SUBJECT FOR THE INSA-UT PHD PROGRAM OF THE CHINA SCHOLARSHIP COUNCIL

SESSION 2022-2023

Institution: Laboratoire Connaissance et Intelligence Artificielle Distribuées (CIAD, http://www.ciad-lab.fr), Université de Technologie de Belfort-Montbéliard (UTBM, http://www.utbm.fr), France

Title of the subject:

Visual Programming of Autonomous Agents for Smart Building Operating System

Keywords: Multiagent Systems, Distributed Artificial Intelligence, Programming Language, Visual Programming

Supervisor: Prof. Dr. Stéphane GALLAND.
Université de Technologie de Belfort-Montbéliard
stephane.galland@utbm.fr
http://www.ciad-lab.fr/author-10836/

1. Description of the Hosting Institution

The 'Université de Bourgogne Franche-Comté' (UBFC) is a community of universities and high schools with around 56,000 students, located at the Center-East of France. It is composed of 22 geographical sites. UBFC was created April 1st, 2015. The founding members are: Burgundy University (uB) , Franche-Comté University (UFC), Université de Technologie de Belfort-Montbéliard (UTBM), High National School of Mecanics and Microtecnics (ENSMM), AgroSup Dijon, Burgundy School of Business (BSB, formerly ESC Dijon).

The 'Université de Technologie de Belfort-Montbéliard' (UTBM) is a public higher education and research institution located in the towns of Belfort, Sévenans and Montbéliard (Franche-Comté, France). The university has 2,555 students for the academic year 2011-2012. UTBM comes from the merging of the National School of Engineers of Belfort (ENIBe) created in 1962, and the department of the University of Technology of Compiègne established in 1985 in Sevenans, becoming "Sévenans Polytechnic Institute" (IPSE) in 1991. UTBM was founded in 1994. It belongs, like the University of Technology of Compiègne, the University of Technology of Troyes, and the University of Technology of Shanghaï, to the Network of the Technology Universities. UTBM is a member of the National Council of the French High Schools, the Conference of Directors of French Engineering Schools, the Conference of University Presidents, founder of the Bourgogne Franche-Comté Research and

Education Pole, founding member of the ARC-Europe Project, and member of UT Group. UTBM is authorized by the Ministry of Higher Education and Research to issue engineering diplomas in the following domains: automatic, industrial electronics, IT, mechanical, production systems, mechanical design and ergonomics. UTBM is developing research activities in cohesion with the industrial environment of the north Franche-Comté: Land transport and energy. Seven laboratories research is organized around UTBM.

The **CIAD laboratory** (Laboratoire Connaissance et Intelligence Artificielle Distribuées, http://www.ciad-lab.fr) is a multidisciplinary research laboratory that is hosted by UTBM and UB. In 2019, Researchers from UTBM and UB has created the CIAD staff. Part of the CIAD activities us supported by the multiagent and simulation research group whose head is Prof. Dr. S. Galland.

The core activity of CIAD in relation with this PhD subject is concerned with computer programming languages, methods and tools, and aims at defining suitable abstractions, methodologies either formal or semi-formal and tools for engineering multiagent software based on organizational and behavioral theories, and that constitute a base blocks for the other research projects of the team. The scientific works of the team is organized around three fundamental axes: Agent-Oriented Software Engineering and Formal models, Multiagent-based Simulation, and Agent's architectures. Our main application areas cover Intelligent Transport Systems (ITS), traffic and pedestrian's simulation in virtual environments. In the context of this PhD subject, the two following areas draw the attention of the team: Simulation in virtual environments, and the Modelling, simulation and control of multimodal traffic.

The PhD candidate will be hosted on the **campus of Belfort of UTBM**.

2. Scientific Context and Problems

In this thesis, we introduce and discuss an approach for agent-oriented visual programming. We argue that this is a first and very important step to investigate whether Multi-Agent Oriented Programming (MAOP) actually may enable individuals without experience in programming to create or modify ever more pervasive and ever more autonomous systems in their surroundings, by raising the level of abstraction from procedures or objects to agents, which people are more familiar with from their everyday experience. In doing so, we also revisit one of the core tenets of agent-oriented programming: the intentional stance and ascribing mental qualities to systems. Having a visual agent programming language geared towards non-technical users would allow us to validate this assumption in the context of systems engineering.

