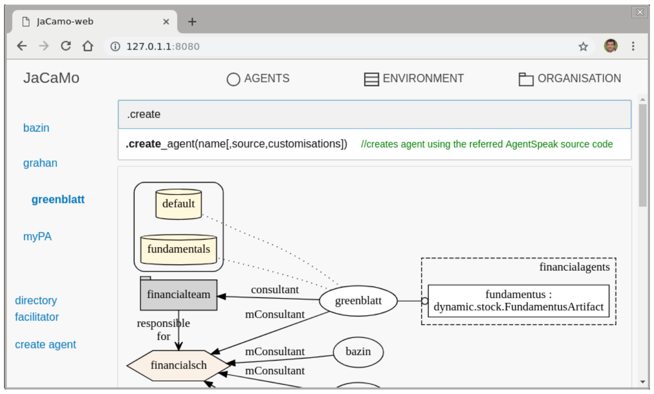


Fig. 2. Application overview showing runtime organisation, agents and environment

[2] Buying conditions: Graham: $Price < \sqrt{(22.5 * EPS * BVPS)}$; Greenblatt: $EBIT/(MarketCap + NetDebt) < 0.1$ and $ROIC < 0.1$; Bazin: $DY >= 0.06$ and $Debt/EV <= 1$.

Jacamo-web allows programmers to exploit the following features:

- Inspect the current state of the agent's belief base, plan library and relations.
- Change the agent's belief base without stopping the agent.
- Modify the agent's plan library while running.
- Create a new agent by command box (Fig. 3) and by menu.
- Kill an agent and recreate him.
- Send a new plan to an agent using *tellHow* performative.
- Consult directory facilitator.
- Inspect workspaces and artifacts seeing agents which are focusing on them.
- Create an artifact based on a new java file.
- Modify an artifact, dispose and create an new instance.
- Inspect organisations.
- Change organisation's structure and schemes.



```
.create
.create_agent(name[,source,customisations])   //creates agent using the referred AgentSpeak source code
```

Fig. 3. Text box with code completion. Diagram of agent greenblatt's relations.

4 Conclusions

We have presented *jacamo-web*; an extension of JaCaMo MAS platform. *Jacamo-web* has shown that it can shorten project life cycle. We could take advantage of instantiated contexts and quickly get responses from new code insertions. While developing the financial application, we have faced common situations that needed changes on agents, environment and organisation that could be applied at running time, and the results have been shown instantly. In case of open systems, they are supposed to be available for new entrants where IDE like *jacamo-web* are useful to help maintain the system's availability. In addition, we think it also facilitate to understand programming aspects, being an important tool for didactic purposes. As far as we know, *jacamo-web* is the first interactive MAOP IDE where the user can interact with the system while it is running.

References

1. Amaral, C., Cranefield, S., Roloff, M.L.: Development of a Multi-Agent System in the Industry 4.0 Context - Using JaCaMo and Apache Camel. In: Anais do IX Congresso Brasileiro de Engenharia de Fabricação. ABCM (2017). https://doi.org/10.26678/ABCM.COBEF2017.COF2017-0027
2. Bazin, D.: Faça Fortuna com Ações, Antes que seja Tarde. CLA Cultural, 6a edn. (2006)
3. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A.: Dimensions in programming multi-agent systems. The Knowledge Engineering Review **34**, e2 (2019). https://doi.org/10.1017/S026988891800005X
4. Reese, J., Forehand, J.: The Guru Investor: How to Beat the Market Using History's Best Investment Strategies. Wiley (2009), https://books.google.es/books?id=Z7J_sg9iX5IC
5. Tung, S.H.S.: Interactive modular programming in Scheme. ACM SIGPLAN Lisp Pointers **V**(1), 86–95 (1992). https://doi.org/10.1145/141478.141512
6. Wang, G., Cook, P.R.: On-the-fly Programming: Using Code as an Expressive Musical Instrument. NIME '04 Proceedings of the 2004 conference on New interfaces for musical Expression pp. 138–143 (2004). https://doi.org/http://dx.doi.org/10.1017/S1092852916000900
7. Wenzel, M.: READ-EVAL-PRINT in Parallel and Asynchronous Proof-checking. Electronic Proceedings in Theoretical Computer Science **118**, 57–71 (2013). https://doi.org/10.4204/EPTCS.118.4, http://arxiv.org/abs/1307.1944v1