Self-Evolving Software Agents

Extended Abstract

Marco Robol
University of Trento
Trento, Italy
marco.robol@unint.it

Paolo Giorgini
University of Trento
Trento, Italy
paolo.giorgini@unitn.it

ABSTRACT

Autonomous agents can adapt their behaviour to changing environments, but remain bound to requirements, goals, and capabilities fixed at design time, preventing genuine software evolution. This paper introduces *self-evolving software agents*, combining BDI reasoning with LLMs to enable autonomous evolution of goals, reasoning, and executable code. We propose a BDI–LLM architecture in which an automated evolution module operates alongside the agent's reasoning loop, eliciting new requirements from experience and synthesizing corresponding design and code updates. A prototype evaluated in a dynamic multi-agent environment shows that agents can autonomously discover new goals and generate executable behaviours from minimal prior knowledge. The results indicate both the feasibility and current limits of LLM-driven evolution, particularly in terms of behavioural inheritance and stability.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; Philosophical/theoretical foundations of artificial intelligence; Intelligent agents; *Generative and developmental approaches*; Simulation evaluation; • **Software and its engineering** → *Software creation and management*; *Software organization and properties.*

KEYWORDS

Software Evolution; Adaptation; Autonomous Agents; BDI model; Artificial Intelligence; LLMs

ACM Reference Format:
Marco Robol and Paolo Giorgini. 2026. Self-Evolving Software Agents: Extended Abstract. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 3 pages. https://doi.org/10.65109/HKPK4104

1 MOTIVATION AND BACKGROUND

Modern software systems increasingly operate in open and dynamic environments shaped by machine learning, IoT, and cloud computing [1, 23], where requirements, assumptions, and operational contexts evolve over time [3, 9]. While self-adaptive systems and autonomous agents can modify their behaviour at runtime, they typically remain bound to goals, requirements, and capabilities defined at design time. As a result, they can adapt, but not genuinely evolve. Software evolution, originally limited to code fixes and incremental updates [15], now entails revising system objectives, internal reasoning structures, and executable capabilities in response to emerging needs [8]. Despite extensive research on software evolution [6, 11, 25, 27], requirements engineering [2, 14, 24], and self-managing systems [10, 13, 22], current agent architectures lack explicit mechanisms to autonomously evolve their own requirements and code while preserving architectural coherence [19, 29]. Recent advances in Large Language Models [4] offer new opportunities to automate parts of the software evolution process, including requirement elicitation, design revision, and code synthesis. However, existing agentic AI approaches largely rely on prompt-driven behaviour and externally defined objectives, providing limited support for structured, long-term evolution.

This work addresses this gap by introducing a framework for self-evolving software agents that integrates automated software evolution principles within a BDI architecture, enabling agents to evolve goals, reasoning, and actions autonomously.

2 SELF-EVOLVING AGENTS ARCHITECTURE

Figure 1 illustrates the proposed BDI–LLM architecture for self-evolving software agents [12, 18, 31]. The model builds on a classical BDI reasoning loop [26, 31, 32], where the agent continuously updates beliefs, deliberates over desires, selects intentions, and executes plans in response to environmental perceptions. In addition to the standard reasoning loop, the architecture introduces an *Automated Evolution Module* that operates independently from runtime decision making. This module monitors the agent's experience and identifies unmet needs or novel opportunities that cannot be addressed by the current knowledge, goals, or action repertoire. When triggered, it initiates an automated software evolution cycle based on variation, selection, and inheritance.

The evolution process acts on three architectural layers of the agent: (i) knowledge representation and reasoning, by extending or revising perceptual and inference mechanisms; (ii) goal generation and decision making, by introducing new goals and intention-selection policies; and (iii) execution, by synthesizing or adapting executable actions and plans. By isolating evolution from the reasoning loop, the architecture preserves behavioural coherence while enabling autonomous, long-term evolution of the agent's internal structure. This approach addresses challenges in adaptive systems [8, 20, 30] and leverages recent advances in LLMs for automated code generation [4, 5, 7, 17].

3 PROTOTYPE AND PRELIMINARY EVALUATION

We implemented a prototype of the proposed self-evolving agent using a BDI control loop extended with an LLM-driven evolution