Our endeavor is furthermore motivated by two concrete issues experienced in a concrete industrial scenario based on the Smart Buildings (SB). The first one is the ever-increasing interest in forms of end-user programming that shall enable not only experienced programmers but ideally domain experts without programming experience to create or modify software systems of different complexity. The second one is the need to create or modify solutions featuring different degrees of autonomy of software components in performing tasks in a flexible way, dealing with open, dynamic, distributed SB environments. From this angle, at the same time, the use of semantic-web technologies allows to discover high-level actions at run-time, which promotes the serendipitous creation of applications in such environments – given a proper level of abstraction for exploiting them.

Smart Building has been transforming the building sector for several years. Thanks to digital technologies, such as the Internet of Things (IOT) and Artificial Intelligence (AI), buildings are becoming real service platforms for users. For example, by using multiagent algorithms, the energy consumption of the occupants of a building is thus adjusted and automated, or the building is able to provide specific services to the inhabitants [1].

3. Goals of the PhD Works

Visual Programming (VP) is defined as the action of programming using more than one dimension to convey semantics [2]. Traditional text-based programming is considered mono-dimensional since, although visually organized on a 2D screen, code can be seen as a single string of characters. In general, the main goals of VP are to make programming more accessible to some particular audience and to improve the correctness and speed with which people perform programming tasks.

VP has been often paired with the idea of End-User Programming (EUP) and End-User Development (EUD), which can be seen as programming done by someone who is not a programmer in their regular work life [3]. This field is highly relevant if we think about the fact that most software (in terms of quantity) is written by these people. EUP is sometimes also defined depending on the goal that the programmer has when writing software rather than his skill level. This is usually to automate a personal task, thus not necessarily taking into consideration all the features that are typical in software engineering such as maintenance or testing [4].

The two most popular examples of Visual Programming Environments applied to end-user (and novice users) are block-based visual programming and flow-based visual programming [5]. In block-based programming the core idea is to present the user with a primitives-as-puzzle-pieces metaphor to give users visual cues indicating where and how instructions may be used by dragging-and-dropping them together. Syntax errors are prevented since the programming environments forbids to snap together blocks that shouldn't be connected [6].

Flow-based abstractions instead see programming as the coordination of parallel flows to transform data. This is achieved through the use of components acting as "black boxes" that can be joined together to create streams of computation. The focus is on reusing predefined functions and combining them together to achieve the final result [7].

Both approaches have been proven successful in introducing people to programming spanning a range of different domains, including IoT [8] which is the one taken in consideration as the use case for our Visual AOP language as well. Some popular examples of block based tools are MIT AppInventor [9] and Scratch [10]. Among of the most famous examples of flow-based programming instead are Node-RED 3 and IFTTT 4.

With the spreading of automation and, hence, the use of computers, through-out all aspects of our lives, there is an ever-growing gap between domain expertise and implementation expertise. The tasks that are delegated to automation systems are becoming increasingly abstract – to give an example, in today's building automation the user is not anymore expected to manually control window blinds; rather, the goal of keeping the building in a comfortable state with respect to lighting, temperature, and glare is delegated to the system, and users merely set the parameters of this goal.

Against this backdrop, it is important that EUP is extended to scenarios where domain experts (e.g., building automation systems engineers) are enabled to create and configure the behavior of a system at a higher abstraction level as well – this extension leads to an extension of the EUP concepts to something akin to domain-expert programming (DEP), where the programmer is not necessarily just any final customer of a software solution, but a person with deep knowledge of the domain in which the software may be useful. With this, though, we're not meaning to exclude "regular end-users" since everybody can be considered the domain expert of his own home.

We argue that by empowering such experts with visual tools it would be possible to reduce this gap. Due to the aforementioned higher abstraction level, it is at the same time highly compelling to build such DEP systems not on the basis of paradigms that traditionally underlie EUP programming (e.g., procedural, flow-based, or object-oriented programming) but to instead use MAOP and its abstractions as the foundation of our system.

