Leveraging Context-Oriented Programming to Implement Normative Rules in Autonomous Systems

Extended Abstract

Roberto Casadei
University of Bologna
Cesena, Italy
roby.casadei@unibo.it

Martina De Sanctis
Gran Sasso Science Institute
L'Aquila, Italy
martina.desanctis@gssi.it

Gianluca Filippone
Gran Sasso Science Institute
L'Aquila, Italy
gianluca.filippone@gssi.it

Sara Pettinari
Gran Sasso Science Institute
L'Aquila, Italy
sara.pettinari@gssi.it

Gian Luca Scoccia
Gran Sasso Science Institute
L'Aquila, Italy
gianluca.scoccia@gssi.it

Nicolas Troquard
Gran Sasso Science Institute
L'Aquila, Italy
nicolas.troquard@gssi.it

ABSTRACT

The increasing sensitivity to human aspects in autonomous systems engineering calls for principled approaches to embed normative concerns into their behaviour. Recent research has focused on expressing and validating sets of social, legal, ethical, empathetic, and cultural (SLEEC) concerns as rules, and on verifying that a system design adheres to them. However, to date, there is limited work related to the actual implementation and actuation of SLEEC-aware behaviours. Yet, we believe that operationalising SLEEC rules can enable responsible behaviour of autonomous systems and advance research on the topic. For this purpose, we propose an operational solution for ethical-aware autonomous systems. Specifically, we devise a principled approach, which we call CO-SLEEC (Context-Oriented SLEEC), connecting the normative setting of SLEEC rules to context-oriented programming (COP). CO-SLEEC promotes runtime adaptation and exhibition of context-dependent ethical behaviour through a modular and transparent design.

KEYWORDS

Ethical-aware systems; normative multi-agent systems; context-aware systems; SLEEC rules; context-oriented programming

ACM Reference Format:
Roberto Casadei, Martina De Sanctis, Gianluca Filippone, Sara Pettinari, Gian Luca Scoccia, and Nicolas Troquard. 2026. Leveraging Context-Oriented Programming to Implement Normative Rules in Autonomous Systems: 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/BLFA6735

1 INTRODUCTION

Autonomous and artificial intelligence (AI) systems interacting with or affecting humans with their behaviour come with associated risks that must be properly addressed. Namely, autonomous agents should transparently behave in accordance with recognised social and human values, and adapt ethically to different users in different contexts [3]. At the time of global AI policy frameworks [8, 9, 13] and regulations [2, 6, 14], the mounting sensitivity on these aspects is driving research on *ethical AI* [5, 7] and so-called *ethical-aware autonomous systems (EthicASs)* [1]. In this direction, Townsend et al. [11] propose a methodology to elicit *social, legal, ethical, empathetic, and cultural (SLEEC)* rules for multi-agent systems. An example rule of this kind is the following, which governs a robotic agent's response to the user requesting to open curtains in an assistive robotic scenario:

`WHEN` CurtainOpenRequest `THEN` OpenCurtains ⓪
`UNLESS` UserUndressed `IN WHICH CASE` (Refuse AND ExplainWhy) ①
`UNLESS` UserDistressed `IN WHICH CASE` (WarnUser AND OpenCurtains) ②

The *default rule* ⓪ prescribes that the robot has to open the curtains when asked. However, two *hedge clauses* are present to introduce context-dependent behaviour: if the user is undressed, the robot refuses and explains the reason ①, to account for privacy and cultural sensitivity; if the user is highly distressed, this clause is in turn overridden and the curtains are opened ②, prioritising emotional state and autonomy.

Recent research has provided ways for turning high-level normative principles into explicit, formalised, consistent sets of SLEEC rules [12, 15]. The decoupling of system designs from norms is useful, as it enables domain experts without programming competencies to define the norms, and it accounts for the separate evolution of norms and systems. However, there is currently a lack of contributions regarding the actual *operationalisation* (namely, the identification of mechanisms and procedures to turn principles into design and then implementations) of SLEEC rules in EthicASs.

We propose a solution to implement SLEEC rulesets in autonomous systems (e.g., multi-service robots). Since SLEEC rules ground normative concerns in specific contexts, we argue that their operationalisation can be achieved with programming abstractions that explicitly represent context. Accordingly, we develop *Context-Oriented SLEEC (CO-SLEEC)*, an approach mapping SLEEC rules to context abstractions, promoting their implementation following the *context-oriented programming (COP)* paradigm [4, 10]. This leads to a principled mapping and design pattern from SLEEC rules to COP constructs, and a reusable implementation of CO-SLEEC within the Robotic Operating System (ROS) framework.

