On the Impact of Fuzzy-logic based BDI Agent Model for Cyber-Physical Systems

Burak Karaduman[a,b,*], Baris Tekin Tezel[c], Moharram Challenger[a,b]

[a]*Department of Computer Science, University of Antwerp, Middelheimlaan 1, Antwerp, 2020, Flanders, Belgium*
[b]*AnSyMo/CoSys Core-lab, Flanders Make Strategic Research Center, Leuven, 3001, Flanders, Belgium*
[c]*Department of Computer Science, Dokuz Eylul University, Buca, Izmir, 35390, Turkey*

Abstract

Cyber-Physical Systems (CPS) interconnect embedded computing technologies into the physical world, forming a complex, multi-disciplinary, physically-effective system. However, interacting with the physical world brings unpredictability, as real-world events are uncertain and dynamic by its nature. Consequently, CPS must be capable of reasoning about these unpredictable situations and adapt their behaviour accordingly. Therefore, it is essential to enhance the intelligence of CPS to overcome this challenge effectively, leading to the development of smart CPS. As an approach, a reasoning mechanism can support the system to reason about its state and actions to handle unpredictable changes. In this study, we employ intelligent Belief-Desire-Intention (BDI) agents to achieve this objective. Nevertheless, traditional logic approaches based on crisp numbers, which are used in typical BDI agents, may not effectively handle uncertainty. Hence, the reasoning mechanism can be extended with fuzzy logic to deal with run-time uncertainties. Thus, the target system's suitable inputs, outputs and reasoning phases are fuzzified to indicate the impact of the fuzzy logic on the CPS. To this end, an architecture is provided to deploy BDI agents integrated with a fuzzy reasoning model for handling imprecise information and improving process control. The approach is validated and evaluated on a complex case study with a heterogeneous structure controlled by multiple agents, namely the "smart production system". The multiple modular and end-to-end experiments conducted reveal that, on the whole, the fuzzy BDI agent outperforms the classical one by up to three times, with only requiring approximately 10% more computation time.

Keywords: Software Agents, BDI Agent Model, Fuzzy Logic, Cyber-Physical Systems

1. Introduction

The rapid advancement of network systems and sensor technologies has brought about a new paradigm shift, namely Cyber-Physical Systems (CPS), posing new challenges for embedded systems. The merging of information processing and computation with communication and control has given rise to CPS, an advanced paradigm (Greer et al., 2019). This evolution expands embedded technology's capabilities to interact with the physical world through computation, communication, and control. It paves the way for the integration of the cyber and physical world as well as considering human factors. CPS holds the potential to greatly enhance various domains such as medical devices, vehicles, highways, robotic systems, and factory automation. These areas can benefit significantly from the remarkable capabilities offered by CPS.

[*]Corresponding author at Department of Computer Science, University of Antwerp, Middelheimlaan 1, Antwerp, 2020, Flanders, Belgium, Mobile Tel: +32 495 17 03 40

Email addresses: `burak.karaduman@uantwerpen.be` (Burak Karaduman), `baris.tezel@deu.edu.tr` (Baris Tekin Tezel), `moharram.challenger@uantwerpen.be` (Moharram Challenger)

Despite the numerous benefits provided by CPS, they still encounter various challenges, with one of the prominent ones being the presence of uncertainty originating from the physical environment. In this context, the traditional programming approaches that use classical logic become inefficient and burdensome to define logical states for uncertainty handling (Calegari et al., 2020). Moreover, consecutive unpredictable events can occur, requiring the system to interpret and adjust its configuration, such as speed, motion, and movement, as these systems are physically intensive. This brings the need for defining the CPS's operation range and behaviour, considering its constraints and thresholds based on its sensor, actuator and somatic components. Besides, time-sensitive systems may also require momentary decision-making solutions, providing explainable and interpretable white-box procedures and requiring fewer data to model the system. In addition, along with the logic-based approaches, machine learning methods can also provide long-term benefits for the system to lessen run-time uncertainty (Gheibi et al., 2021). However, they are usually intense data-dependent approaches and provide a black-box approach for which the results are not easy to explain. In this study, we focus on the former approach, considering further extensions with the sub-symbolic AI approaches. Therefore, enhancing CPS with a symbolic AI technique for short-term decision-making to handle run-time uncertainty becomes our goal. As one of the symbolic AI approaches, fuzzy logic plays a significant role in addressing the uncertainty that arises in the physical world within CPS (Calegari et al., 2020). Unlike classical logic, fuzzy logic incorporates the system's rule definitions, constraints, and thresholds, allowing for a more nuanced and flexible approach to handling uncertainty. This extension enables CPS to make more informed decisions and take appropriate actions in response to the dynamic and uncertain nature of the physical environment (Leitão et al., 2022).

Fuzzy logic is a mathematical model that works with imprecise data, which can rectify deficiencies by smoothing the output actions, widening the perceptual capabilities, and enhancing the reasoning mechanism. It also allows for rule definitions that can be easily added and modified when the system's component is changed or its operation/configuration parameters are altered. For these reasons, fuzzy logic has been preferred as an AI method and the basis for providing intelligence to achieve a smart CPS (Karaduman et al., 2022b,a, 2021b; Queiroz et al., 2022).

While fuzzy logic provides a way to deal with the uncertainty (Zadeh, 1983; Calegari et al., 2020), the multi-agent systems (MAS), especially the intelligent BDI agents, offer a high-level programming abstraction to deal with the complexity of the CPS, allowing the modular deployment of the intelligent software entities. Therefore, many studies in the literature have proposed benefiting from the abilities of multi-agent systems (Semwal & Nair, 2016; Karnouskos et al., 2018; Karaduman et al., 2022b) to deal with the challenges of complex systems such as CPS. Accordingly, MAS can be preferred as the basic paradigm to provide the smartness, decentralization, autonomy, and socialization of CPS (Leitao et al., 2016).

Mainly, MAS can increase the efficacy of CPS by furnishing fortified functionalities for production and automation (Leitão et al., 2018). The software agents can decide on re-configuring the control parameters, monitor the transition between processes, and observe human errors considering the system and human safety. Moreover, they can also decide on a suitable plan to preserve product quality and prevent damage during critical processes for the current context (Calinescu et al., 2020; Weyns, 2020). Specifically, one of the advantages of BDI-based MAS is the interoperability with theoretical approaches and applicability with various methods. Ultimately, the BDI agents define a devoted model that adopts a human-being mechanism. On the other hand, fuzzy-based approaches can be a way to deal with imprecise information, creating even more resembling human reasoning. Therefore, this study integrates and utilises these two approaches for the CPS, *fuzzy-logic-based BDI agents*.

The initial work of this study was achieved in (Tezel et al., 2016) and then refurbished for CPS in (Karaduman et al., 2021b) in which the fuzzy-BDI approach was introduced and evaluated using an abstract case study. The study (Karaduman et al., 2022a) revealed the applicability of the fuzzy-logic approach on a mobile system using a simple-reflex (SR) agent. Lastly, the study (Karaduman et al., 2022b) covered integrating a fuzzy-inference system (FIS) with a BDI agent using software artifacts. However, the designed FIS was not included in the BDI agents' reasoning mechanism.

The proposed approach has been evaluated using different scenarios in a complex, multi-phase and heterogeneous case study. This method can alleviate the concerns towards the run-time sustainability of the CPS. Furthermore, engineers can consider fuzzy logic as an alternative way to enhance the system under scrutiny and apply it during the design phase. The incorporation of this multi-logic technique constitutes the main contribution of this study, using a higher-level agent development language and addressing uncertainty more effectively.

As the CPS paradigm comprises physical and cyber parts, a production line case study was considered using a physical prototyping technology such as LEGO. It provides a concrete case study and allows the creation of physical segments of a CPS using somatic components that are LEGO bricks and parts. LEGO has emerged as a popular technology in the literature for constructing tangible applications in a range of CPS studies. It has been utilized in diverse research works, such as software development in CPS (Schoofs et al., 2021; Semwal & Nair, 2016) and industrial operations' simulation (Ltaief et al., 2022; Yalcin et al., 2021). In the context of our study, we leveraged LEGO as a foundational technology to create a concrete case study. This case study aimed to showcase the effectiveness and validation of our proposed approach, which integrates fuzzy logic and agent-based principles into a CPS framework. By utilizing LEGO, we were able to implement a practical demonstration of our enhanced CPS, providing tangible evidence of the feasibility and potential of our approach.

The focus of this paper is to thoroughly investigate the latest advancements in the field, including the current directions of state-of-the-art approaches, thereby resulting in the introduction of several novel contributions. These contributions include:

- **Multi-Logic Technique:** We propose the utilization of a multi-logic technique that combines fuzzy logic and BDI agent approaches for CPS. This technique aims to address uncertainty by fuzzifying the reasoning mechanism of BDI agents, encompassing perceptions, beliefs, plan selection, and actions. By integrating fuzzy logic into the BDI agent framework, we enhance the agent's ability to reason and make decisions in uncertain environments.

- **Enhancement of Reference Architecture:** Building upon our previous reference architecture (Karaduman et al., 2023b), we extend and enhance it to accommodate logic-based improvements. Our goal is to provide a framework that is adaptable, extensible, and suitable for the integration of BDI agents, taking into account the diverse design options that can be contributed by a broad audience of researchers and practitioners.

- **From Feasibility to Effectiveness Validation:** We demonstrate the feasibility, integrability, applicability, and effectiveness of our fuzzy-logic-based distributed BDI agent approach. To validate our approach, we utilize a multi-phase, heterogeneous, and complex case study: a smart production line system. Through modular and end-to-end experiments, we evaluate the performance and time metrics across multiple scenarios. The results are analyzed using statistical methods to provide a comprehensive assessment of our approach.

These contributions collectively contribute to advancing the understanding and application of fuzzy-logic-based distributed BDI agents in complex systems. By addressing uncertainty and enhancing the reasoning capabilities of agents, we provide a valuable approach to decision-making in dynamic and uncertain environments. In this regard, we aim to answer the following research questions.

1. **R.Q.1 (Feasibility)** How can we enhance and deploy the BDI agents using fuzzy logic to increase their capacity to deal with run-time uncertainty?

2. **R.Q.2 (Integrability)** How agents' sensing, reasoning, plan selection and execution phases can be *fuzzified* and bind to the device-specific software?
3. **R.Q.3 (Applicability)** How the proposed approach can be implemented and deployed on a concrete physical setup?
4. **R.Q.4 (Effectiveness)** Are fuzzy-BDI agents effective for the uncertainty of agent-based CPS?

The rest of this paper is organised as follows: Section 2 presents background information for this work. Section 3 explores and dissects previous studies conducted in the field, specifically focusing on identifying the research gap within the existing literature by means of a comparative manner. Section 4 introduces the fuzzy procedural reasoning model, the enhanced and extended fuzzy BDI architecture, and presents the smart production case study. Section 5 provides a concrete implementation of the fuzzy-BDI agents conforming to the proposed architecture, including enhancements and extensions, showcased in the validating complex case study. Section 6 elaborates on the empirical evaluation and provides detailed results. Section 7 discusses the study's results. Finally, in Section 8, we conclude the paper by summarizing the main findings and contributions. We also discuss the limitations of our paper and propose directions for future research.

2. Background

This background section provides adequate details on the interrelated topics of cyber-physical systems, multi-agent systems, and fuzzy logic, referring to their basic concepts and definitions.

2.1. Multi-Agent Systems

The concept of agentification of a system provides an enhancement to the CPS by deploying intelligent software entities which can achieve their goals by dealing with distributed complexity and heterogeneous structure (Pico-Valencia & Holgado-Terriza, 2018). These software entities (agents) should work cooperatively with other agents in the same heterogeneous environment and in other systems to achieve a global goal where a single agent is not enough to reach this global goal. Specifically, rational BDI agents use a cognitive approach and contain a representative physical world model to develop plans and make decisions using a reasoning mechanism.

Extensive research and development efforts have been dedicated to multi-agent systems, with a focus on achieving modularization for dynamic systems, decentralization for distributed systems, autonomy for production, and re-usability to facilitate the future development of physical systems(Merdan et al., 2011). These systems can be widely adopted to enhance various functionalities in complex and distributed systems by incorporating techniques from diverse application domains. By leveraging the capabilities of multi-agent systems, these complex systems can be effectively improved and adapted to meet specific requirements.

The software agent paradigm raises the abstraction level of designing, programming and developing applications using these autonomous software entities, which can be deployed in an environment. These agents can autonomously achieve their goals by collaborating, interacting with the environment, and communicating with other systems. The usefulness of these agents is augmenting the systems' capabilities suited to operate in a dynamic and unpredictable environment. The agents can react to phenomena expected to be perceived, managed and controlled by the CPS to sustain its operation.

The pursuit of a multi-agent system is to find a solution that can be a global problem in the space of that system. Therefore, once a system deploys autonomous, goal-oriented, adaptive, and reactive agents, these agents can produce specific solutions for the upcoming problems that emerge around the agents' environment or internally. They behave towards collecting and sharing information to complete their goals. Once a goal is achieved, they select the next plan that leads them to their next goal.

