A Framework for Collaborative and Interactive Agent-oriented Developer Operations*

Demonstration

Cleber Jorge Amaral
Federal Institute of Santa Catarina
São José, SC, Brazil
cleber.amaral@ifsc.edu.br

Timotheus Kampik
Umeå University
Umeå, Sweden
tkampik@cs.umu.se

Stephen Cranefield
University of Otago
Dunedin, New Zealand
stephen.cranefield@otago.ac.nz

ABSTRACT

Considering the increasing prevalence of autonomous systems in today's society, one could expect that agent-oriented programming (AOP) is gaining traction among mainstream software engineering practitioners. However, the tools and frameworks that are used and developed in the academic multi-agent systems engineering community struggle to keep up with recent developments in the software industry in regards to how complex information systems are developed and maintained. An important aspect of recent changes in software engineering practices is the application of technologies that support the increasingly fast iteration of a programming-testing-deployment cycle. Such approaches require intense collaboration that crosses boundaries between traditionally separated roles like software development, quality assurance, and operations; these approaches are often referred to as *DevOps*. In this paper, we work towards the integration of DevOps and AOP by introducing an extension of *jacamo-web*, an Integrated Development Environment (IDE) that supports the collaborative, web-based development and real-time continuous integration of autonomous agents and Multi-Agent Systems (MAS).

KEYWORDS

Agent-oriented programming; IDE; Iterative software development; Engineering multi-agent systems

ACM Reference Format:
Cleber Jorge Amaral, Timotheus Kampik, and Stephen Cranefield. 2020. A Framework for Collaborative and Interactive Agent-oriented Developer Operations. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), Auckland, New Zealand, May 2020,* IFAAMAS, 4 pages.

1 INTRODUCTION

In many organizations, the software development process from specification to deployment is executed at an increasingly rapid pace; often, software teams aim at improving their processes to support *continuous deployments*, *i.e.* the ability to deploy small updates to the code base without system downtime in a fully automated manner and at any point in time [9, 11]. Consequently, hand-overs between distinct technology ecosystems and teams become increasingly impractical [10]. To address this challenge, the concept of *Developer Operations* (DevOps) has emerged as a widely employed best practice for engineering and maintaining complex information systems [7]. DevOps comes with its own set of tools, which range from business process execution engines that bridge the gap between business and IT [6] to container orchestration tools that allow for the automation of deployments and operations of system instances across a heterogeneous and scalable device and infrastructure landscape [4].

Interactive programming refers to the programming of a running system in real time [14]. Because changes to the program code are applied to the running system instance(s) on-the-fly as soon as they are made or committed, interactive programming speeds up development time [12]. Consequently, interactive programming facilitates prototyping and debugging, as well as iterative development, especially in scenarios, in which no precise design documents exist [5, 12].

Among software engineering practitioners, it is largely acknowledged that programming is an inherently collaborative activity. On the technology side, frameworks and tool-chains emerged in which collaboration features are treated as first-class citizens. Most notably, version control systems and abstractions on top of them (for example: GitHub and GitLab) [2] are facilitating collaborative work in large teams, integration of specification documents (*issues*) and source code, and continuous integration. To take collaboration support to a new level, IDEs with (near) real-time collaboration features have been developed during the last decade [8, 13]; yet, in contrast to collaborative version control systems, such IDEs have so far not been widely adopted in practice. To the best of our knowledge, no collaborative programming approaches and technology frameworks exist specifically for AOP.

2 REAL TIME COLLABORATIVE AOP

We extended *jacamo-web* [1], a JaCaMo-based [3] IDE for developing Multi-Agent Systems under the perspective of the dimensions of the agents, environment and organizations[1]. *Jacamo-web* makes use of Read-Eval-Print Loop (REPL) [15] functions, allowing the engineer to interactively insert blocks of code into running agents. These on-the-fly updates keep the agent's context and current intentions when changes to its plans library are made. In addition, *Jacamo-web* supports on-the-fly development of organizations and

*Work partially supported by CAPES, project PrInt CAPES-UFSC "Automation 4.0" and by the Wallenberg AI, Autonomous Systems and Software Program (WASP).

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 2020, Auckland, New Zealand.

[1] A demo instance runs at http://jacamo-web.herokuapp.com/.

artifacts thanks to a built-in Java compiler. The extension we demonstrate adds integrative and real-time collaborative features for AOP to *jacamo-web*.

Let us introduce a running example to demonstrate the features. We implement an online business-to-business marketplace for autonomous supply chain integration. The marketplace is modeled as an MAS, composed of three organizations: *i)* a factory that needs to buy items; *ii)* a supplier that wants to sell items; and *iii)* the marketplace, representing the institution in which agents can adopt *buyer* and *seller* roles and trade in compliance with its norms.