The idea presented in this PhD thesis proposal is strongly related to works in literature that investigate high-level approaches to agent-oriented programming, that have been proposed since the very beginning of agent programming literature. In [11], authors proposed an end-user programming system enabling users to program the behavior of their personal software agent using rules for their agents to follow. This work can be considered a specific example of approaches that explored programming by demonstration or programming by example [1], investigating a different way to think about programming, towards a perspective in which the programmer is more a teacher instructing an agent what to do. A main seminal example in this case is the KidSim environment [12], designed for children, to support their creative constructions of simulations to learn. In the related context of teaching novice programming [13], [14] proposed a visual programming language/environment (an extension of Scratch called BYOB, today known as Snap[1] makes it possible to specify goals and plans using a block-based visual notation.

The goal of the PhD thesis is to extend the existing open-source framework **SARL** (www.sarl.io), which is an agent oriented programming language and its associated meta-model, in order to propose a graphical programming that has the same operational semantic as SARL, or simply able to be translated to SARL code.

The expected application of this thesis is to propose a tool for implementing autonomous agents for a smart building. In particular, the PhD student will consider the definition of a Building Operating System (BOS) in which the visually programmed agents will be deployed. BOS represents the necessary link between a real building and its virtual twin, between a building and its data on the one hand, and the services on the other hand. The objective of BOS is indeed to pool information flows between field sensors and service applications. Consequently, the generated agents will be connected to and enhance the features provided by BOS. **One of the goal of this PhD thesis is to contribute to the creation of a "Smart BOS" – or "Agent-oriented BOS".**

1 https://snap.berkeley.edu

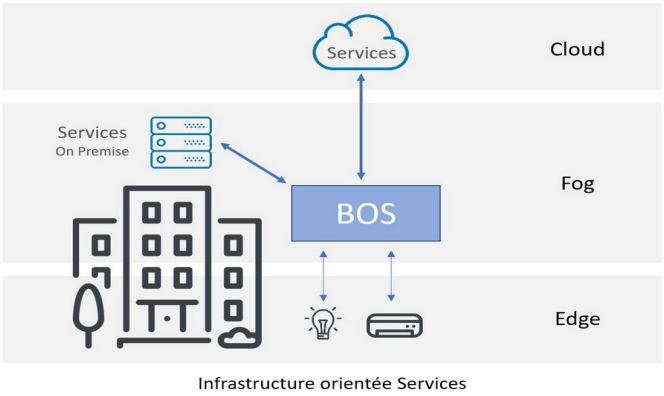

Figure 1: BOS in smart buildings

4. Privacy, Data Management and European GDPR

No personal or sensitive data source that are covered by the GDPR will be gathered or used in this thesis.
Only synthetic data, i.e. automatically generated by computer models, will be used into the simulation scenarios.

5. Expected Background for the Candidates

This section lists the expected background for the candidates:

- **Computer programming with object-oriented paradigm (mandatory knowledge)**
- **Object-oriented programming language, e.g. Java or C# (mandatory knowledge)**
- Artificial Intelligence and Multiagent Systems (recommended but not mandatory)
- Computer Language Theory (recommended but not mandatory)
- Computer language grammars and compilers (recommended but not mandatory)

6. Expected Working Plan

Year 1 (Months 1-12)

- Do a Systematic Literature Review (SLR) according to the international standards in order to highlight the key research questions in the field of this PhD thesis
- Selection of one or two research questions from the SLR in order to be handled by the PhD candidate.
- Writing and publication of one paper into an international journal of Rank Q1 or Q2 that explains the SLR

Year 2 (Months 13-24)

- Elaboration of the agent metamodel, models and tools.
- Definition of the visual language operational semantic and syntax.
- Writing and publication of papers into international conferences with ready committee.
- Writing and publication of a paper into an international journals of Rank Q1 or Q2 that explains the proposed models

Year 3 (Months 25-36)

- Implementation of an experimentation in the smart building of the UTBM campus.
- Writing and publication of papers into an international conference with ready committee.
- Writing and publication of a paper into an international journals of Rank Q1 or Q2 that explains the experiments.
- Preparation of the final PhD document

Year 4 (Months 37-42)