2.2. BDI Agents for CPS

Logic-based reasoning mechanism studies root back to the late 50's (McCarthy & Shannon, 1958) with the idea of commonsense reasoning. Over the years, logic-based techniques have evolved, considering deduction on first-order logic. Other approaches have also been built upon the inductive and abductive reasoning principles (Calegari et al., 2021). After this, some languages, such as the Prolog, have been engineered on deductive principles.

Jason (Bordini et al., 2007; Boissier et al., 2020), as a prolog-like agent programming language, is based on Agentspeak (Rao, 1996). It incorporates an environment layer that allows seamless integration with Java applications. Furthermore, Jason utilizes the widely recognized Procedural Reasoning System (PRS) architecture mentioned in the study (Georgeff & Ingrand, 1989). It intelligibly represents the belief-desire-intention (BDI) model mentioned in (Rao & Georgeff, 1998). By adopting the BDI model, Jason provides a structured framework for agents to reason about their beliefs, desires, and intentions, facilitating intelligent decision-making and action coordination. BDI is a kind of model logic for implementing intelligent agents. When multiple BDI agents establish communication to achieve a global goal, multi-agent systems emerge.

The BDI agents maintain continuous observation of their environment. They monitor and perceive changes in the environment and respond accordingly by executing plans and considering contextual information. This dynamic interaction between the agent and its environment allows BDI agents to adapt their behaviour and decision-making in response to environmental changes, ensuring effective and timely reactions to evolving conditions. According to their reasoning mechanism, it can behave re-actively and goal-oriented based on their plan structure. The plan structure defines the system's behaviours and actions that can be adapted for the same input. The cognitive ability emerges when the agent can persistently reach a goal using its reasoning and simultaneously react to the instant changes to sustain its operation. These reactions are selected according to the agent's mental state and the current context of its perimeter. Simple reflex agents, in contrast to BDI agents, are the most basic type of agents that operate purely in a reactive manner, focusing on the behavioural level of abstraction. These agents respond to specific inputs by executing predetermined actions without engaging in any form of planning or taking into account the agent's internal state or long-term implications. Therefore, the simple-reflex agents may be summarized as a subset of the BDI agents from the logic and reasoning perspectives. As previously mentioned, the BDI model consists of three mental components: beliefs, desires, and intentions.

Beliefs encompass the information stored in the agent's knowledge base, including information about itself, other agents, and the surrounding environment.

Desires epitomize all goals that hold the potential to become successful states. Any desire can serve as a catalyst for the actions of the agent.

Intentions pertain to the chosen course of action or plan that the agent decides to pursue. After a deliberation process, the agent selects the most appropriate plan to achieve a given goal and transforms it into an intention.

Beliefs serve as the foundation for agent reasoning. In each cycle, Jason aims to find a suitable plan for a given goal by taking into account the contextual information or triggering events stowed in beliefs. By serving as indicators, these triggering events signal environmental changes that may necessitate the agent's attention or engagement in some form of action.

As previously stated, BDI agents demonstrate their commitment to goals by executing plans. Furthermore, BDI agents have the ability to simultaneously reason about events and react accordingly. During each reasoning cycle, they check the preconditions of plans, also known as the application context, by utilizing the information stored in the belief base. When there is a change in the context, the agent responds by applying a plan that is suitable and aligned with the updated application context. For the validation case study mentioned, the Jason BDI agent programming framework was utilized. It provided an environment for agent development, encompassing the essential features and characteristics of the software agents paradigm mentioned earlier.

Generally speaking, the core idea of CPS is inter-playing the cyber and physical components to make them operate and influence each other in an environment where events occur over time connected in a network. While a CPS is interacting with the physical world, there is a phenomenon that has to be answered by the system. The cyber part causes the physical component of the system to change state after it receives feedback from sensor input. The cyber part then adapts itself according to this change. Therefore, the design of smart CPS requires a reasoning mechanism to deal with the unpredictable events during the system's run-time (Tepjit et al., 2019).

In this regard, Multi-Agent System technology can address some challenges (Leitao et al., 2016) in the design of CPS (Karnouskos et al., 2019), such as distributed topology, integrity, modularity and collaboration. Accordingly, intelligent BDI agents are required to establish smart CPSs, considering autonomous, interactive and adaptive behaviour. Moreover, a CPS should be aware of its context, specifically the environment in which it is deployed. According to the contextual changes, it should be able to decide the goals and adaptation plans for undesirable events that emerge because of uncertainty. Therefore, the BDI agents can be deployed on the cyber side to manage the CPS (Karaduman et al., 2022b,a; Ltaief et al., 2022; Karaduman & Challenger, 2022; Can et al., 2022; Karaduman et al., 2023a; Yalcin et al., 2021) promoting the CPS to the sCPS, including logic-based enhancements. In the following subsection 2.3, fuzzy logic to deal with uncertainty and enhancement for the BDI agents is mentioned in detail.

2.3. Fuzzy Logic for Handling Uncertainty

Zadeh (Zadeh, 1996) has introduced the fuzzy theory to handle problems that emerged because of imprecise information and uncertain events. A fuzzy set can be seen as an extension of a classical notation set. It is a collection of objects that are defined using a membership function. The membership function is responsible for assigning a membership degree to each object, indicating the degree to which it belongs to a particular class. The membership degree is typically represented as a value between zero and one, where zero represents no membership or complete non-belonging, and one represents full membership or complete belonging to the class. By using a membership function, fuzzy sets provide a more flexible and nuanced way of representing uncertainty and partial membership compared to classical sets (Kinay & Tezel, 2022). The definition of a fuzzy set is given by Equation 1 (Cuevas et al., 2022; Castillo et al., 2016).

Definition 1. *A fuzzy set is a pair in the universe X and $\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) | x \in X\}$ where the element $x = (-\infty, \infty)$ is denoted with the aid of its membership function $\mu_{\tilde{A}} : X \to [0,1]$ using to map each element $x \in X$ of the fuzzy set that has a value in the range of interval* $[0,1]$.

In 1973, Zadeh published his influential paper (Zadeh, 1973), presenting a novel approach to analyzing complex systems by capturing human knowledge using fuzzy rules. A fuzzy rule typically follows the form:

R : if x is A, then y is B

Here, A and B represent linguistic values defined by fuzzy sets on the discourse universes X and Y, respectively. These fuzzy rules provide a way to express relationships and reasoning in a more flexible and human-like manner, allowing for the representation of imprecise or uncertain information. By employing fuzzy rules, complex systems can be effectively analyzed and modelled, incorporating human expertise and knowledge into the decision-making processes.

In recent years, there has been an increased utilization of fuzzy set theory in the fields of complex systems theory and decision-making processes. Fuzzy set theory has gained recognition as a valuable tool for dealing with uncertainty, imprecision, and complexity in various domains. Its ability to handle vague or incomplete information and represent linguistic terms has made it particularly useful in modelling and analyzing complex systems.

Fuzzy set theory provides a framework for capturing and manipulating imprecise or uncertain data, allowing for a more nuanced and flexible representation of information. This has proven beneficial in complex systems theory, where traditional mathematical models may fall short in capturing the intricacies and uncertainties of real-world systems. Fuzzy set theory enables researchers and practitioners to incorporate subjective and expert knowledge, as well as handle data with inherent fuzziness or ambiguity.

Moreover, fuzzy set theory has found application in decision-making processes. It offers a means to model and evaluate preferences, uncertainties, and trade-offs in decision-making problems. By using fuzzy logic and fuzzy inference, decision processes can be enriched with the ability to handle imprecise inputs, make approximate reasoning, and generate more human-like and interpretable outputs.

Overall, the increased adoption of fuzzy set theory in recent years reflects its growing recognition as a valuable approach for addressing the challenges posed by complex systems and decision-making under uncertainty.

Fuzzy logic applications have become widespread in many areas, from finance, marketing, and other decision-making problems to microcontroller-based and large-scale process control systems (Tezel & Mert, 2021). Subsequently, the fuzzy theory has gained much attention from academia and industry for enhancing decision-making, solving the problems that emerged in control and automation engineering and increasing the impact of intelligent systems (Alcalá-Fdez & Alonso, 2015) during the past few decades. Fuzzy-based approaches provide advantageous augmentations for dynamic systems to achieve smooth measures instead of traditional logic.

Fuzzy logic offers a means to emulate the continuous nature of human decision processes, thereby improving upon methods based on binary logic. In dynamic environments, external and instantaneous changes can affect the system, requiring it to stabilize and adapt accordingly. Unlike conventional methods, fuzzy theory provides a platform-independent mathematical model that can be applied in CPS. Its design methodology revolves around the representation of uncertainty and reasoning, enabling precise decision-making outcomes that enhance system stability and mitigate uncertainty.

By incorporating fuzzy logic into CPS, it contributes to transforming them into smart CPS. Fuzzy logic serves as an enhancement by enhancing the decision-making process, facilitating action executions, improving reasoning capabilities, and aiding in plan selection under run-time uncertainties. This integration of fuzzy logic into intelligent BDI agents, known as fuzzy-BDI agents, aims to enhance their overall functionality in dynamically changing environments. The utilization of fuzzy logic empowers these agents to make more precise and adaptable decisions, allowing them to effectively cope with uncertainties as they arise.

In summary, the integration of fuzzy logic theory into software BDI agents brings about improvements in decision-making, action execution, reasoning, and plan selection capabilities, particularly in scenarios where uncertainties are present during run-time. This integration paves the way for more robust and intelligent CPS systems.

2.4. LEGO Technology for Prototyping CPS

The integration of MAS and CPS opens up possibilities for employing agents in various domains related to CPS (Karnouskos et al., 2019; Leitao et al., 2016). When agents gain control over the physical components of a CPS, they can contribute to solving cyber-related problems by reasoning at a high-level abstraction. However, in certain situations, creating an actual CPS may not be feasible due to factors such as cost, safety concerns, or the size of the planned system. In such scenarios, it becomes essential to downscale the intended CPS while preserving its functionality, accuracy, and goals. To achieve this, the utilization of a composable and user-friendly technology can be beneficial in emulating the existing system, offering re-usability that facilitates the modification or creation of another system. This can be achieved by decomposing the current system or directly incorporating new materials. LEGO technology presents an option for constructing a scaled-down version of the CPS. By integrating LEGO technology's sensing and actuation components into the hardware interface of the embedded hardware system and utilizing LEGO bricks/parts for the creation of the physical plant, a functional CPS can be readily implemented. This approach allows for a cost-effective and safe way to mimic and experiment with CPS functionalities, making it a valuable tool for development and testing.

In summary, integrating MAS and CPS enables the utilization of agents in various CPS-related domains. When it is not practical to create a full-scale CPS, scaling down the system using composable technologies like LEGO allows for the construction of a functioning and adaptable CPS model. The interested readers may find the process model of building a CPS plant using the LEGO technology in (Karaduman et al., 2021a).

3. Related Work

In this section, related work of the study is given. The study (Calegari et al., 2020) discusses logic-based technologies for intelligent systems by giving state-of-the-art from various perspectives. They also mention that the logic approaches lay at the core of symbolic AI and the centre of the many agent-based technologies. Notably, the Prolog language is one of the most exploited languages in AI applications. (Dawson et al., 1996). Moreover, two of the mentioned reasoning approaches and techniques, the BDI agent paradigm and fuzzy logic, are underlined as practical approaches, especially for manufacturing technologies and industrial processes that suffer quantitative data about I/O operations. Accordingly, in this study, we preferred to enhance the Jason BDI, a Prolog-like framework with fuzzy logic. Lastly, some studies such as (Esfahani & Malek, 2013; Fredericks & Cheng, 2015; Ramirez et al., 2012; Fredericks et al., 2014) do not consider applying a multi-logic approach (Calegari et al., 2020, 2021) and deployment of the software agents. They focus on the user-level requirements specification (Cheng et al., 2009; Fredericks et al., 2014; Fredericks & Cheng, 2015), that conceptually and formally mention uncertainty in the CPS using the possibilistic perspective. However, they have inspired us for our system-level current and future works. We then have seen the necessity of the subtle shift from the single logic approaches (Cheng et al., 2009; Fredericks et al., 2014; Fredericks & Cheng, 2015) to multi-logic augmentations (Calegari et al., 2020, 2021) using a fundamental paradigm such as the BDI agents. Moreover, we have also considered the current challenges of (Karnouskos et al., 2020; Leitao et al., 2016) complex process control and run-time uncertainties of the CPS, in addition to the need for autonomy, distributed deployment, reactivity, goal-orientedness and collaboration. Thus, we have pertained our scope to the fuzzy-BDI agents and then carried out our work by enhancing the BDI agents based on an integrated and extensible architecture.

The studies in the literature have similar approaches and disjoint methods that can be divided and correspond to some categories: simple-reflex and BDI agents.

