A Framework for Collaborative and Interactive Agent-oriented Developer Operations

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:

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. 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. Jacamo-web supports on-the-fly development of organizations and artifacts thanks to a built-in Java compiler. Our extension 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.

Our extension provides new interactive facilities. The IDE automatically updates its content keeping the engineer informed about changes done by others. For instance, when the engineer creates and updates the seller agent, the engineer that is programming the buyer may perceive and interact with the new state of the seller. 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.

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 paid instead of paid. As the usefulness checking function does not find any mention of paid 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 without affecting running instances.

![Figure 1: Understandability warning.](image1)

![Figure 2: Notification of concurrent use to prevent data loss.](image2)

![Figure 3: The commit dialog highlights potential issues.](image3)

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


