Extended Abstract: Skill-Based Architectures in Autonomous Systems: Lessons Learnt

Pierre Malafosse[1,2], Alexandre Albore[2], Jeremie Guiochet[1] and Charles Lesire[2]

[1]TRUST, LAAS-CNRS, Université de Toulouse, 31000, Toulouse, France
[2]DTIS, ONERA, Université de Toulouse, 31000, Toulouse, France

Abstract
The deployment of complex autonomous systems into open environment calls for robust, modular and verifiable software architectures. Skillset models, which encapsulate the capabilities of the autonomous robots into modular Skills, have emerged to address these challenges. Expressed within an intermediary layer of the robot's architecture, Skillsets break down high level actions produced by the Deliberative layer into lower level actions, executed by functional components. The use of a Domain Specific Language (DSL) for Skillset modeling allow formal methods to provide robust code generation and model verification. However, as Skill-based architectures are more and more developed, the true benefits of opting for such an approach have yet to be investigated.

This extended abstract presents an early-stage analysis of recent Skill-based architectures making use of the ONERA Robot Skill Toolchain[1] with a focus on identifying best practices, common pitfalls, and offering guidelines for future developments. The experiment relies on the works of several french laboratories, all making use of the Robot Language DSL formalized by Albore et al.[1]. The findings suggest that while Skill-based architectures present undeniable advantages in the development of autonomous systems, there is currently a need for more standardized frameworks.

Keywords
Autonomous systems, robotic architectures, robotics, Skill-based architectures

1. Introduction

The development of autonomous systems presents the major challenge of integrating low-level functions (e.g., actuation) and high-level decision-making capabilities (e.g., task scheduling) into robust and reliable software architectures [2, 3, 4]. A common approach involves using layered architectures, especially a three-layer architecture [5, 6] (Functional, Executive, and Deliberative layers). In this architecture, *Skills* are abstractions of the robot's capabilities [7]. A robot has a *Skillset* from which Skills can be executed following plans provided by the Deliberative layer. Skills are formally modeled in the Robot Language DSL [8] to facilitate formal verification [1, 9, 10] and reuse. Their flexibility and adaptability enables non-experts to program and control robots by redesigning missions or adding new Skills [11, 12]. This *Skillset model* is used for code generation creating the robot's Executive layer, the *Skillset Manager*. Code generation reduces the occurrence of faults and ensures the Skills behave as defined in the DSL. Finally, developers define the *Skillset Implementation*, the interface between the Functional layer and the Skillset Manager.

The Robot Language DSL has been evolving over the past few years, improving both on the aspects of robustness and verifiability [1, 9, 10]. Several developers have implemented the Skill-based Executive layer on different platforms and for different use cases. Yet, there is currently no established development process or framework for developing such architectures. The modeling of Skills, the structure of the Skillset implementation and the location of error handling features are left to the user's preference. Design choices and trends must be investigated in the hope of unveiling good practices and common mistakes. This study marks the beginning of a Ph.D. thesis with the aim to improve the confidence

[1]Robot Skills documentation

PhD Symposium of the 19th International Conference on Integrated Formal Methods (iFM) at the University of Manchester, UK, 12 November 2024.

pierre.malafosse@onera.fr (P. Malafosse); alexandre.albore@onera.fr (A. Albore); guiochet@laas.fr (J. Guiochet); charles.lesire@onera.fr (C. Lesire)

level in the three-layer architectures used in autonomous systems by proposing new approaches to specify Skill-level and multi-level recovery strategies, with a focus on a Skill-based Executive layer as defined by Albore et al.[1], and displayed in Figure 1.

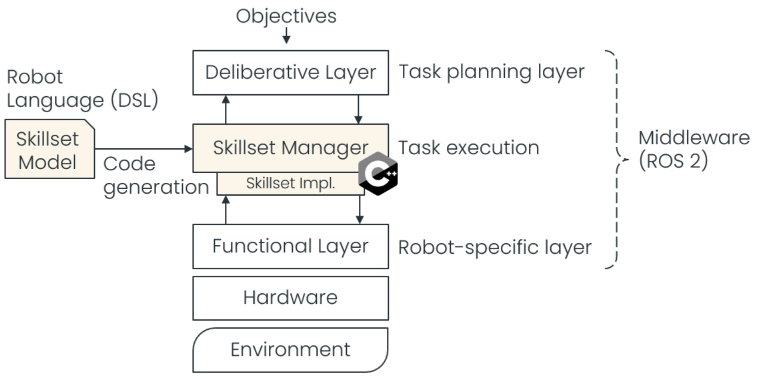

Figure 1: 3-layer architecture as defined by Albore et al.[1] with the Skill-based layer implemented as the Executive layer.