3.1. Simple-reflex agent-based studies

The study (Rocha et al., 2019) mentions the simple-reflex agents and fuzzy-inference system (FIS) at the cloud level. Their architecture concentrates on the social ability of the agents to find the cheapest. In contrast, they use fuzzy logic at the cloud level for algorithm selection; we use fuzzy logic at the edge level to tackle the run-time uncertainties that emerge in the physical world. Their study also does not mention how they integrated the FIS and the behaviours of the simple-reflex agents in detail. However, their work also inspires us for the possible future outcome of extending the fuzzy-enhanced proposed architecture to the cloud level. Lastly, they evaluate their work in a simulation environment.

(Peres et al., 2017) mentions that the manufacturing systems of Industry 4.0 should have distributed data acquisition and analysis techniques that support run-time decision-making and be self-adjustable to mitigate the effect of unwanted events. Their study utilizes simple-reflex agents to exemplify their approach.

(Queiroz et al., 2022) utilizes fuzzy logic to create a recommendation system to support the designers in selecting data analysis tasks and computation layers in design time. Parallel with their vision, we applied the fuzzy logic enhancement for the BDI agents' run-time reasoning.

In study (Gomes et al., 2020), fuzzy logic is used to classify the various sensor inputs. However, their approach includes no complex actuation or physical process execution. Moreover, they use a loosely coupled approach to gather sensor data from the microcontrollers using network protocols. As can be seen, some studies applied or envisioned using fuzzy logic with the simple-reflex agents for specific tasks such as design-time recommendation, sensor data classification and simple on/off control. We aim to shed light on the impact of the fuzzy-logic-based BDI agents on the CPS. In the following subsection, the related studies which focused on BDI agents without fuzzy logic are given to indicate the rationale for proposing an integrated architecture before applying the fuzzy-logic enhancement to the BDI agents underlined.

3.2. BDI agent-based studies

Wei & Hindriks (2012) presented an architecture for agent-based cognitive robots using a loosely coupled approach. Their approach was demonstrated using a single BDI agent without any enhancement. In this study, we are motivated to benefit from multiple BDI agents enhanced with fuzzy logic.

Pantoja et al. proposed an architecture named ARGO (Pantoja et al., 2016), considering a case study with a single agent without applying the fuzzy logic. They provide expansion hardware to dilate the sensor/actuator components, causing a burden in terms of extra cost, power consumption and hardware design complexity. Therefore, a fully integrated architecture and embedded hardware with an extended interface may be a better solution.

In their work, Wesz (2015) developed a tool that establishes a connection between Jason agents and ROS (Robot Operating System). Their study focused on creating a simulation environment that allows for the programming of single-agent mobile robots. However, their work did not specifically address the deployment of agents to embedded devices or incorporate fuzzy-logic enhancements.

Fichera et al. (2017) presented a framework called PROFETA, which implements AgentSpeak using Python. Their work offers an integrated solution similar to other proposed architectures, but it falls short in terms of providing a complete agent-oriented programming environment like Jason. Additionally, the study does not address physical deployment on embedded devices or incorporate fuzzified decision-making techniques. Furthermore, the study provides no actual deployment of the agents to an embedded device or fuzzified decision-making. Similarly, (Vachtsevanou et al., 2023) proposes a constraint and relegated version of Jason integration to the low-power embedded devices for the IoT domain using a simple light switch on/off control case study with C++. Their study does not utilise fuzzy logic or any other intelligence technique combined with the BDI. Moreover, their work requires further evaluation of the deliberation overhead on complex case studies for the CPS domain, but it motivates us to apply for our enhancements and extensions on low-power energy-efficient devices using fuzzy logic and conforming to our reference architecture.

Menegol et al. (2018b,a) aims to integrate the Jason and ROS. However, they utilized an additional programming language and an extra thread to establish wireless communication. Moreover, they use a simulation environment while we aim to augment the reasoning mechanism using fuzzy logic, considering an integrated and layered architecture where we can deploy the agent mechanism and low-level control on the same hardware without requiring extra board, programming language or thread mechanism.

Semwal & Nair (2016) propose a prolog-based agent framework called Tartarus. They deploy their agents into Lego NXT devices following a loosely coupled approach.

As can be noticed, so far, the studies which utilize the BDI agents mainly focused on a loosely coupled architecture without considering the loss of autonomy and its logic-based augmentations. Therefore, a tightly coupled architecture should be followed to deploy the BDI agents to eliminate the case of losing the agents and their enhancements. Moreover, some studies only focused on a single agent and simulation environments. Thus, we validated the effectiveness of our method on a complex, multi-stage and heterogeneous multi-agent system. In the following, the related works that implemented their BDI agents with fuzzy logic are introduced.

The fuzzy logic and the software agents integration have roots back nearly two decades ago(Flake et al., 1999; Long & Esterline, 2000; Shi & Xu, 2009; Shen et al., 2004; Tezel et al., 2016). However, these theoretical and conceptual studies have not gained enough maturity and concreteness (Rosin et al., 2022) to be applied to the cyber-physical systems domain to tackle the run-time uncertainties by indicating the integrability, applicability, feasibility, usability and effectiveness of the method-based on a fuzzy-logic enhanced architecture. Therefore, in this study, we aim to fill this gap. Thus, the following studies are discussed comparatively to conform to this aim.

Alaya et al. (2017) proposes a method to deploy multi-agent systems based on an architecture offered by Lee et al. (2015). They mention that logical approaches such as neural networks and fuzzy logic can be used for decision-making with cognitive agents. However, their approach does not mention how the agent and fuzzy logic were integrated, mainly how the reasoning mechanism for low-level control was applied and achieved in detail. Also, their study does not evaluate the cognitive agents for decision-making on a complex CPS.

Muto et al. (2021) provides a framework, namely CHANS, to simulate the small-scale farmer BDI agents using the fuzzy logic similar to this and our previous work(Karaduman et al., 2022b,a, 2021b). While their focus is decision-making for farming simulations, we aim to provide insights for the complex CPS to deal with the run-time uncertainty, considering the usability, applicability, integrability and evaluation based on the proposed architecture. They also stated that their work has no detailed explanation of the fuzzy logic and the BDI integration.

In a similar study (Rodriguez-Ubeda et al., 2015), the reasoning cycle of Jason was extended with the fuzzy logic to control a conceptual and simple sprinkler system for irrigation such as in a prior work of us (Karaduman et al., 2021b, 2022b). However, their study only focused on the input and output control without considering the fuzzy-logic-based plan selection, usability of the method, and evaluation of the approach as well as giving the applicability and integrability details of their implementation.

Xiaochao et al. (2019) proposes an agent-based simulation for combat simulation tests using the fuzzy BDI approach to provide a basis for the content change. Correlated with our previous outcomes (Karaduman et al., 2022b), they showcase their capacity to modify their membership functions in line with the belief states. However, their study does not mention any plan selection based on the fuzzy membership degrees and action triggering. Moreover, the study is constrained to a 2D simulation environment. Eventually, a more complex case study, which consists of multiple components and processes, is required to indicate the effectiveness of the fuzzy-BDI approach for CPS.

Cruz et al. (2021) provides formal definitions for the fuzzy BDI semantics similar to (Chen, 2015). They also suggest that the fuzzy membership degrees can be used for plan selection, as we did both in this work and our previous studies (Karaduman et al., 2021b, 2022b). (Garrab et al., 2017) proposed using the fuzzy-logic controller and BDI agents to reduce energy consumption by tuning the power use of the home appliances as a similar study. However, their work does not mention how they integrated the BDI agents and the fuzzy-logic controller created in MATLAB. Moreover, they only focused on reducing energy use using fuzzy logic, where the decision-making is only given to reducing the operation power of the static entities. However, as the CPS is a physically intensive domain with complex and multi-stage processes, uncertainty is mostly expected in the somatic system's dynamic components, modules and parts during the operation. Therefore, this also raises the need to define an integrated reference architecture first, then enhance the BDI agents with the fuzzy logic, and lastly deploy them on the embedded hardware in a tightly coupled manner. In this way, both the fuzzy logic enhancement and the BDI reasoning mechanism work inside the same cyber container.

In study (Ben Mekki et al., 2016), a loosely coupled fuzzy logic using MATLAB, such as in (Garrab et al., 2017), and BDI application was studied. Their work only focuses on supply chain monitoring for the economic efficiency of the enterprises.Specifically, they discussed that their study only covers the economic aspects while we devised our research method considering the cyber-physical systems.Their work also does not evaluate the fuzzy-logic enhancement of the BDI agents comprehensively and comparatively, such as the logic (Vu et al., 2013; Karaduman et al., 2021b) or any other logic-based approach. Therefore, the method's usability is not effectively exhibited. The integration of the fuzzy logic with the BDI agents' reasoning mechanism was also not demonstrated, which relegates the replicability and usability of the technique to an abstract form.

Vu et al. (2013) compares the crisp set and fuzzy set in an agent-based simulation of a 2D and two-player soccer game using the BDI agents. Although their study was realised in a 2D simulation environment less complex than the real-world applications, we share a similar evaluation approach to display the impact of the fuzzy logic, which is mentioned in Section 6. In addition, the integration between the reasoning mechanism, plan selection and fuzzy logic is also not evident in their study, as mentioned in Section 4.

Othmane et al. (2018) presents the CARS, a spatiotemporal agent-based recommender system based on the BDI agents. Using both position and time variables, they simulate a 2D traffic network in the NetLogo. Their work is similar to our previous study, in which we used temporal information and hedge modifier to arrange the membership function in study(Karaduman et al., 2022b). Their method can also be future work for us. Still, at first, we aimed to reveal the impact of the fuzzy-logic-based BDI enhancement on the CPS domain to pave the way for further symbolic and sub-symbolic augmentations (Calegari et al., 2020) considering spatial and temporal information based on the reference architecture (Karaduman et al., 2023a). Nevertheless, the 2D simulation environment may not be enough to tackle the uncertainties in the 3D physical world. Therefore, a concrete, heterogeneous and complex case study may conduct the expected impact of the fuzzy logic on the CPS.

Morris & Ulieru (2011) proposes an abstract and conceptual BDI architecture based on the neuro-fuzzy paradigm for virtual reality games using body sensors. Although their work has similar architectural layers to ours, we define the middleware layer as creating loosely coupled interactions to establish a connection with external services. At the same time, they describe it at the same level as the API. From our perspective, the middleware and API layers should be separated as the middleware may contain different APIs for different hardware interfaces and embedded hardware preferences by creating the device-specific and device-independent layers, considering the requirements of the CPS domain. Moreover, their study does not mention integrating this architecture with any BDI development framework, concrete form of their architecture, application of their proposed method, proof-of-concept, and evaluation of the approach. Yet, their work inspires us to extend our work with the neuro-fuzzy systems conforming to our proposed architecture.

In the study (Rosales et al., 2017), Type-2 fuzzy sets were utilised with the two BDI agents to simulate the human and machine interaction for improving the learning process of the visitors in a museum's room. They only fuzzified the perceptions of their agent using the Jason framework and did not apply any approach for fuzzy logic-based plan selection and execution. In their work, the integration of the fuzzy logic and the reasoning mechanism of the BDI agents is not apparent, nor is the deployment of the agents to the target systems. Moreover, their study lacks implementation details that narrow the method's reproducibility. In contrast to their controlled environment which is a museum room, a complex and heterogeneous system, such as a production line with many process phases, may be prone to more uncertainty as there are more dynamic parts and products. Lastly, their focus is human interaction, so they evaluated their work based on ethnographic research by taking notes, which might not be a suitable approach to assess cyber-physical systems.

A relevant reader can find additional information about the prior studies related to this work in the following sources. In two of those works, (Yalcin et al., 2021; Schoofs et al., 2021), it was desired to follow an integrated approach to demonstrate the implemented case studies. In the study (Schoofs et al., 2021), simple-reflex agents were used to create a multi-robot case study. The study (Schoofs et al., 2021) was expanded in (Can et al., 2022) using the mobile robots, and then a fuzzy-inference system was created in study (Karaduman et al., 2022a) to enhance the simple-reflex agents. Moreover, in study (Yalcin et al., 2021), SPADE[1] and Python-based LEGO API was preferred to build a production line, then it was extended in the study (Ltaief et al., 2022). In another previous study, we introduced a fuzzy-logic-based fan controller system with a single Jason BDI agent (Karaduman et al., 2021b), heeded by the preliminary work (Karaduman et al., 2022b), that employed the fuzzy-inference system using the CArtAgO artifacts (Ricci et al., 2011) which combined with the Jason BDI agents. Table 1 summarizes the mentioned related works.

In the literature, the deployment of software agents into embedded systems, formal approaches for fuzzifying the BDI agents and conceptual architecture approaches have been studied. Moreover, most of the related work has been explicitly reproduced for the CPS in our earlier works (Karaduman et al., 2021b, 2022b,a). Ultimately, to the best of our knowledge, they have not offered an effective enhancement using fuzzy-logic-based design architecture to apply it to the multi-agent systems by providing a concrete, complex and multi-stage validating use case in a feasible integrated/tightly coupled manner. In the following section, fuzzy BDI agents for CPS are introduced.