We have two engineers developing this system; one is responsible for the factory side and the other for the supplier. The engineers have to develop their agents according to the marketplace protocol, which specifies three steps to be performed in order to fully process a trade transaction: (i) an item must be ordered; (ii) the payment must be made; and (iii) the item must be delivered.

In our example, both factory and supplier organizations have the common objective of trading with each other. The engineers are free to decide the details of how the objective can be achieved, which illustrates a common challenge that modern software engineering teams face.

Let us highlight the importance of having interactive facilities such as reactive functions to inform an engineer of changes done by others. It can be achieved when the IDE updates its content automatically. For instance, when the engineer creates and updates the *seller* agent, the other engineer that is programming the *buyer* may perceive and interact with the current state of the *seller* immediately. The interactive development approach is often adopted when specifications are unclear. For effective, rapid small upgrades, it is essential to have a hot-swap facility providing an immediate response from the system. It helps engineers to understand the system's behavior and effects of each change made in MAS models.

Furthermore, continuous deployment requires fast quality assurance functionalities. *Jacamo-web* provides facilities such as code highlighting and code completion that increase efficiency and help prevent mistakes, but comprehensive test measures are needed for updates. Our extension performs tests on temporary running instances. This allows the framework to check compatibility using the real scenario.

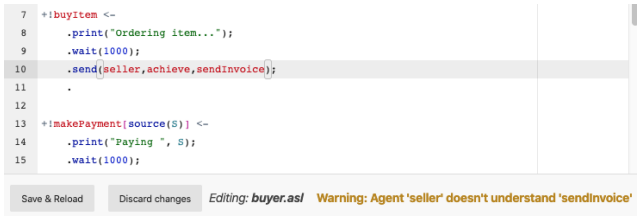

Figure 1: *Understandability* check warns that the target agent has no plans to achieve the goal *sendInvoice*.

Collaborative facilities are critical factors for the proper development of systems with high integration requirements. Our extension introduces two functionalities to test agents' integration: understandability and usefulness checks. In our marketplace, let us say the *buyer*'s engineer adds a request to send to the *seller* agent to achieve the goal *sendInvoice*. In case the recipient does not have any plan to be triggered to achieve such goal, the engineer of the *buyer* agent is notified by an understandability warning (Fig. 1). Using the same idea, consider that the *buyer* agent intends to send a belief to the agent *seller* informing that the payment was made. Beliefs can trigger events and can also be used in decision-making processes. Let us say, due to a typo, the agent is wrongly sending *paidd* instead of *paid*. As the usefulness checking function does not find any mention of *paidd* in the recipient's plans library, it reports a warning helping the engineer to prevent that error. In both checking functions, tests are performed while the engineer types (in real-time) without affecting running instances.

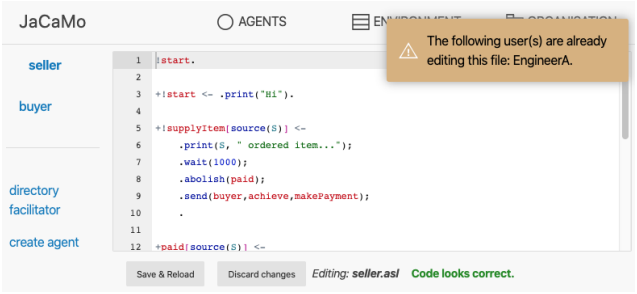

Figure 2: Notification of concurrent use to prevent data loss.

In addition, our extension provides integrative facilities for preventing conflicts when developers attempt to edit a resource simultaneously, as shown in Fig. 2, and for managing versions using the version control system git [2], as illustrated in Fig. 3. This facilitates collaboration, as well as the integration of different MAS instances, for example to provide separation between a sand-boxed development instance and the production environment.

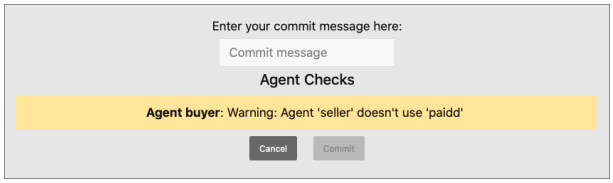

Figure 3: The commit dialog highlights potential issues.

During the development of our marketplace, we can see the importance of monitoring tools on several levels and from different perspectives. *Jacamo-web* provides facilities for observing the system using fragmented views, i.e., from the perspective of each agent. It is also possible to inspect the current state of an agent's mind and the current value of an observable property of an artifact or the state of an organization. Our extension adds live observability, *i.e.*, the interface instantly updates the observable state of agents, artifacts and organizations.