As a first step, we present an early-stage analysis of recent Skill-based architectures making use of the ONERA Robot Skill Toolchain[1] with a focus on identifying best practices, common pitfalls, and offering guidelines for future developments. To this end, we focus on three research questions (RQs):

RQ 1. To which extent is the level of abstraction proposed by the Skills suitable for different applications?
Metrics: Assessment of the Reusability and adaptivity of Skills both across missions and across systems.

RQ 2. In what ways does the abstraction level of Skills influence the management of system complexity?
Metrics: Assessment of the Mission-level complexity and architecture-level complexity

RQ 3. Does the use of Skills facilitate the integration of error detection and handling mechanisms?
Metrics: Nature and localization of the error handling mechanisms in the Skill-based architecture.

With these three research questions set, we present the sources and methods used for the experiment in Section 2. Section 3 then summarizes our observations and results before concluding with Section 4.

2. Materials and methods

First, lets us point out that the studied Skill-based approach is an open-source research project, with a limited community of developers. Nevertheless, over the past few years, numerous works using this technology have been conducted. Discarding versions form before 2022, the 22 most recent projects involving a Skill-based architecture have been investigated in the present work. Data ranging from interpretations of the code to interviews with the developers has been collected. These projects were conduced by several French laboratories, namely ONERA and LAAS-CNRS in Toulouse as well as the LIRMM in Montpellier. The data is diverse, as are involved a wide variety of robot types: Manipulator Arms, Unmanned Aerial Vehicles (UAVs), Unmanned Marine Vehicles (UMVs), Unmanned Ground Vehicles (UGVs) and Legged Robots, sometimes working all together to achieve the same goal. Each project is treated as an individual case study and each Skill-based architecture is assessed, following the research questions, on how the Skills were designed, reused and adapted. Thirteen individuals (including Ph.D. students and senior researchers) involved in these projects were interviewed. In addition to code analysis, semi-structured interviews were conducted to elucidate the mission contexts, rationale behind design decisions, and practical challenges faced during the implementation of the Skills. The interview protocol was meticulously designed to address the three primary research questions.

[1]Robot Skills documentation

Developers provided valuable insights into the process of implementing Skill-based architectures, specifically focusing on strategies for managing system complexity, Skill correctness, and integrating error-handling mechanisms.

3. Results

3.1. RQ 1: Suitability of Skill Abstraction

First, we investigated the reusability and genericity of Skills. The analysis of the 22 project codes shows that while Skills are theoretically meant to be reusable and platform-independent, they tend to only be reused within the same robot type (e.g., UAVs, UGVs). Generic Skillset models have been developed for each type and the crossing of a Skill model between different robot types is rare. This is mainly due to diverging functional needs. For example, while UGVs and UAVs may share the same capability of reaching a specific waypoint, UAVs need to process additionnal information such as altitude values, flight status or safety requirements. Therefore, distinct movement-related Skills have been implemented. This observation leads to the matter of adaptivity of the Skills. Due to the lack of modularity and specific project needs, developer teams adopted diverging solutions. Some directly modified generic Skillsets to fit mission-specific needs, while others created new, dedicated Skillsets for a precise robot type. We observe an unbalance between genericity and adaptivity of Skill models: On the one hand, the more a Skill is generic, the more difficult it is to implement for specific needs. On the other hand, highly adaptive Skillsets are bound to their initial robot type and cannot easily be adapted elsewhere. The framework does not currently propose explicit, documented solutions on the matter. However, we observe significant similarities between the existing models and argue that the creation of a truly generic Skillset model is possible. To this end, an unified framework for designing Skill-based architectures is needed.

The key takeaways regarding RQ 1 are:

- The framework does not currently propose an explicit and documented list of generic Skills.
- The framework does not include a built-in extension mechanism (such as inheritance in object-oriented programming) to ease the redefinition and specialization of Skillsets.
- There is currently no training material on generic Skill reuse.