4. Fuzzy BDI agents for CPS

This section introduces the proposed architecture and fuzzy Procedural Reasoning Model to deploy BDI agents with a fuzzy reasoning model to deal with imprecise information and run-time uncertainties during the process control.

4.1. Motivating Example

In this study, we fuzzified the various agents' monitoring and execution phases. We also benefited from the fuzzy membership degrees for the plan selection phase, considering the rules defined in the design time contained in the agents' knowledge bases. Moreover, we utilized membership functions and a temporal approach (Whittle et al., 2009) for uncertainty mitigation. Typically, an agent monitors the environment through its sensors. However, the classical BDI agent development approach depends on the conventional logic where the sensor data is imprecise mainly because of the environmental noise. In addition, when the diversity of the data is increased, such as a colour sensor that consists of R (Red), G (Green), and B (Blue) values, it becomes a burden or unlikely to achieve the engineering design, also considering the time-consumption and cost. In a classical implementation, these R, G, and B values refer to some integer value that specifies a few colours, relegating the range of variety by creating information ambiguity, e.g., red and purple are represented by integer 5.

[1]https://spade-mas.readthedocs.io/

Table 1 Comparison table of the related works for integrating and deploying the enhanced BDI agents into CPS.

Publication	Framework	Multi-Agent	Reasoning	Enhancement	Physical Environment(PE) Simulation Environment(SE)	Application Field	Tightly Coupled	Way of Use
Arokiasami et al.	SOIFRA/JADE	✓	SR	✗	PE and SE	Obstactle Detection and Avoidance	✗	Motion
Schoofs et al.	JADE	✓	SR	✗	PE	Multi-robot	✓	Motion
Can et al.	JADE	✗	SR	✗	PE	Robot Navigation	✓	Robot Control
Ltaief et al.	JADE	✓	SR	✗	PE	Production Line and Robotic Arm	✓	Process Control
Yalcin et al.	SPADE	✓	SR	✗	PE	Production Line	✓	Process Control
Rocha et al.	JADE	✓	SR	Partial	SE	Production Line	✗	Algorithm Selection
Peres et al.	JADE	✓	SR	Conceptual	N/A	Production Line	✗	Data Analysis and Self-adjustment
Queiroz et al.	N/A	✗	N/A	Fuzzy Logic	PE	Electric Machine	✗	Recommedation System and Data Analysis
Gomes et al.	JADE	✓	SR	Fuzzy Logic	PE	Workplaces	✓	Recommendation System and Data Analysis
Wei and Hindriks	GOAL	✗	Belief and Goal-oriented	✗	PE	Robot Navigation	✗	Robot Control
Karaduman et al.	JADE	✗	SR	Fuzzy Logic	PE	Robot Navigation	✓	Robot Control
Karaduman et al.	JaCa	✓	BDI	Fuzzy Logic	PE/SE	Fan Controller	✓	Speed Adjustment
Pantoja et al.	Jason/ARGO	✗	BDI	✗	PE	Collision Detection	✗	Motion
Wesz and Meneguzzi	JaCaROS	✓	BDI	✗	SE	Mobile Robot	✗	Robot Control
Fichera et al.	PROFETA	✗	BDI	✗	PE	Mobile Robot	✓	Motion & Palet Control
KC et al.	Jason+ROS	✗	BDI	✗	PE	Social Robots	✗	Voice
Vachtsevanou et al.	Jason	✓	embedded-BDI	✗	PE	Light Automation	✓	On/Off Control
Menegol et al.	Jason+ROS	✓	BDI	✗	SE	Human-Robot Interaction	✗	Robot Control
D'Urso et al.	PHIDIAS	✓	BDI	✗	PE	Light Automation	✓	On/Off Control
Alaya et al.	Custom	✗	BDI	Fuzzy Logic	PE	Plastic Manufacturing	✗	Ejector Cylinder
Muto et al.	CHANS	✗	BDI	Fuzzy Logic	SE	Farming	✗	Negotiation and Sale
Rodriguez-Ubeda el at.	Jason	✗	BDI	Fuzzy Logic	SE	Irrigation System	✓	Sprinker
Xiaochao et al.	CGF	✗	BDI	Fuzzy Logic	SE	Air Combat	✓	Jet Control
Cruz et al.	N/A	✗	BDI	Fuzzy Logic	N/A	Formal Definition	✓	General Purpose
Mekki et al.	Custom/ MATLAB	✓	BDI	Fuzzy Logic	PE	Logistics	✗	Data Analysis
Othmane et al.	CARS	✓	BDI	Fuzzy Logic	SE	Traffic Network	✓	Recommendation System and Data Analysis
Morris and Ulieru	FRIEND	✗	BDI	Neuro-Fuzzy	SE	Social Simulation	✓	Human-Agent Interaction
Rosales et al.	Custom	✓	BDI	Fuzzy Logic	SE	Social Simulation	✓	Human-Machine Interaction
This study	Jason	✓	BDI	Fuzzy Logic	PE	Production Line	✓	Process Control

However, diverse colours cannot be represented in nature by a limited integer set. Therefore, these R, G, and B values should be fuzzified and turned into linguistic variables. These linguistic variables can even detect intermediate (blended) colour states between primary colours, widening the colour detection range (i.e., product recognition). Moreover, the combination of these colour types may have to be defined into subcategories using quantifiers such as *low green*, *high red*, and *middle blue* that ease the vagueness. In this case, classifying the data and achieving conjunctive statements became cumbersome or unattainable for the developers using classical logic. Hence, fuzzy logic can be applied to conventional agent development to enhance the abilities of the BDI agents.

Moreover, the aleatoric uncertainty that emerged from the sensors that do not have linear operation characteristics or are imprecise can be fuzzified instead of defining many control statements bound to relegated classical logic. Thus, fuzzy logic can lessen the engineering trial and error endeavour, development burden and time consumption. The execution phase of the motor actions can be smoothed when the velocity parameter should be decreased or increased calmly instead of being exposed to alternating and instant speed shifts. For instance, in case of any external epistemic uncertainty that leads the product to suck occurs, a fuzzified manoeuvre may be required when a conveyor belt should be moved back and forth so that the smooth movement transitions may not harm the motor, conveyor belt and products. Furthermore, according to the colour of the products and benefiting from the membership degrees, the speed and, consequently, the force of a dynamic system can be arranged considering organic products. This might be highly correlated between the colour of the products and their rigidness. Therefore, a specific physical action might be required when the product characteristics vary concerning the durability of that product. For instance, if the product is a red apple, it is likely a rigid one so that a high force can be applied. However, if it is spoiled, it means that its colour is between red and purple, so the same force cannot be used not to shatter the product. Therefore, the force can be arranged according to membership degree to the red colour.

As a result, the fuzzification process can be dedicated to the system under scrutiny to mitigate the uncertainty that can jeopardise sustainability and decrease the system's efficiency. The BDI agents can then perceive fuzzy inputs, store fuzzified data in their belief base, and select the plans according to linguistic states and environmental context, applying a plan selection phase based on the membership degrees. They then trigger the fuzzified actions within the selected plan, employing the membership degrees to configure the execution parameters. The fuzzy procedural reasoning model is further explained in subsection 4.2.

4.2. Fuzzy Procedural Reasoning Model

The Procedural Reasoning System (PRS) is one of the most used approaches (Calegari et al., 2021) in agent development, which understandably embodies the BDI paradigm (recall subsection 2.2). In PRS, an agent does not plan from scratch. Instead, it has a collection of pre-tailored plans. The developer creates these plans in the development time to respond to possible scenarios the agent may face during the run time. As mentioned, these plans have been designed using classical logic, which may not be enough to handle the run-time uncertainties for the CPS. Moreover, fuzzy logic allows us to define execution and sensing boundaries considering time-based and/or event-based operations. Hence, the fuzzy logic that extends the classical logic can be applied to augment the monitoring, analysis, planning and execution phases of the CPS, creating a rule-based system over a knowledge-based integrated with the BDI agents.

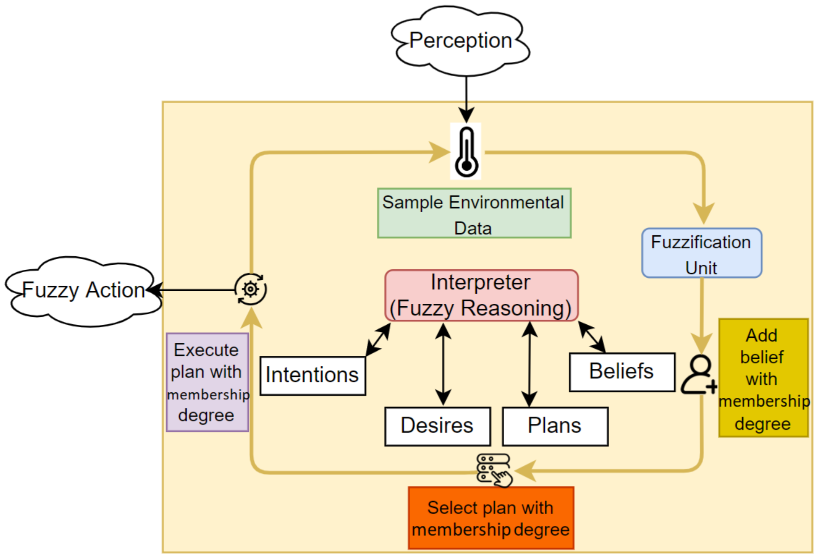
Therefore, in this section, the PRS enhanced with a fuzzy logic model is used to reveal the fuzzified stages of the reasoning cycles of BDI agents. These stages are fuzzy perception, fuzzy analyses, fuzzy plan selection, and fuzzy action execution. As the BDI agents are context-aware entities provided by the PRS, they can collect data and store them in their belief base to select a plan according to the current context (recall 2.2). Listing 1 directs the Jason plan structure:

Listing 1: The structure of the Jason plan mechanism.

```
triggering_event: application_context <– plan_body.
```

The fuzzy membership functions transform the data perceived from the environment into verbal variables. At the end of this transformation, the membership degrees of each verbal variable are also created and stored as fuzzy beliefs in the agents' belief base. Thus, instead of evaluating the preconditions of the plans to be selected as true or false, they are instructed as fuzzy rules conforming to the rule base. At first, the membership degree of each plan is fetched from the belief base. After evaluating the fuzzy preconditions of the plans, the one with the most elevated membership degree is designated for execution. During the selected plan's operation, the membership degree also ensures that the actions within the plan are carried out in a fuzzified manner. In other words, the membership degrees are utilised as quantities that can be multiplied by the configuration parameter assigned in the design time. In this way, fuzzified actions can be performed in case of uncertainties. An illustration of the proposed fuzzy-logic-based model and the enhanced cycle of the PRS architecture is displayed in Figure 1. A relevant reader can further inspect our previous studies (Karaduman et al., 2022b; Karaduman & Challenger, 2022) that illustrate this model on the MAPE-K loop and conceptually, respectively.

Figure 1 Procedural reasoning system enhanced with fuzzy logic.


Perception
Sample Environmental Data
Fuzzification Unit
Fuzzy Action
Interpreter (Fuzzy Reasoning)
Execute plan with membership degree
Intentions
Beliefs
Add belief with membership degree
Desires
Plans
Select plan with membership degree

At the first deployment and run of the system, each BDI agent is represented by a set of initial beliefs, goals, and a collection of plans. In the proposed model, beliefs are first-order logical expressions enriched with certain membership degrees obtained by fuzzification of the crisp values perceived by the environment. When a BDI agent starts, the goals desired to be achieved by the agent are pushed to the intention stack via a *triggering event* (recall 2.2). The intention stack contains all the goals to be satisfied. The agent, in turn, selects the goal at the top of the intention stack and determines the plans it can use to achieve that goal. Only some suitable plans become possible options because the *application context of the plans* (pre-conditions) should be matched with the terms in the agents' current beliefs. Then, the BDI agent runs the actions (i.e. post-conditions) in the selected *plan body* to reach their goals.

Choosing between different possible plans is based on using utilities, which are numerical values, of plans in the traditional PRS system. When the plan has the highest utility, the agent considers it the highest priority. If no priority exists among the existing plans, the priority levels are considered the same, and any of them is selected. Therefore, the line order of writing multiple plans with the same priority level is regarded as the contextual checking order. In the proposed approach, the membership degree is calculated for each plan to evaluate the pre-conditions. As a result, the picked plan is the one that has the highest membership degree. The membership degrees of the plans are also considered as the configuration parameter in corresponding actions within the plan in a fuzzified manner alternative to the traditional defuzzification phase. In subsection 4.3, the fuzzy-logic enhanced proposed architecture is mentioned considering the cyber, physical and network aspects.