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/BLFA6735

SLEEC element	COP concept
Specification	
Rule trigger	Event triggering a method call
Default rule obligation	Base method implementation
Hedge clause	COP layer
Hedge clause condition	Runtime context
Hedge clause obligation	Layer-specific method implementation
Runtime	
Rule evaluation	Layer activation
Rule trigger occurrence	Layered method dispatch

Table 1: Mapping of SLEEC elements to COP concepts.

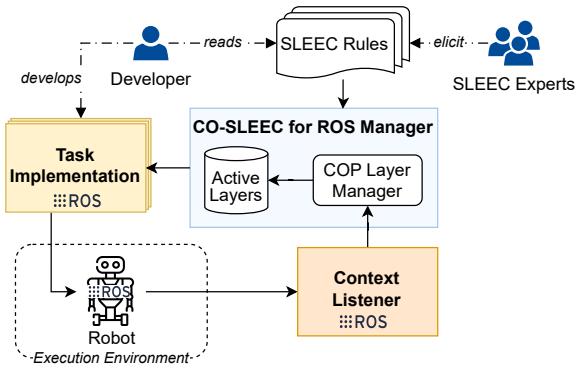

Figure 1: CO-SLEEC architecture.

2 THE CO-SLEEC APPROACH

CO-SLEEC consists of two main ingredients: (i) a principled mapping from SLEEC rules to COP abstractions, and (ii) a reusable design and implementation schema tailored to ROS-based systems.

2.1 Operationalising SLEEC Rules through COP

The core idea of CO-SLEEC is to exploit COP as a programming abstraction for implementing the context-dependent behaviour prescribed by SLEEC rules. A SLEEC rule specifies how a system should react to an event under different contextual conditions; similarly, COP enables behavioural variation through dynamically activated layers. Overall, each SLEEC rule is associated with a method (with multiple COP layer-bound implementations, each one implementing a different obligation), one event handler triggering the method, and a set of layers that regulate the selection of the correct method implementation variant as per the SLEEC semantics.

Table 1 maps SLEEC rule elements and COP concepts. The *trigger* of a SLEEC rule corresponds to an *event* that has to be captured to request the dispatch (*call*) of the method, associated with the rule, realizing the rule's obligation. The *default rule* obligation maps to the *base implementation* of the method associated with the rule (i.e., the one dispatched when no layers are active). Each *hedge clause* can defeat previous obligations, and is mapped to a distinct COP *layer*. The *hedge clause condition* is mapped to a predicate that is evaluated against the *runtime context* and used to drive proper layer activation. Finally, the *hedge clause obligation* is mapped to a distinct *implementation* of the method associated with the rule, labelled with the layer corresponding to the hedge clause. At runtime, SLEEC rules are continuously evaluated against the current context to determine the possibly *active hedge clauses*, i.e., the clauses whose obligation must be enforced when their rule trigger occurs. According to the SLEEC semantics [12], the active hedge clause of a rule is the last clause in the ordered sequence whose condition holds; hence, at most one hedge clause per rule may be active.

More formally, a system contains a finite set Γ of SLEEC rules, and a SLEEC rule $R_j \in \Gamma$ is of the form: `WHEN` C_0^j `THEN` O_0^j `UNLESS` C_1^j `IN WHICH CASE` O_1^j ...`UNLESS` $C_{n_j}^j$ `IN WHICH CASE` $O_{n_j}^j$. Given a runtime context χ, a hedge clause $hc_i^j = \langle \ell(R_j)_i, C_i^j, O_i^j \rangle$ is active iff:

$$\begin{cases} \chi \models C_1^j \wedge \cdots \wedge C_i^j \wedge \neg C_{i+1}^j & \text{if } i < n_j, \\ \chi \models C_1^j \wedge \cdots \wedge C_i^j & \text{if } i = n_j. \end{cases}$$

The layers labelled as $\ell(R_j)_i$, corresponding to the active hedge clause hc_i^j of each SLEEC rule R_j, are then activated in the COP runtime. When the trigger of a rule occurs, the method associated with the rule is dispatched, and the implementation bound to the active layer is executed (or the default one if no layer is active), ensuring that the system behaviour conforms to the intended SLEEC semantics.

2.2 CO-SLEEC for ROS-based Systems

We implement CO-SLEEC within the ROS framework, which structures robotic applications as distributed nodes communicating via publish/subscribe topics. Figure 1 illustrates an overview of the CO-SLEEC architecture. Two main roles are involved: *SLEEC experts*, who elicit domain-specific SLEEC rules, and the *developer*, who implements ROS nodes augmented with CO-SLEEC facilities.

The integration of CO-SLEEC into ROS is realized via an *Manager* module that bridges rule reasoning and robotic execution. This module supports the parsing and evaluation of SLEEC rules and manages COP layers, by providing the necessary ROS nodes, message types, and interfaces. A dedicated *Context Listener* node monitors contextual conditions and domain changes and triggers the evaluation of the relevant SLEEC rules. Based on these updates, the *COP Layer Manager* determines and activates the set of layers corresponding to the current context.