3.2. RQ 2: Effect on Complexity Management

Complexity management can be tackled in many ways. First insights from the interviewed participants concerned mission complexity. In this context, the Skill abstraction helps simplify interactions with non-experts and stakeholders by focusing on what the robot does rather than how it operates at a low level. This approach was particularly useful in projects with non-technical stakeholders, such as marine biologists in one of the case studies.

Another perspective is at the architecture level, where Skills may also play a role for managing complexity. According to the interviewed participants, without a Skill-based architecture, the development process would be longer and significantly more complex. Indeed, the abstraction of the robot's capabilities into Skills prevents the developers from making direct connections between high level tasks and low-level functional components: The Deliberative layer only reasons in term of Skill activation while the Skills manage the execution of the lower-level components. This modular approach also simplifies the integration of new functionalities. The use of ROS2 as the middleware distributing the Skill-based architecture may also influence the architecture-level complexity. The Executive layer uses ROS2 to harvest and propagate useful information to the whole system. Furthermore, robots from different projects (e.g., Boston Dynamics®Spot and Kinova®manipulator) were able to communicate and coordinate Skills effectively thanks to the shared middleware.

Key takeways regarding RQ 2 are:

- The abstraction of the robot's capabilities allows for simpler and clearer mission definitions.

- The abstracted Skills simplify the implementation and/or modification of the architecture by allowing for clear connections between modular components.
- Thanks to the ROS2 middleware, the Skillset Manager is able to centralize and spread information internally as well as externally, in multi-agent scenarios.

3.3. RQ 3: Effect on Error Detection and Handling

The Skillset manager is C++ code generated from a Skillset model which has been formally verified beforehand [1, 9]. This approach reduces the occurrence of faults within the Executive layer and ensures the Skills behave as intended. Furthermore, Skills provide error detection and handling mechanisms through the definition of preconditions, invariants, and postconditions which prevent a failure from the neighboring layers to propagate further. The Skillset implementation, making the link with the functional layer, is user defined and allows developers to integrate custom error detection and recovery mechanisms. However, what has transpired through both the interviews and the code reviews is the absence of method to identify error conditions and reactions and how to include them into the architecture. A work was done by Medina et al. [13] using Fault Tree Analysis (FTA) to create Skill fault models and help identify the potential causes of failures of the system. Albore et al.[1] made use of these Skill fault models to implement fallback modes onto a Behavior Tree and identified missing detection mechanisms within the Skillset implementation. However, we observed no unified practice for including error detection and handling within the 22 projects, apart from the mandatory formal verification of the Skillset model.

Key takeways regarding RQ 3 are:

- There is a lack of method to bind fault propagation analysis to Skill models
- There is a need for documented good practices to help developers choose the nature and location of the error detection and recovery mechanisms
- There is a need for tools to identify scenarios in which multiple (possibly incompatible) recovery actions are simultaneously triggered.

4. Conclusion

The present study has demonstrated, through an examination of 22 projects, the benefits and challenges inherent to the use of Skill-based architectures. Our findings indicate that, while Skills offer a robust mechanism for modular programming and rapid reconfiguration, they have yet to meet all their promises. Indeed, due to a lack of genericity and adaptivity, such implementations are currently not fully accessible to non-experts, because applications to specific robotic types, with specific functional needs, necessitate a process of careful consideration and adaptation. Regarding the genericity and adaptability of Skills, we posit that the development of truly generic Skill models is feasible. While this endeavor is beyond the scope of the current Ph.D. research, future studies should be conducted in this direction.

This study has highlighted a clear benefit of implementing Skill-based architectures into increasingly complex autonomous systems: A list of benefits in term of mission-level and architecture-level complexity management has transpired from the interviews while no significant downsides have been observed. On top of the the previously mentioned aspects, complexity management can therefore be a reason why developers choose to use Skills.

Our study also underscores the need of a clearly defined framework for the development of both the Skillset models and the Skillset implementations. The absence of formal guidelines allows for significant flexibility in the approach taken by the developer. This may result in inconsistencies within the implementations and could eventually compromise the safety and robustness of the system. The same lack of precise guidelines applies for the implementation of error detection and handling mechanisms.