Algorithm 1 Fuzzy Logic Enhanced Procedural Reasoning System of the BDI Agents, in pseudo-code

```
1: $\widetilde{B} \leftarrow \widetilde{B}_0$ /* $\widetilde{B}_0$ are initial fuzzy beliefs */
2: $I \leftarrow I_0$ /* $I_0$ are initial intentions */
3: $K \leftarrow K_0$ /* $K_0$ is the initial knowledge base */
4: while true do
5:     get next percept $p$ via sensors;
6:     $\widetilde{B} \leftarrow bff(\widetilde{B}, p)$;
7:     $D \leftarrow options(\widetilde{B}, I)$;
8:     $I \leftarrow filter(\widetilde{B}, D, I)$;
9:     $[\widetilde{\pi}, \lambda] \leftarrow plan(\widetilde{B}, I, Ac)$; /* $Ac$ is the set of actions, $\widetilde{B}$ is the set of fuzzified beliefs, and $\lambda$ is the triggering degree */
10:    while not $(empty(\widetilde{\pi})$ or $succeeded(I,\widetilde{B})$ or $impossible(I,\widetilde{B}))$ do
11:        $\widetilde{\alpha} \leftarrow$ first element of $\widetilde{\pi}$;
12:        $execute(\widetilde{\alpha}, \lambda, \widetilde{B}^{+})$;
13:        $\widetilde{\pi} \leftarrow$ tail of $\widetilde{\pi}$;
14:        observe environment to get next percept $p$;
15:        $\widetilde{B} \leftarrow bff(\widetilde{B}, p)$;
16:        if $reconsider(I,\widetilde{B})$ then
17:            $D \leftarrow options(\widetilde{B}, I)$;
18:            $I \leftarrow filter(\widetilde{B}, D, I)$;
19:        end if
20:    end while
21: end while
22: procedure BFF($x$)                                  ▷ / bff - *Belief Fuzzification Function*/
23:    $[A, \widetilde{u}_A] \leftarrow fuzzifier(p, K)$
24:    $\widetilde{B}' \leftarrow [A, \widetilde{u}_A]$
25:    $\widetilde{B} \leftarrow brf(\widetilde{B}')$
26:    $return(\widetilde{B})$
27: end procedure
```

Algorithm 1 shows the fuzzy logic enhanced procedural reasoning system considering the related works (Rodriguez-Ubeda et al., 2015; Bordini et al., 2007; Bosello & Ricci, 2020). In line 3, the initial knowledge base is defined. This knowledge base also refers to the fuzzy rules defined as Prolog-rules mentioned in Section 5. In line 6, $\widetilde{B}$ represents the fuzzified(~) beliefs that contain the membership degrees $\widetilde{u}_A$ (recall Definition 2.3) in the BDI agents' belief-base. In other words, $\widetilde{B}$ is also a type of agents' belief, but also in the form of fuzzy. In this regard, the belief fuzzification function (i.e., procedure) (*bff*) fuzzifies the percepts ***p*** gathered from the environment. This function works as the belief revision function (*brf*), but before revising the beliefs, it also fuzzifies.

Lines 22 and 25 define the *bff*. Specifically, in line 22, the percepts are fuzzified in the *fuzzifier* phase using the fuzzy functions represented in Equation 1 mentioned in Section 5. These fuzzy functions are stored in the knowledge base K, which correlates them using linguistic terms and logical statements. Moreover, membership degrees ($\widetilde{u}_A$) of each percept are calculated to represent the percepts as the degree of truth concerning their fuzzy sets defined by the fuzzy functions. For each agent cycle, the membership degrees $\widetilde{u}_A$, are calculated, as new fuzzified beliefs $\widetilde{B}^{'}$. The fuzzified beliefs are then revised by the *brf*. Lastly, the bff returns to line 6.

In lines 6-9 and 15-18, the desire option consideration and intention filtering are also done using the fuzzified beliefs $\widetilde{B}$ for selecting a plan using triggering degree (λ) and returning λ for fuzzified actions. Therefore, during the precondition checking phase of the candidate plans, the plan with the highest $\widetilde{u}_A$ is selected as the λ among the literals such as *low*, *middle*, *high* etc. in a fuzzified manner ($\widetilde{\pi}$). The selected fuzzified plan's actions are also performed using its triggering degree (λ) within the fuzzified plan (π) as fuzzified executions $\widetilde{\alpha}$.

In line 13, fuzzified plan execution $\widetilde{\alpha}$ also uses the $\widetilde{B}^{+}$ which is in the regular expression form where at least one and at most many fuzzy beliefs are expected to execute the plan steps/actions using the corresponding membership degrees or at least the triggering degree (λ). In the following subsection, we mentioned the proposed fuzzy BDI architecture that we enhanced and extended on our previous work (Karaduman et al., 2023b).

4.3. *Proposed Fuzzy BDI Architecture*

The proposed architecture has been devised to develop a distributed agent-based CPS established on an integrated pattern that considers cyber and physical layers, allowing logic-based enhancements and further extensions such as learning (Bosello & Ricci, 2020). Moreover, this architecture has also been designed considering the requirements of deploying software agents into the embedded system without requiring additional programming language, additional hardware, and loosely coupled connections in study (Karaduman et al., 2023a). Using fuzzy logic, this study considers its *enhancements* and *extensions*. The overall objective is to establish a tightly coupled platform that facilitates future logic-based enhancements. This is achieved through the initial validation of our fuzzy logic-based approach integrated with BDI agents. The architecture is illustrated in Figure 2. Initially, the layers had been defined in study (Karaduman et al., 2023a) solely using the BDI agents. In this study, we briefly mention these layers to maintain correlation and integrity. Each layer is re-elaborated and explained in the following sub-sections.

4.3.1. *Physical Layers*

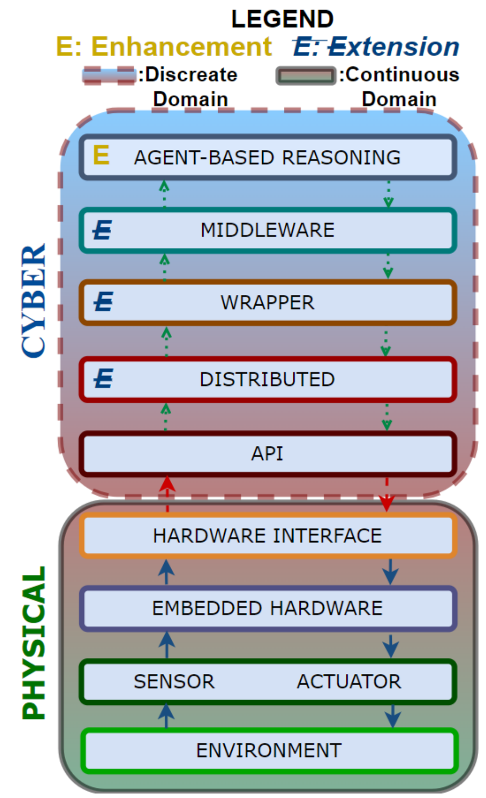
The physical aspect of the system is divided into four layers given below.

Environment Layer: portrays the habitat of the CPS created and a source of the unpredictable events. This environment can also be in cyberspace, considering the simulation-based approaches. Eventually, the simulated CPS should be deployed into a physical environment or have a hybrid establishment. A Cyber-Physical System exerts an effect on its circumference and is susceptible to environmental occurrences triggered by both various natural and human-induced factors, which may lead to uncertainties.

Sensor and Actuator Layer: defines the physical monitoring and execution phases. An agent-based CPS monitors the events generated by the environment using its sensors. Generally, these sensors are influenced by environmental noise and limited data processing due to the traditional approaches. Therefore, the selected sensors' information processing capability should be improved with intelligent methods for better perceptual ability. Actuators are the influencer components of a CPS controlled by the agents to create motion and movement upon an entity. Therefore, the executions of the agent-based CPS should be advanced smoothly and timely considering the state of the physical entities.

Embedded Hardware Layer: represents a microcontroller capable of computation using a programmable microprocessor extended with additional hardware features such as input and output ports for sensors and actuators' connections. The chosen embedded hardware must possess the necessary capabilities

Figure 2 The proposed enhanced and extended architecture's platform-independent overview.



to effectively execute the desired agent framework and additional enhancements of the agents that may bring extra computational costs, such as Bayesian methods, fuzzy logic, neural networks, and neuro-fuzzy models.

Hardware Interface Layer: covers the extension hardware utilised for technology-specific applications. Optionally, these boards may also have specific computation elements or extra threads for time-consuming computations.

4.3.2. Cyber Layers

The system's cyber aspect comprises five layers introduced below.

API Layer: expresses the code library for a specific hardware technology. The conventional approaches provide a basic abstraction using the traditional logic approaches to access the functions of the selected embedded hardware. These functions also depend on the hardware characteristics of the chosen sensors and actuators. Therefore, the API operations must allow access to refined functionalities such as data gathering modes, actuation types/parameters, etc. In this way, the enhancement(s) can be applied to the refined or abstracted functionalities of the hardware.

Distributed Layer: enables creating scale-able network topology for distributed agent-based CPSs. In this way, multiple agents can be contained by different containers/embedded devices while they can send messages to each other using agent communication. In addition to the agent communication protocols, technologies such as the MQTT, OPC-UA, and TCP/IP can be preferred to inter-operate non-autonomous paradigms.

The Wrapper Layer: describes wrapping the multiple computational steps, such as low-level device details, enhancement methods such as fuzzy, Bayesian and neural models, belief addition/deletion and binding methods with agent mechanism into a method to provide an abstraction. These invariant steps are encapsulated to provide an easier way to use this method within the agent software. This way, the complexity is lessened without creating an extra burden for the plan structure and reasoning mechanism instead of defining many consecutive actions. Moreover, design patterns and Agents&Artifact approaches can also be utilised in this layer, having more modularity in replacing the low-level API functions or enhancement models without altering the agent software.

Middleware Layer: provides an infrastructure for internally binding the wrapped functions to the agent-oriented software and externally providing communications. It also allows for object-oriented level programming, library inclusions and file operations. This layer may also provide a horizontal extension for data analysis and information gathering from external systems and services to support autonomy. The enhancement models, such as fuzzy logic, can also be used within the middleware layer for data filtering or data fusion delivered by the network. The fuzzified beliefs are also added to the agent's mind via this layer (also belief deletion/update). In this way, it reduces the complexity of the agent programming. The middleware layer connects the wrapped functions with the agent's actions in the plan structure, bridging the reasoning mechanism and the low-level control established in the embedded hardware layer.

Agent-based Reasoning Layer: is the layer of cognition for leveraging the CPS to sCPS, allowing multiple enhancements such as fuzzy logic, neural networks, learning, and neuro-fuzzy approaches. In this layer, the procedural reasoning layer should be augmented by the proposed AI approach and blended with the planning mechanism to create a more intelligent mechanism to sustain the system during runtime. The reasoning mechanism should not be bound to conventional approaches. Instead, it should follow multi-logic models integrated and interoperated with each other to form a better judgment of uncertainties, then should execute the plans using the multi-logic methods. In addition, learning approaches can also be adopted by extending this architecture vertically. Figure 4 illustrates the Component Diagram representation of the enhanced and extended points resulting in the impact on the sensor actuator layer. As the CPS is a paradigm in which the cyber and physical aspects influence each other, the enhancement and extensions on the cyber side impact the operation of the physical side.

We introduced the hypothetical structure of the architecture and its layers that advocate applying AI methods to enhance these layers. The developers and practitioners should also extend the architecture to achieve more sustainable systems to create various design choices conforming to its abstract representation. Seeing the aforementioned abstract architecture described in Figure 2, we implemented the proposed case study using the concrete architecture represented by Figure 3. As we aim to run this case study in a physical environment, LEGO technology was selected to create a concrete system representing the physical side. To run the Java program in an embedded device, we preferred to use RaspberryPI 3. To facilitate the control of LEGO sensors and actuators, the implementation requires a hardware interface. In this case, the BrickPI [2] board was chosen as the hardware interface. The BrickPI board serves as a means to connect and interface with the LEGO components, forming the physical side of the implementation. It enables communication and control between the software agents and the LEGO sensors and actuators, allowing for the integration of the agent-based control system with the physical components of the CPS.