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/HKPK4104

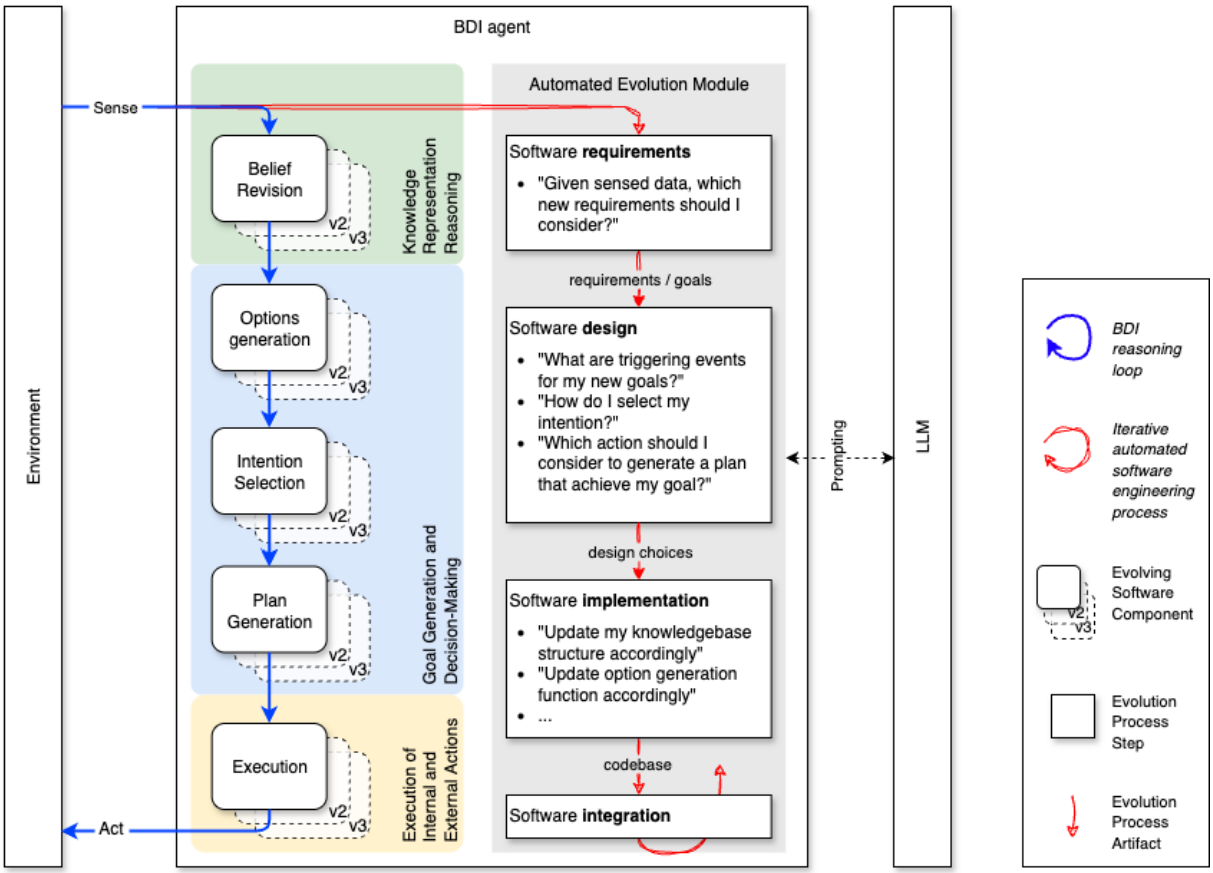

Figure 1: BDI–LLM architecture for self-evolving software agents. An automated evolution module operates alongside the classical BDI reasoning loop, enabling autonomous evolution of knowledge, goals, and executable actions.

module [21, 28]. The agent operates in a dynamic multi-agent environment inspired by the Deliveroo.js framework, where it must perceive the environment, discover goals, and generate executable plans while interacting with other agents and environmental constraints. The agent is initially provided only with a textual description of the environment and a minimal set of APIs, without predefined domain-specific knowledge or goals. The evolution module is triggered when perceived information cannot be interpreted or exploited using the current knowledge and goal structures. In such cases, the agent autonomously generates new goals, revises its reasoning structures, and synthesizes executable actions, which are then validated through interaction with the environment. Successful behaviours are retained and reused in subsequent situations, while ineffective ones are discarded.

Preliminary experiments show that the agent is able to autonomously discover operational goals and generate executable behaviours starting from minimal prior knowledge. At the same time, the results highlight current limitations of LLM-driven evolution, particularly in terms of behavioural inheritance and robustness as environmental complexity increases. These observations confirm the feasibility of autonomous software evolution while motivating further investigation on mechanisms for stabilising and reinforcing evolved behaviours.

4 CONCLUSIONS AND OUTLOOK

This paper introduced a framework for self-evolving software agents that integrates automated software evolution principles within a BDI architecture augmented by LLMs. By separating runtime reasoning from an explicit evolution module, the proposed approach enables agents to autonomously revise goals, reasoning structures, and executable actions, moving beyond traditional notions of behavioural adaptation. The prototype and preliminary evaluation demonstrate the feasibility of autonomous evolution in dynamic multi-agent environments, while also revealing current limitations in behavioural inheritance, stability, and scalability. These challenges point to the need for reinforcement mechanisms, memory consolidation, and more robust selection strategies.

Future work will focus on strengthening inheritance and long-term consistency, extending the evaluation to more complex multi-agent scenarios, and exploring collective and cooperative forms of software evolution among agents, as well as leveraging retrieval-augmented generation [16] to enhance LLM reasoning with external knowledge.