Robot missions are defined as collections of tasks, each implemented as a ROS node. When a task is constrained by SLEEC rules, its *implementation* consists of a default behaviour and multiple behavioural variants, associated with the rule's hedge clauses and corresponding COP layers. At runtime, the COP mechanism dispatches the implementation according to the active layer, enabling dynamic adaptation of robot behaviour in accordance with normative and contextual requirements.

ACKNOWLEDGMENTS

The authors acknowledge PRIN project "HALO: etHical-aware AdjustabLe autOnomous systems" (grant 2022JKA4SL), PRIN PNRR 2022 project "RoboChor: Robot Choreography" (grant P2022RSW5W), and the support of the MUR (Italy) Department of Excellence 2023–2027 for GSSI.

REFERENCES

[1] Marco Autili, Martina De Sanctis, Paola Inverardi, Mashal Afzal Memon, Patrizio Pelliccione, and Sara Pettinari. 2026. A reference architecture for ethical-aware autonomous systems. *Journal of Systems and Software* 235 (2026), 112749. https://doi.org/10.1016/j.jss.2025.112749

[2] Cyberspace Administration of China. 2023. Interim Measures for the Management of Generative Artificial Intelligence Services. https://www.cac.gov.cn/2023-07/13/c_1690898327029107.htm

[3] Virginia Dignum. 2025. Responsible AI and Autonomous Agents: Governance, Ethics, and Sustainable Innovation. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025*. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 1–2. https://doi.org/10.5555/3709347.3743508

[4] Achiya Elyasaf, Nicolás Cardozo, and Arnon Sturm. 2023. A framework for analyzing context-oriented programming languages. *Journal of Systems and Software* 198 (April 2023), 111614. https://doi.org/10.1016/j.jss.2023.111614

[5] High-Level Expert Group on AI. 2019, last update 31 January 2024. Ethics guidelines for trustworthy AI. https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

[6] Soler Garrido J, De Nigris S, Bassani E, Sanchez I, Evas T, André A, and Boulangé T. 2024. *Harmonised Standards for the European AI Act.* Technical Report. Seville (Spain).

[7] Brent Mittelstadt. 2019. Principles alone cannot guarantee ethical AI. *Nature machine intelligence* 1, 11 (2019), 501–507.

[8] OECD.AI Policy Observatory. 2019 updated May 2024. OECD AI Principles overview. https://oecd.ai/en/ai-principles

[9] Petar Radanliev. 2025. AI Ethics: Integrating Transparency, Fairness, and Privacy in AI Development. *Applied Artificial Intelligence* 39, 1 (Feb. 2025). https://doi.org/10.1080/08839514.2025.2463722

[10] Guido Salvaneschi, Carlo Ghezzi, and Matteo Pradella. 2012. Context-oriented programming: A software engineering perspective. *J. Syst. Softw.* 85, 8 (2012), 1801–1817. https://doi.org/10.1016/J.JSS.2012.03.024

[11] Beverley Townsend, Colin Paterson, T. T. Arvind, Gabriel Nemirovsky, Radu Calinescu, Ana Cavalcanti, Ibrahim Habli, and Alan Thomas. 2022. From Pluralistic Normative Principles to Autonomous-Agent Rules. *Minds and Machines* 32, 4 (Oct. 2022), 683–715. https://doi.org/10.1007/s11023-022-09614-w

[12] Nicolas Troquard, Martina De Sanctis, Paola Inverardi, Patrizio Pelliccione, and Gian Luca Scoccia. 2024. Social, Legal, Ethical, Empathetic, and Cultural Rules: Compilation and Reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 20 (March 2024), 22385–22392. https://doi.org/10.1609/aaai.v38i20.30245

[13] UNESCO. 2022. Recommendation on the Ethics of Artificial Intelligence. https://www.unesco.org/en/artificial-intelligence/recommendation-ethics Accessed on: February 2026.

[14] United States Government. 2022. Blueprint for an AI Bill of Rights Making Automated Systems Work for the American People. https://www.whitehouse.gov/wp-content/uploads/2022/10/Blueprint-for-an-AI-Bill-of-Rights.pdf

[15] Sinem Yaman, Pedro Ribeiro, Ana Cavalcanti, Radu Calinescu, Colin Paterson, and Beverley Townsend. 2025. Specification, validation and verification of social, legal, ethical, empathetic and cultural requirements for autonomous agents. *Journal of Systems and Software* 220 (Feb. 2025), 112229. https://doi.org/10.1016/j.jss.2024.112229