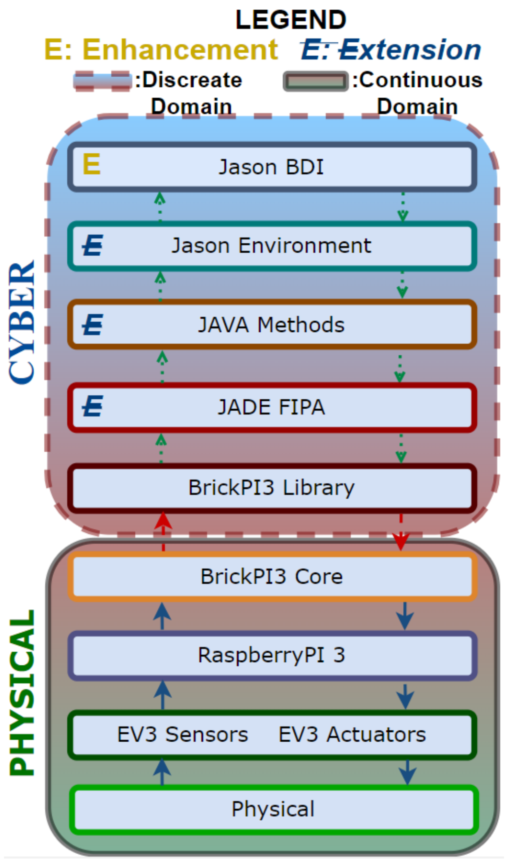
The Java-based API of the LEGO technology is utilized to program the BrickPI board on the cyber side. This API provides a set of Java functions that allow for the control and interaction with the LEGO sensors and actuators connected to the BrickPI. To improve usability and provide convenience, encapsulation of the LEGO API functions is achieved through the use of Java wrapper methods. These wrapper functions serve as a layer of abstraction, simplifying the usage of the LEGO API and providing a more user-friendly interface for programming the BrickPI and controlling the LEGO components. We used the

[2]https://www.dexterindustries.com/brickpi

JADE infrastructure option of Jason to create a distributed MAS contained by two embedded devices, layer 1 and layer 2. It should be noted that Jason creates a JADE environment and agent containers automatically to establish a distributed MAS. The Jason Environment were integrated with the API methods and then wrapped using the Java functions. The use of *Literals* definitions and *Terms* depictions of the Jason Environment and its library allowed us to correlate the actions of the agents with their corresponding Java functions to perform specific functionalities. Lastly, Jason is used to creating BDI agents and for procedural reasoning and distributed BDI agents communicated using the JADE infrastructure within Jason[3].

The next section provides comprehensive details regarding the implementation and integration of agents with the hardware components.

Figure 3 Concrete overview of the proposed enhanced and extended architecture.

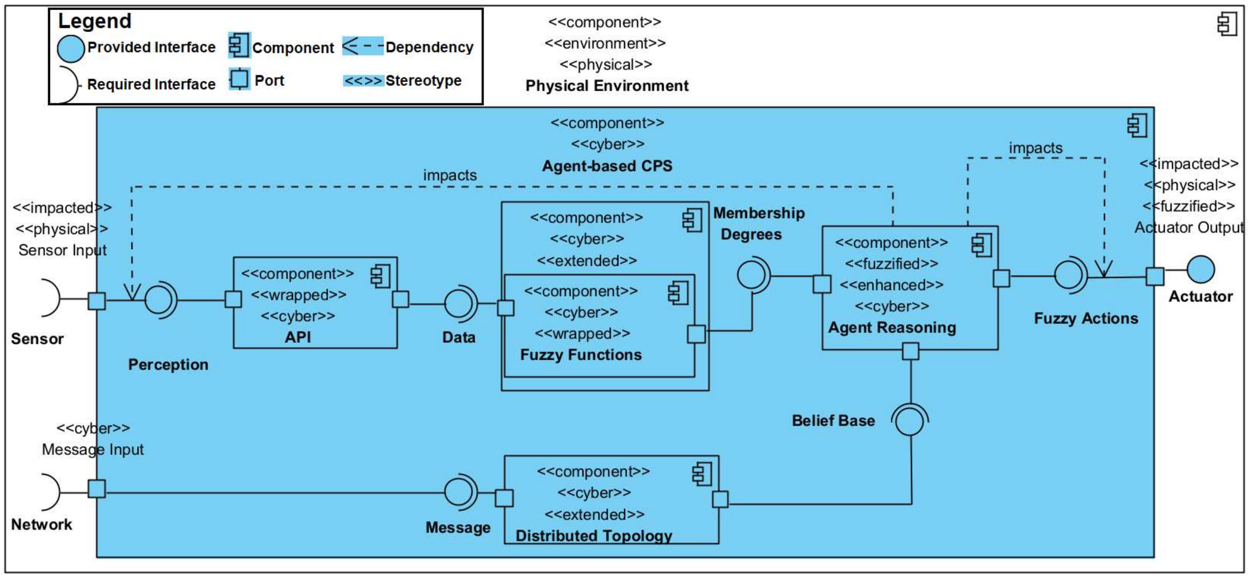


4.4. Case Study: A Smart Product Line

This section presents LEGO's agent-based smart production system, a concrete and composable technology. Moreover, we have studied traditional simple-reflex agents without fuzzy logic enhancement in an earlier study (Yalcin et al., 2021) using SPADE framework (Gregori et al., 2006) and have seen the necessity of an enhancement, fuzzy-logic-based BDI agent approach (Calegari et al., 2020). A relevant reader is guided to an earlier version of this study (Yalcin et al., 2021). This study used the Jason framework to develop and deploy the BDI agents.

[3]https://jason.sourceforge.net/mini-tutorial/jason-jade

Figure 4 The Component Diagram representation of the Enhancement and Extension points of the proposed architecture and the resulting impact on the sensor actuator layer.



In general, the system mimics the behaviour and processes of industrial classes of CPS. However, it simplifies their cyber and physical parts to the degree that allows fuzzy- and agent-based experimentation for developing our approach. We aimed to mimic a sauce production process in the proposed case study. The aim is to use red tomatoes to make sauces. To a certain degree, spoiled tomatoes can be tolerable. However, adding too many spoiled tomatoes can be a problem for the product quality. Moreover, green peppers can also be added to create a mixture of red tomatoes and green peppers to make different sauces. It is also possible that other green peppers with various tones can be added to the mixture. Before that, the system should be able to distinguish the spoiled tomatoes and try to eliminate them. Nevertheless, the colour sensor is imperfect and has to work with uncertainty since only constraint colour information can be processed. Therefore, we should apply a fuzzy-logic-based approach to mitigate these problems. In addition, during the system's operation, a person may approach the system as human behaviour is inherently unpredictable (Garlan, 2010), which accordingly causes external uncertainty in the system. Therefore, ultrasonic sensors should detect human intervention. Still, the system cannot stop the conveyor belt instantly not to delay the production process. The system should also consider the person's safety near the functioning system. According to the distance, the speed of the conveyor belt should be controlled smoothly and considering safety/delay. On the one hand, if the motor of the conveyor belt is stopped so fast, it may harm the motor and the product on the conveyor belt.

On the other hand, if the conveyor belt speed cannot be controlled smoothly, this may harm a person critically. In this regard, a fuzzy-based approach can be applied where it is a burden or impossible to implement smooth speed transitions using traditional logic. Therefore, we can devote the fuzzy-based approach to provide smoothness and easiness. Moreover, the products may be stuck during the conveyor belt operation. Thus, the conveyor belt should be moved back and forth smoothly. Therefore, an adaptation plan should be applied to mitigate this problem. To satisfy these requirements, we implemented a fuzzy and multi-agent-based approach to control the production line case study processes and tackle a degree of uncertainty, benefiting from the BDI architecture. The BDI agents that were deployed in our study are as follows:

- **Init Agent:** The primary purpose of this agent is to handle the initialization, taking charge of sending messages to initiate the sequence of actions of the system. It was also required to delay the system start-up until device configurations were completed. There are two individual Init agents to initialize the system configurations for the first and second conveyor belts. The First conveyor belt includes *Drop Agent*, *Shred Agent*, and *Sort Agent*, while the second conveyor belt refers to *Push Agent* and *Press Agent*. The division of these agents was realized according to the control operations on two conveyor belts.

- **Drop Agent:** In charge of overseeing the product input, this agent actively manages the process by dropping a product (brick) onto the first conveyor belt when a message is received. It also ensures that products are properly dropped onto the first conveyor belt for further processing.

- **Shred Agent:** Tasked with controlling multiple components, including the first conveyor belt, the motor of the shredding component and the ultrasonic sensor, this agent directs their operations by managing the movement of the belt and activating the shredding motor when required, guided by inputs from the ultrasonic sensor.

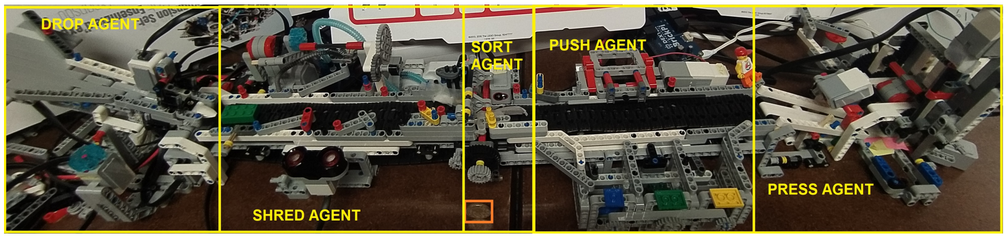
- **Sort Agent:** Taking on the role of colour determination, this agent's task is to resolve the colour of each product (brick) and exert control over the second conveyor belt. By employing the colour information, this agent positions the products in front of their predefined buckets on the second conveyor belt.

- **Push Agent:** Tasked with message handling for pushing the products (bricks), this agent exercises control over the pushing movement. It stages the actions of the pushing component to put the products in their respective buckets.

- **Press Agent:** Tasked with managing the combination of products (bricks), this agent receives the products, utilizes the press component to hold the first product, merges it with the second product to generate a concatenated products, and subsequently releases the concatenated products from the press reservoir.

  These agents work together to perform the necessary actions and coordination for the efficient operation of the system, ensuring that bricks are processed, sorted, and combined according to the desired specifications. Agents and their related segments are illustrated by Figure 5.

Figure 5 Agents and their corresponding sections.



The abstract architecture that allows for fuzzy logic enhancement and its concrete implementation on the presented BDI agents deployed on the production line system is described further in Section 5.

5. An Implementation of the Proposed Architecture

This section mentions a concrete implementation of the proposed architecture enhanced with fuzzy logic. As previously presented, software agents are self-contained entities that can achieve their goals by providing local control for the different parts of any system. Despite trying to achieve their goals, distributed agents work harmoniously to control the heterogeneous parts of the proposed case study. Accordingly, the BDI agents can perceive their environment through sensors, process the environmental data and behave goal-oriented to achieve their goals enhanced with fuzzy logic. In the proposed case study, agents are geared towards controlling heterogeneous segments of the system based on fuzzified beliefs, pre-defined rules and fuzzified smooth execution inherited from the fuzzy logic. The two conveyor belts, somatic components and different functions construct the multi-stage, heterogeneous and complex system.

First, the system is initialised to be ready to run. Then, the first colour sensor detects whether there is a product (i.e., Lego bricks). Secondly, If any product is inputted, the product is dropped onto the first conveyor belt. Once this action is triggered, the conveyor belt is started to deliver the product to the next phases. A brick moves through the border between the first and second conveyor belts. The second colour sensor detects the product and decides its colour based on the fuzzified R, G and B data. The fuzzy-BDI agent's belief towards the colour of the brick has to deal with both the noise and the reading distance that varies from 0.5 cm to 2 cm. The random change of distance of each arrived product causes the alternation of the R, G and B values, leading to vagueness and the need for linguistic definitions such as low, medium, high, very high, ultra, and ultra-high.

Moreover, the conveyor belt is not stopped not to cause a processing delay, product loss and degeneration of the physical components. In this case, the data may become more imprecise and hard to interpret the combination of R, G and B values to extract the meaning. The fuzzy-BDI agent can mitigate this uncertainty as the fuzzy logic provides generality and simplicity to group the data. The product's colour is decided based on the defined rules in the knowledge base using fuzzy beliefs. The highest membership degrees of the membership functions are characterised by the mentioned linguistic variables and are calculated. Therefore, the *Colour*, *ReverseAction* and *Speed* columns, which are mentioned in the following, are either decision result that creates another goal or actions within the BDI agent's plan (recall subsection 4.2). The source codes regarding the implementation details can be reached on this link[4]. The agent software can be found under *src* folder[5] and low-level API code can be reached in *java* folder[6]. To balance succinctness and transparency, the implementation details refer to this repository.

The Prolog rules that belong to Sort Agent[7] and Shred Agent[8] are specified to select the highest membership degree for each term such as R, G, and B colours, conveyor belt speed and motor reversing force. Referring to the Sort Agent's and Shred Agent's fuzzified plans (recall Listing 1), that apply the aforementioned Prolog rules, can be reached on[9],[10][11]. Moreover, these plans corresponding rules are listed in Tables for colour distinguishment Table 4, reverse movements for the stuck brick Table 2 and emergency control Table 3, respectively[12].