REFERENCES

[1] L. Bettini. 2015. *Implementing Domain-Specific Languages with Xtext and Xtend*. Packt Publishing, Birmingham, UK.

[2] B. W. Boehm. 1988. A spiral model of software development and enhancement. *ACM SIGSOFT Software Engineering Notes* 11, 4 (1988), 14–24.

[3] M. Böhm and A. Zimmermann. 2020. The Autonomous System Dilemma: Balancing Adaptability and Predictability. *IEEE Software* 37, 4 (2020), 44–49.

[4] R. et al. Bommasani. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021), e220119.

[5] Tom B Brown et al. 2020. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901.

[6] J. M. Burge and D. C. Brown. 1999. Software change: Cost, causes, and complexity. *Software Engineering Journal* 14, 3 (1999), 180–190.

[7] Mark Chen et al. 2021. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374* (2021).

[8] B. H. Cheng, H. Giese, P. Inverardi, and J. Magee. 2009. Software Engineering for Self-Adaptive Systems: A Research Roadmap. *Software Engineering for Self-Adaptive Systems* (2009), 1–26.

[9] T. H. Davenport and R. Kalakota. 2019. The potential for artificial intelligence in healthcare. *Future Healthcare Journal* 6, 2 (2019), 94–98.

[10] R. de Lemos, H. Giese, H. A. Müller, and M. Shaw. 2001. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems* 4, 2 (2001), 1–25.

[11] Juan Fernandez-Ramil, Dewayne Perry, and Nazim H. Madhavji (Eds.). 2006. *Software Evolution and Feedback: Theory and Practice*. Wiley, Chichester.

[12] S. Franklin and A. Graesser. 1996. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the International Workshop on Agent Theories, Architectures, and Languages*. Springer, Berlin, Heidelberg, 21–35.

[13] D. Garlan, S. Cheng, and A. Huang. 2004. Software architecture-based self-adaptation. *ACM SIGSOFT Software Engineering Notes* 30, 4 (2004), 1–7.

[14] M. Jackson. 1995. *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*. ACM Press/Addison-Wesley, New York, NY, USA.

[15] M. M. Lehman. 1980. Programs, life cycles, and laws of software evolution. *Proc. IEEE* 68, 9 (1980), 1060–1076.

[16] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[17] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *Science* 378, 6624 (2022), 1092–1097.

[18] Michael Luck, Peter McBurney, Onn Shehory, and Steve Willmott. 2005. *Agent Technology: Computing as Interaction (a roadmap for agent based computing)*. University of Southampton, Southampton, UK.

[19] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. C. Cheng. 2004. Composing adaptive software. *IEEE Computer* 37, 7 (2004), 56–64.

[20] Jörg P. Müller and Klaus Fischer. 2014. Application Impact of Multi-Agent Systems and Technologies: A Survey. In *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks*. Springer, Berlin, Heidelberg, 27–53.

[21] OpenAI. 2024. GPT-4o System Card. https://openai.com/research/gpt-4o. Accessed October 2025.

[22] P. Oreizy, N. Medvidovic, and R. N. Taylor. 1999. Architecture-based runtime software evolution. In *Proceedings of the 20th International Conference on Software Engineering*. IEEE, Kyoto, Japan, 177–186.

[23] J. Paris, L. Bass, and R. Kazman. 2021. Architecting AI-Based Systems: A Systematic Mapping Study. *Journal of Systems and Software* 175 (2021), 110895.

[24] D. L. Parnas. 1994. Software aging. In *Proceedings of the 16th International Conference on Software Engineering*. IEEE, Sorrento, Italy, 279–287.

[25] R. S. Pressman. 2005. *Software Engineering: A Practitioner's Approach* (6th ed.). McGraw-Hill, New York.

[26] A. S. Rao and M. P. Georgeff. 1995. BDI Agents: From Theory to Practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*. MIT Press, San Francisco, CA, USA, 312–319.

[27] I. Sommerville. 2010. *Software Engineering* (9th ed.). Addison-Wesley, Boston.

[28] Francesco Vaccari. 2024. *Self-Evolving Software Agents: An LLM-Based Approach*. Ph.D. Dissertation. University of Trento.

[29] N. M. Villegas and H. A. Müller. 1997. Software adaptation in dynamic environments. *Comput. Surveys* 35, 1 (1997), 34–45.

[30] J. Whittle, J. Hutchinson, and M. Rouncefield. 2011. The state of practice in model-driven engineering. *IEEE Software* 28, 3 (2011), 22–28.

[31] Michael Wooldridge. 2009. *An Introduction to MultiAgent Systems* (2nd ed.). John Wiley & Sons, Chichester, UK.

[32] M. Wooldridge and N. R. Jennings. 1995. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review* 10, 2 (1995), 115–152.