3 CONCLUSION

We have presented an extension of *jacamo-web* that combines its original interactive programming approach with collaborative and

integrative tools to promote fast and continuous deployment. Our marketplace scenario highlights the importance of such facilities in the development of MAS. To further advance the integration of AOP and collaborative software engineering and *DevOps* approaches, we suggest to *i)* transfer ideas of AOP into existing, production-scale DevOps tooling, *ii)* further develop *Agent DevOps* as a software engineering paradigm, and *iii)* continue with the implementation of collaborative DevOps features in AOP frameworks.

REFERENCES

[1] Cleber Jorge Amaral and Jomi Fred Hübner. 2019. Jacamo-web is on the fly: an interactive Multi-Agent System IDE. In *7th International Workshop on Engineering Multi-Agent Systems (EMAS 2019)*. https://cgi.csc.liv.ac.uk/~lad/emas2019/accepted/EMAS2019_paper_8.pdf

[2] John D. Blischak, Emily R. Davenport, and Greg Wilson. 2016. A Quick Introduction to Version Control with Git and GitHub. *PLOS Computational Biology* 12, 1 (2016), 1–18. https://doi.org/10.1371/journal.pcbi.1004668

[3] Olivier Boissier, Rafael H. Bordini, Jomi F. Hübner, and Alessandro Ricci. 2019. Dimensions in programming multi-agent systems. *The Knowledge Engineering Review* 34 (2019), e2. https://doi.org/10.1017/S026988891800005X

[4] Eric A. Brewer. 2015. Kubernetes and the Path to Cloud Native. In *Proceedings of the Sixth ACM Symposium on Cloud Computing (SoCC '15)*. Association for Computing Machinery, New York, NY, USA, 1. https://doi.org/10.1145/2806777.2809955

[5] William Choi. 2008. Rehearse: Coding Interactively while Prototyping, In Proceedings of the 21th Annual ACM Symposium on User Interface Software and Technology. *Extended Abstracts of UIST '08* 8, 1–3.

[6] Matt Cumberlidge. 2007. *Business process management with JBoss jBPM*. Packt Publishing Ltd.

[7] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano. 2016. DevOps. *IEEE Software* 33, 3 (2016), 94–100. https://doi.org/10.1109/MS.2016.68

[8] Max Goldman, Greg Little, and Robert C. Miller. 2011. Real-Time Collaborative Coding in a Web IDE. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. Association for Computing Machinery, New York, NY, USA, 155–164. https://doi.org/10.1145/2047196.2047215

[9] Koen V. Hindriks. 2014. The Shaping of the Agent-Oriented Mindset. In *Engineering Multi-Agent Systems, LNCS, volume 8758*, Fabiano Dalpiaz, Jürgen Dix, and M. Birna van Riemsdijk (Eds.). Springer International Publishing, Cham, 1–14.

[10] Viviana Mascardi, Danny Weyns, Alessandro Ricci, Clara Benac Earle, Arthur Casals, Moharram Challenger, Amit Chopra, Andrei Ciortea, Louise A. Dennis, Álvaro Fernández Díaz, Amal El Fallah-Seghrouchni, Angelo Ferrando, Lars-Åke Fredlund, Eleonora Giunchiglia, Zahia Guessoum, Akin Günay, Koen Hindriks, Carlos A. Iglesias, Brian Logan, Timotheus Kampik, Geylani Kardas, Vincent J. Koeman, John Bruntse Larsen, Simon Mayer, Tasio Méndez, Tasio Méndez, Juan Carlos Nieves, Valeria Seidita, Baris Tekin Tezel, László Z. Varga, and Michael Winikoff. 2019. Engineering Multi-Agent Systems: State of Affairs and the Road Ahead. *SIGSOFT Engineering Notes (SEN)* (January 2019).

[11] T. Savor, M. Douglas, M. Gentili, L. Williams, K. Beck, and M. Stumm. 2016. Continuous Deployment at Facebook and OANDA. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. Association for Computing Machinery, 21–30.

[12] Sho-Huan Simon Tung. 1992. Interactive modular programming in Scheme. *ACM SIGPLAN Lisp Pointers* V, 1 (1992), 86–95. https://doi.org/10.1145/141478.141512

[13] Jason Vandeventer and Benjamin Barbour. 2012. CodeWave: A Real-Time, Collaborative IDE for Enhanced Learning in Computer Science. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)*. Association for Computing Machinery, New York, NY, USA, 75–80. https://doi.org/10.1145/2157136.2157160

[14] Ge Wang and Perry R Cook. 2004. On-the-fly Programming: Using Code as an Expressive Musical Instrument. *NIME '04 Proceedings of the 2004 conference on New interfaces for musical Expression* (2004). https://doi.org/10.1017/S1092852916000900

[15] Makarius Wenzel. 2013. READ-EVAL-PRINT in Parallel and Asynchronous Proof-checking. *Electronic Proceedings in Theoretical Computer Science* 118 (2013), 57–71. https://doi.org/10.4204/EPTCS.118.4