In Sort Agent's software, line 43, *samplecolour* plan is represented. This plan collects the R, G and B values from the colour sensor, then runs *reverseMoments* action to check whether any activity is required to perform in case of the stuck brick (if there is a stuck brick, it does not arrive at the Sort Agent's colour sensor, see Figure 5). Then, it intends to run the *!decidecolourF* plan (defined in lines 52 and 107). The

[4]`https://github.com/micss-lab/FuzzyBDIAgents`
[5]`https://github.com/micss-lab/FuzzyBDIAgents/tree/main/src/asl`
[6]`https://github.com/micss-lab/FuzzyBDIAgents/blob/main/src/java/PLEnv.java`
[7]`https://github.com/micss-lab/FuzzyBDIAgents/blob/main/src/asl/sortAgent.asl#L6-L14`
[8]`https://github.com/micss-lab/FuzzyBDIAgents/blob/main/src/asl/shredAgent.asl#L6`
[9]`https://github.com/micss-lab/FuzzyBDIAgents/blob/main/src/asl/sortAgent.asl#L52-L105`
[10]`https://github.com/micss-lab/FuzzyBDIAgents/blob/main/src/asl/sortAgent.asl#L118-L124`
[11]`https://github.com/micss-lab/FuzzyBDIAgents/blob/main/src/asl/shredAgent.asl#L43-L48`
[12]For simplicity, the rest of the paper refers to these links using agents' names such as Sort Agent, Shred Agent etc.

plan *!decidecolourF* is triggered to decide red, purple, light green, middle green and dark green bricks. The plan has an application context such as *isitRed(high) & isitGreen(low) & isitBlue(medium)* (recall 1) that runs the previously mentioned Prolog rules provided by Jason. In this way, the Sort Agent determines the brick's colour. The Sort agent then guides the brick to either the Press agent's build section or the Push agent's push section. Besides, this plan structure also represents the rules given in Table 4. In other words, each *!decidecolourF* plan corresponds to precisely one rule given in the Table 4, e.g., *isitRed(high) & isitGreen(low) & isitBlue(medium)* represents the Rule 1 and decision result is the colour type of the product. According to the product colour, an action is performed to send the product to either the press section or the push section on the system. In this regard, plans named *!decidecolorF* in Sort Agent lines 52-105 correspond to the rules 1-25, respectively. The variety of the rules (or plans) is because of the reasons: i) our design approach of the membership functions, ii) the variable distance between the Sort Agent's colour sensor and the width of the conveyor belt that creates change in the R, G and B values.

The Drop Agent[13] informs the Sort Agent when it drops a brick on the conveyor belt. The Sort agent checks whether there is a stuck brick by pursuing the *!decideApplyReverse* goal. As lines 113-114 describe, if the bricks arrive in time, this counter is reset, and the goal switches to the *!samplecolour*. If the bricks do not arrive in a pre-defined (i.e., threshold) time configured according to the expert's opinion, the Sort agent drives the *!deciveRev* goal. Then, according to the altering context, a plan is selected for the *!deciveRev* goal, i.e., the time passes while the brick is kept in a stuck state, so the membership degrees change. Therefore, the Prolog rule defined in line 6 of the Sort Agent is instructed to trigger the suitable plan for this goal. The Sort agent's *!decideRev* plan for executing the reverse actions for the brick recovery defined in lines 118-122 are represented as Rules 1, 2 and 3 in the Table 4. Lastly, the Shred agent follows the goal *!emergency* based on the collected data from the ultrasonic sensor considering the safety. The emergency plan is to arrange the conveyor belt's speed if an obstacle is near the production line system. Lines 43 and 46 define two *+!emergency* plans corresponding to the rules in Table 3.

As can be noticed, *changeConveyorSpeed* action has two parameters that take the pre-defined speed of the conveyor belt and the membership degree. When the suitable plan is according to the context (i.e., highest membership degree) for each cycle, the membership degree is also used within the plan to dynamically re-configure the system for adaptation. The complement of the membership degree is taken by extracting from 1 to correlate the meaning of the membership function and its propagation on the action within the plan. Referring back to the colour decision phase, after the Sort agent decides the colour of the brick (product), that product is either pushed into the buckets applying the fuzzy-logic-based execution or sent to the pressing process. Lastly, one brick is held by clippers of the pressing platform. When the second brick arrives in the pressing platform, it is merged with the first one, and the merged brick is ejected to the outside of the system through the exit bucket.

To control the aforementioned complex process, we implemented our BDI agents to control defined actions of the production line system by applying fuzzy logic. In this way, they can communicate to work in harmony when a process is completed and another process should be triggered next. Their collaborative behaviour creates a sustainable and continuous system that applies fuzzy logic instead of traditional logic. During the process steps, if any uncertainty occurs that prevents the product's process, the system tries to mitigate it using alternative plans of agents benefiting from the fuzzy logic enhancement.

As illustrated by Figures 6 and 7 described by Equation 1, we preferred to use the fuzzy triangular sets as we have previous experiences (Ltaief et al., 2022; Karaduman et al., 2022b,a; Karaduman & Challenger, 2022; Yalcin et al., 2021; Can et al., 2022; Karaduman et al., 2023a) and considered the studies in the literature (Farias et al., 2010; Ben Mekki et al., 2016; Queiroz et al., 2022) that led us to the reasonable results. To define a fuzzy set using a triangular function, we can use the functions represented by Equation (1).

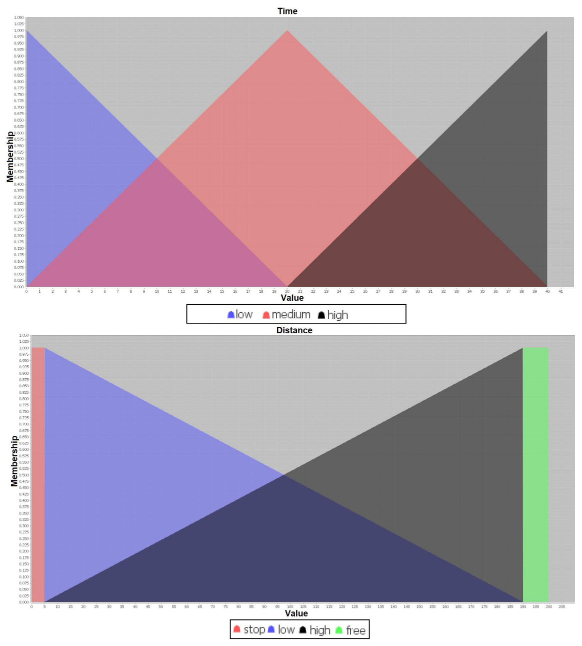
[13]`https://github.com/micss-lab/FuzzyBDIAgents/blob/main/src/asl/dropAgent.asl`

Let us define a triangular fuzzy set with a lower limit a, upper limit b, and a midpoint m between these limits, where $a < m < b$. The membership function for this fuzzy set can be defined as follows:

$$\mu_{trian}(x) = \begin{cases} 0, & \mathrm{x} \leq \mathrm{a} \\ \frac{x-a}{m-a}, & \mathrm{a} < x \leq \mathrm{m} \\ \frac{b-x}{b-m}, & \mathrm{m} < x \leq \mathrm{b} \\ 0, & \mathrm{x} \geq \mathrm{b} \end{cases} \tag{1}$$

In this definition, the membership value $\mu_{trian}(x)$ represents the degree to which an element x belongs to the fuzzy set. For values of x below the lower limit a or above the upper limit b, the membership value is zero, indicating no membership. For values of x between a and m, the membership value increases linearly from zero to one. Similarly, for values of x between m and b, the membership value decreases linearly from one to zero. By defining the membership function in this way, we can represent a triangular fuzzy set with the desired lower limit, upper limit, and midpoint, allowing us to model and reason with fuzzy concepts in a precise and flexible manner. In order to provide greater clarity for the rest of this subsection, the fuzzy rules and membership functions are introduced, offering beneficial perspectives. As mentioned, the Prolog rules defined in lines 6 and 14 of the Sort Agent's software and Shred Agent's line 6 select the highest membership degree among the defined fuzzy sets. Shred Agent's line 6 is defined as the rule for speed control that was experimented in subsection 6.3.2 and selects the highest membership for that agent cycle, which can be either *low speed* or *high speed*. Similarly, Sort Agent's line 6 represents the Prolog rule for selecting the highest membership among the membership functions that define the *Actionlow*, *Actionmedium*, and *Actionhigh* manoeuvres to save the stuck brick(s) experimented in subsection 6.3.3. Figure 7[14] and 6[15] illustrate the triangular membership functions we integrated to enhance the procedural reasoning system with fuzzy logic.

Figure 6 Distance and Reverseaction Membership Functions.



[14]The jFuzzyLogic was only used for the illustrative purposes.
[15]The jFuzzyLogic was only used for the illustrative purposes.

Figure 7 Colour Membership Functions.

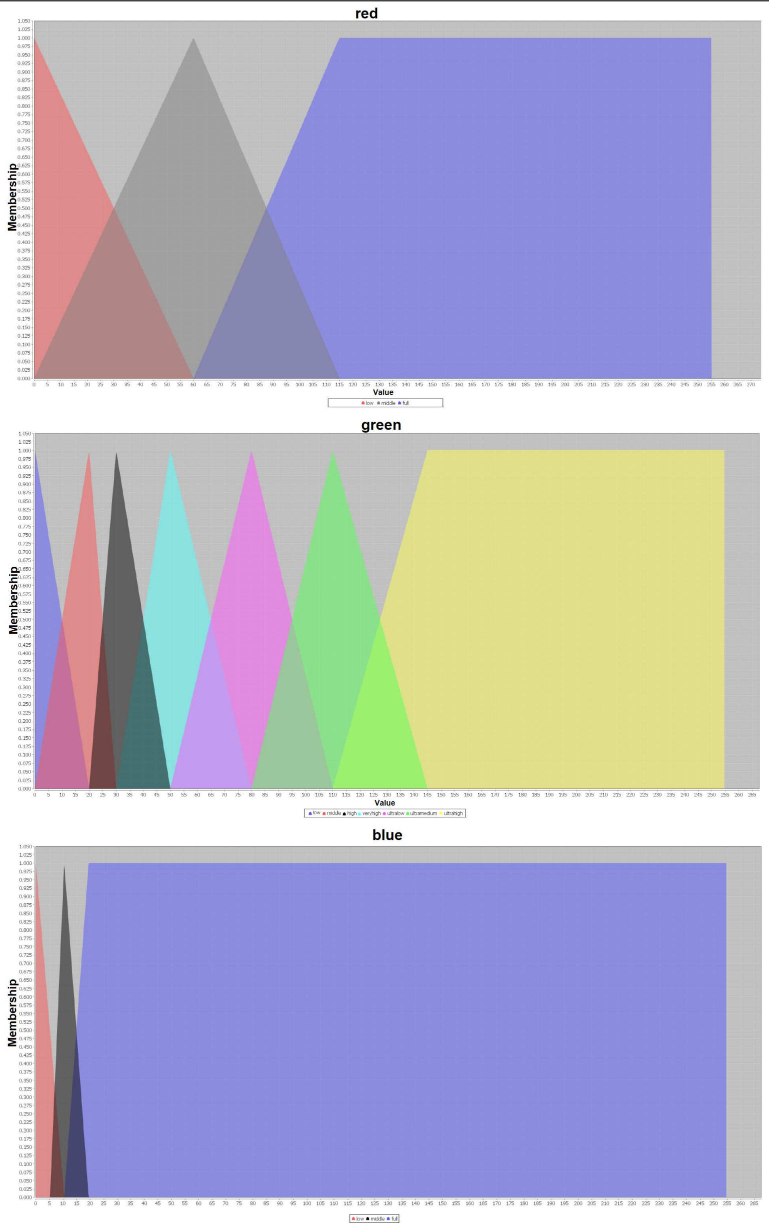


Table 2 Brick Stuck Save Action Rules of the Sort Agent

RULES	**Time**		**ReverseAction**
RULE 1	low	->	Actionlow
RULE 2	medium	->	Actionmedium
RULE 3	high	->	Actionhigh

Table 3 The distance and speed arrangement for the safety purposes of the Shred agent.

Rules	**Distance**		**Speed**
Rule 1	low	->	SpeedLow
Rule 2	high	->	SpeedHigh

Table 4 Fuzzy rules of the Sort agent as logical condition representation

Rules	Red		Green		Blue		Colour
Rule 1	high	AND	low	AND	medium	->	Red
Rule 2	high	AND	medium	AND	medium	->	Red
Rule 3	high	AND	high	AND	low	->	Red
Rule 4	medium	AND	high	AND	low	->	Red
Rule 5	medium	AND	medium	AND	low	->	Red
Rule 6	medium	AND	veryhigh	AND	low	->	Red
Rule 7	high	AND	veryhigh	AND	low	->	Red
Rule 8	medium	AND	medium	AND	medium	->	Purple
Rule 9	medium	AND	high	AND	medium	->	Purple
Rule 10	high	AND	high	AND	medium	->	Purple
Rule 11	medium	AND	medium	AND	high	->	Purple
Rule 12	high	AND	medium	AND	low	->	Purple
Rule 13	medium	AND	veryhigh	AND	medium	->	Purple
Rule 14	medium	AND	veryhigh	AND	high	->	Purple
Rule 15	high	AND	veryhigh	AND	high	->	Purple
Rule 16	medium	AND	ultramedium	AND	medium	->	LightGreen
Rule 17	medium	AND	ultralow	AND	medium	->	LightGreen
Rule 18	medium	AND	ultrahigh	AND	medium	->	LightGreen
Rule 19	low	AND	ultralow	AND	medium	->	MiddleGreen
Rule 20	low	AND	ultramedium	AND	low	->	MiddleGreen
Rule 21	low	AND	ultralow	AND	low	->	MiddleGreen
Rule 22	low	AND	ultramedium	AND	medium	->	MiddleGreen
Rule 23	low	AND	high	AND	low	->	DarkGreen
Rule 24	low	AND	high	AND	medium	->	DarkGreen
Rule 25	low	AND	veryhigh	AND	low	->	DarkGreen

So far, the plan selection and fuzzy membership functions have been mentioned. When an action within the plan is instructed, it is mapped and instructed a corresponding method contained by the Jason environment. For brevity, we only mention the fuzzy functions for the blue colour, as the other functionalities are more or less the same. A relevant reader can access Jason's Java environment for the implementation of the smart production line on this link[16][17].