- Preparation of the final PhD document
- Official oral defense

7. Five significant scientific publications related to this PhD subject

- **S. Galland**, S. Rodiguez. "Semantic Transformation from SARL Agent-oriented Statements to Java Object-oriented Statements." In International Journal of Artificial Intelligence, vol 139-153, pp. 2, oct 2019. ISSN: 0974-0635. **IF : 9.88; Q1**.
- Stéphane GALLAND, Sebastian RODRIGUEZ, Nicolas GAUD. *"Run-time Environment for the SARL Agent-Programming Language: the Example of the Janus platform."* In International Journal on Future Generation Computer Systems, Elsevier, 2017. ISSN 0167-739X. **IF : 6.125 ; Q1.**
- S. Galland, S. Rodiguez. "Semantic Transformation from SARL Agent-oriented Statements to Java Object-oriented Statements." In International Journal of Artificial Intelligence, vol 139-153, pp. 2, oct 2019. ISSN: 0974-0635. **IF : 9.88; Q1**.
- Yazan MUALLA, Amro NAJJAR, Alaa DAOUD, Stéphane GALLAND, Christophe NICOLLE, Ansar-Ul-Haque YASAR, Elhadi SHAKSHUKI. "Agent-based simulation of unmanned aerial vehicles in civilian applications: A systematic literature review and research directions." In Future Generation Computer Systems (FGCS), vol. 100, pp. 344-364, Elsevier, 2019. DOI: 10.1016/j.future.2019.04.051. **IF : 6.125 ; Q1.**
- A. Lombard, J. Buisson, A. Abbas-Turki, **S. Galland**, A. Koukam. "Curvature-Based Geometric Approach for the Lateral Control of Autonomous Cars." In Journal of Franklin Institute, jul 2020. ISSN: 0016-0032. DOI: 10.1016/j.jfranklin.2020.07.015. **IF : 4.036 ; Q1**.

8. References

1. BUCKMAN, Alex H., MAYFIELD, Martin, et BECK, Stephen BM. What is a smart building?. *Smart and Sustainable Built Environment*, 2014.
2. Burnett, M.M., McIntyre, D.W.: Visual programming. COMPUTER-LOS ALAMITOS- 28, 14–14 (1995)
3. Nardi, B.A.: A small matter of programming: perspectives on end user computing. MIT press (1993)
4. Ko, A.J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., et al.: The state of the art in end-user software engineering. ACM Computing Surveys (CSUR) 43(3), 1–44 (2011)
5. Mason, D., Dave, K.: Block-based versus flow-based programming for naive programmers. In: 2017 IEEE blocks and beyond workshop (B&B). pp. 25–28. IEEE (2017)
6. Weintrop, D.: Block-based programming in computer science education. Communications of the ACM 62(8), 22–25 (2019)

7. Morrison, J.P.: Flow-based programming. In: Proc. 1st International Workshop on Software Engineering for Parallel and Distributed Systems. pp. 25–29 (1994)
8. Ray, P.P.: A survey on visual programming languages in internet of things. Scientific Programming 2017 (2017)
9. Pokress, S.C., Veiga, J.J.D.: MIT App Inventor: Enabling personal mobile computing. arXiv preprint arXiv:1310.2830 (2013)
10. Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E.: The scratch programming language and environment. ACM Transactions on Computing Education (TOCE) 10(4), 1–15 (2010)
11. Terveen, L.G., Murray, L.T.: Helping users program their personal agents. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. p. 355–361. CHI '96, Association for Computing Machinery, New York, NY, USA (1996). https://doi.org/10.1145/238386.238568, https://doi-org.ezproxy.unibo.it/10.1145/238386.238568
12. Cypher, A., Smith, D.C.: Kidsim: End user programming of simulations. In: Proceedings of the SIGCHI Conference on Human Factors in Comput- ing Systems. p. 27–34. CHI '95, ACM Press/Addison-Wesley Publishing Co., USA (1995). https://doi.org/10.1145/223904.223908, https://doi-org.ezproxy.unibo.it/10.1145/223904.223908
13. Smith, D.C., Cypher, A., Tesler, L.: Programming by example: Novice programming comes of age. Commun. ACM 43(3), 75–81 (mar 2000). https://doi.org/10.1145/330534.330544, https://doi.org/10.1145/330534.330544.
14. Hu, M., Winikoff, M., Cranefield, S.: Teaching novice programming using goals and plans in a visual notation. In: Proceedings of the Fourteenth Australasian Computing Education Conference. vol. 123, pp. 43–52 (01 2012)