In this context, the Future works related to the Ph.D. will be focusing on Skill-level error handling as well as a multi-level recovery framework with the aim of addressing the need for guidelines to safer, more robust implementations. To this end, we currently investigate the capabilities of BDI agents in term of failure handling [14]. A BDI-based Deliberative layer coupled with a Skill-based Executive layer could indeed prove interesting in term of failure handling, and multi-level recovery [15]. Furthermore, the formal aspect of both paradigms may allow for the creation of robust formal verification tools [16, 17, 18].

References

[1] A. Albore, D. Doose, C. Grand, J. Guiochet, C. Lesire, A. Manecy, Skill-based design of dependable robotic architectures, Robotics and Autonomous Systems 160 (2023). doi:10.1016/j.robot.2022.104318.

[2] T. Lozano-Perez, Robot programming, Proceedings of the IEEE 71 (1983) 821–841.

[3] T. Lozano-Pérez, R. A. Brooks, An approach to automatic robot programming, International Conference on Scientific Computing (1986) 61–69. doi:10.1145/324634.325195.

[4] R. A. Brooks, A robust layered control system for a mobile robot, IEEE Journal on Robotics and Automation 2 (1986) 14–23. doi:10.1109/JRA.1986.1087032.

[5] E. Gat, R. P. Bonnasso, R. Murphy, et al., On three-layer architectures, Artificial intelligence and mobile robots 195 (1998) 210.

[6] R. Simmons, D. Apfelbaum, A task description language for robot control, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications, 1998.

[7] J. Urzelai, D. Floreano, M. Dorigo, M. Colombetti, Incremental robot shaping, Connection Science 10 (1998) 341–360. doi:10.1080/095400998116486.

[8] C. Lesire, D. Doose, C. Grand, Formalization of robot skills with descriptive and operational models, IEEE International Conference on Intelligent Robots and Systems (2020) 7227–7232. doi:10.1109/IROS45743.2020.9340698.

[9] B. Pelletier, C. Lesire, D. Doose, K. Godary-Dejean, C. Dramé-Maigné, Skinet, a petri net generation tool for the verification of skillset-based autonomous systems, Electronic Proceedings in Theoretical Computer Science, EPTCS 371 (2022) 120–138. doi:10.4204/EPTCS.371.9.

[10] B. Pelletier, C. Lesire, C. Grand, D. Doose, M. Rognant, Predictive runtime verification of skill-based robotic systems using petri nets, in: Proc. of IEEE International Conference on Robotics and Automation, volume 2023-May, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 10580–10586. doi:10.1109/ICRA48891.2023.10160434.

[11] C. Laugier, T. Fraichard, P. Garnier, I. E. Paromtchik, A. Scheuer, Sensor-based control architecture for a car-like vehicle, Autonomous Robots 6 (1999) 165–185. doi:10.1023/A:1008835527875/METRICS.

[12] A. Bjorkelund, L. Edstrom, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. G. Robertz, D. Storkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt, H. Bruyninckx, On the integration of skilled robot motions for productivity in manufacturing, in: Proc. of IEEE International Symposium on Assembly and Manufacturing (ISAM), 2011. doi:10.1109/ISAM.2011.5942366.

[13] G. C. Medina, J. Guiochet, C. Lesire, A. Manecy, A skill fault model for autonomous systems, in: Proceedings of the 4th International Workshop on Robotics Software Engineering, 2022, pp. 55–62.

[14] S. Sardina, L. Padgham, A BDI agent programming language with failure handling, declarative goals, and planning, Autonomous Agents and Multi-Agent Systems 23 (2011) 18–70.

[15] L. A. Dennis, M. Fisher, Actions with durations and failures in BDI languages, in: ECAI 2014, IOS Press, 2014, pp. 995–996.

[16] B. Archibald, M. Calder, M. Sevegnani, M. Xu, Quantitative modelling and analysis of bdi agents, Software and Systems Modeling 23 (2024) 343–367.

[17] M. Xu, T. Rivoalen, B. Archibald, M. Sevegnani, can-verify: A verification tool for bdi agents, in: International Conference on Integrated Formal Methods, Springer, 2023, pp. 364–373.

[18] P. Stringer, R. C. Cardoso, C. Dixon, M. Fisher, L. A. Dennis, Adaptive cognitive agents: Updating

action descriptions and plans, in: European Conference on Multi-Agent Systems, Springer, 2023, pp. 345–362.