The Jason environment code in line 977 represents how the *sampleColour* action is performed (recall Sort Agent's line 6). Firstly, the Jason BDI instructs the *action.equals* command to check whether the action is sampleColour. As the Sort Agent executes this action, the *fuzzyColourSensor* method is called. As the wrapper layer describes, this method is a wrapper function that includes the low-level device details, fuzzy function definitions and binding techniques. It wraps these invariant steps to provide an abstraction for the reasoning layer. This way, the complexity is lessened without creating an extra burden for the plan structure and reasoning mechanism instead of defining many consecutive actions. The *fuzzyColourSensor* method is described in lines 409 and 553. At line 417, the colour sensor is set to RGB mode. The sample is taken once at line 431, and sampled R, G and B data are transferred to the red, green and blue variables. The BrickPI board provides these functionalities that conform to the Hardware Interface layer and BrickPI's API defined in the API layer.

When the *fetchSample* function is called, a reading is taken from the physical environment via the colour sensor defined in the sensor and actuator layer. The previous perceptions related to the three membership degrees of the blue colour are cleaned just in case in lines between 467 and 469. In lines 16 and 20, fuzzy functions are defined. The gathered blue value at line 434 corresponding to the mem-

[16]`https://github.com/micss-lab/FuzzyBDIAgents/blob/main/src/java/PLEnv.java`

[17]Hereafter, the paper mentions this file as Jason environment code

bership functions is converted to the membership degrees. Lines 497 and 499 define the *Literals* for these three membership functions described by the linguistic variables such as *low*, *medium* and *high*. In the agent's belief base, these linguistic variables can also be represented as beliefs such as *colourBlue (low,0.7), colourBlue (medium,0.2), colourBlue (high,0.1)*. Lastly, these Literals are added to the Sort agent's mind as beliefs. Figure of the agent's run-time mind with fuzzified beliefs and linguistic variables can be accessed in the repository[18] As can be seen, it gathers the RGB values from the sensor API and then fuzzifies these inputs using the linguistic terms and membership degrees. In the next section, the empirical evaluation of the study is mentioned.

6. Empirical Evaluation

In this section, demonstrating scenarios, experimental setup and carried-out experiments are introduced.

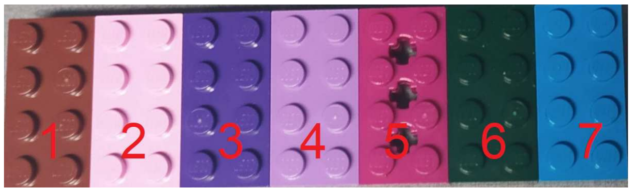
6.1. Demonstrating Scenarios

This subsection demonstrates scenarios to provide insights for the experiments mentioned in 6.3.

6.1.1. Colour Distinguishing Scenario

Figure 8 illustrates the bricks that have various colours. In reality, these colours are slightly darker than in Figure. As a traditional approach, the BrickPI API only supports certain tones of these colours as it uses the crisp values of Boolean logic. For example, 1, 3, and 4 are perceived as blue or black, 5 are recognized as red, 2 are identified as magenta, 7 are determined as blue, and 6 are decided as brown. Therefore, this emerges as a necessity to enhance the monitoring ability of the Sort agent using fuzzy logic for our candidate and specific products (bricks). In this way, we could provide correct grouping and specific distinguishing between the same colour groups such as light green, dark green and middle green.

Figure 8 Bricks that are recognized as wrong colours.



In our opinion, colour distinguishing has an external [19] aleatoric[20] and internal[21] epistemic uncertainties[22] that can emerge during the run-time, so the information processing capability of the BDI agent can be enhanced and the imprecision range can be grouped using fuzzy logic.

Red and Spoiled Red

[18]`https://github.com/micss-lab/FuzzyBDIAgents/blob/main/SortAgentMindHalf.png`

[19]The uncertainty is referred to as external because the environment's propagated effects caused imprecision in multiple sampled data when the system entities were kept static.

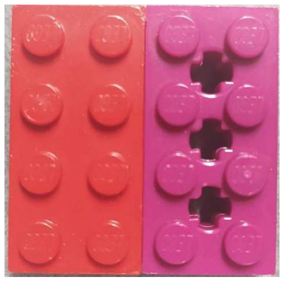
[20]The uncertainty is referred to as aleatoric because the R, G and B values vary in specified boundaries due to random noise from the environment.

[21]The uncertainty is also referred to as internal because the products (bricks) are controlled by and under the influence of the BDI agents and conveyor belts.

[22]The uncertainty is referred to as epistemic because the two types of BDI agents' logical abilities are evaluated to recognize the colour based on the beliefs.

As mentioned, the traditional configuration of the sensor API cannot distinguish between red and spoiled red, which is described by Figure 9 since spoiled red is perceived as a red brick. Therefore, we created and tuned membership functions based on our expert opinion (recall Figure 7). Spoiled red is a tone of colour that can be perceived as neither purple nor magenta because it is a pretty close colour to red. In this scenario, we aim to process more red tomatoes and eliminate spoiled ones as much as possible to achieve a healthy mixture. The demonstration that displays the red and spoiled red distinguishing process can be reached on [23].

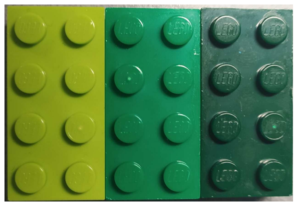
Figure 9 Red (left) and spoiled red (right) bricks.



Tones of Green

As mentioned, the traditional configuration of the sensor API only supports limited colours or incorrectly sees the different tones of type. Moreover, there is a vagueness in knowing which tone of green is perceived, narrowing the scope of the usable products. Figure 11 illustrates the selected green bricks (product). At first, we tried these green bricks and noticed that the colour sensor could not recognize light green and dark green as green colour. We selected these bricks to work with light and dark tones as well. The demonstration video, which displays the grouping of the green products in the same bucket, can be accessed at [24]. In the next subsubsection, the emergency control scenario is mentioned.

Figure 10 Colours of green: Light Green (left), Green (middle), Dark Green (right).



6.1.2. Emergency Control Scenario

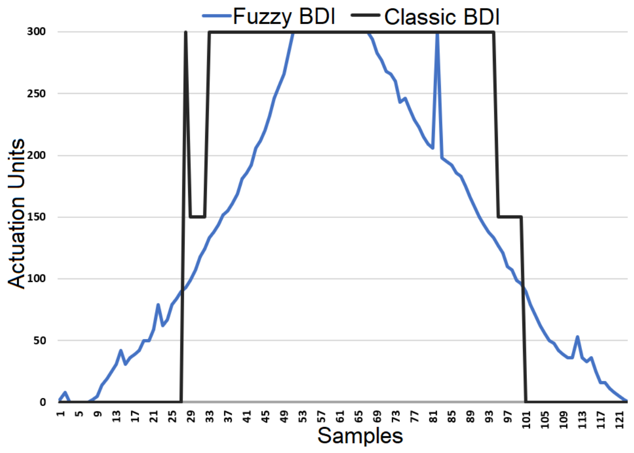
To achieve smooth conveyor speed adjustment during the conveyor belt operation in case of emergency, we fuzzified the Shred Agent's perceptual ability on the ultrasonic sensor. To compare both the traditional logic and fuzzy logic approaches, we implemented a traditional logic-based approach to control the speed of the conveyor belt. We observed that the ultrasonic sensor has no linear characteristic, so it may cause sharper strikes on the conveyor belt's motor during speed adaptation. In this case, the BDI agent's monitoring feature can be fuzzified.

Figure 11 illustrates a sample output of the fuzzy-BDI agent's (blue line) and the traditional BDI agent implementation control (black line) actions (actuation units of the motor) on the conveyor belt based

[23]`https://youtu.be/SmSpZaDiBSI`
[24]`https://youtu.be/tWh5w6eEABQ`

on the sampled perceptions gathered from the emergency sensor. The sample video demonstrates how the Shred agent applies the emergency plan using fuzzified beliefs, selecting the plan according to the membership degree and arranging the conveyor belt's speed by configuring the *changeConveyorSpeed* action within the plan using the membership degree. The demonstration of the implementation can be found on[25].

Figure 11 Emergency sensor perceptions and corresponding actions of both approaches.



Thus, in this experiment, emergency control of the conveyor belt using the fuzzy-BDI agent and classical-BDI agent was compared considering the external[26] epistemic[27] uncertainty may endanger both systems and humans. In the following subsubsection, the stuck brick recovery scenario is expressed.

6.1.3. Stuck Brick Recovery Scenario

In this scenario, it was assumed that the bricks (products) were stuck as the internal epistemic uncertainty because of the internal mechanism of the conveyor belt and the unpredictability of the stuck event. It was considered that if the product does not arrive in front of the Sort agent's colour sensor (see Figure 17) in 5 seconds, then the Sort agent activates the brick recovery plan. The bricks were placed vertically where bottlenecks on the conveyor belt are to realize this experiment, and the goal is turning the bricks horizontally, which is their normal run on the conveyor belt.

As the fuzzy BDI approach, the fuzzified executions were performed to save the brick from the stuck state. The demonstration that displays the recovery plan can be reached on[28],[29] and [30] for one, two and three stuck bricks, respectively. The demonstration of the classical BDI version recovery capability can be reached on[31] and[32].

Even though the videos show a short recovery time for the classical BDI approach, they were just trials that should be repeated many times. Eventually, this led us to the comprehensive evaluation, as it

[25]`https://youtu.be/fWr0hM0brXM`

[26]The uncertainty is referred to as external because a dynamic factor outside the system border may affect the system's operation during the run-time.

[27]The uncertainty is referred to as epistemic because of lack of knowledge about that dynamic factor's collision to the system, so the system's dynamic behaviour is lessened.

[28]`https://youtu.be/iYON4tvq1Ms`

[29]`https://youtu.be/36KWT9vxYXo`

[30]`https://youtu.be/VUX6pi3R0OE`

[31]`https://youtu.be/JsIo7AvA-fc`

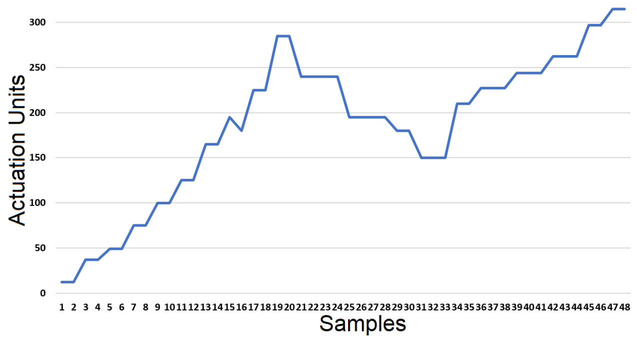
[32]`https://youtu.be/UhJX3RXHg08`

was also observed that the traditional method might not resolve the case in 40 seconds, which is demonstrated on[33]. Besides, this conventional approach reverses the motor instantly with a fixed speed. As may be envisioned, it does not seem so convenient considering industrial conditions such as the motor characteristics and abrasion of the system. It could also have led to a deadlock and could not resolve the situation in 40 seconds sometimes.

It was aimed to compare the brick recovery success of both approaches. Therefore, the bricks were placed by one, by two and by three in a stuck manner on the conveyor belt. The adaptation plan was triggered when the Sort agent could not perceive any colour after 5 seconds (recall Sort Agent[34]). The limit value to save the bricks was determined as 40 seconds. If this limit is exceeded, it was assumed that the operator would manually intervene to save the stuck bricks. The brick recovery plan was then triggered by the *Sort agent* to save the stuck brick(s) on the conveyor belt.

On the other hand, the classical implementation of the Sort agent tries to save the stuck brick(s) by reversing the conveyor belt's motor instantly with a fixed speed and angle in contrast to the fuzzy BDI agent's alternating fuzzified speed and angle executions. Figure 12 depicts a sample excerpt of the fuzzified executions as the motor's actuation units of the conveyor belt over the sampled perceptions to save the stuck bricks.

Figure 12 Fuzzified executions to save the stuck brick(s).



The maximum speed that the fuzzy BDI agent can apply was limited to the same fixed speed as the classical BDI agent can apply, considering the fairness of both methods. To determine the successful recovery, all brick(s) should be saved from their stuck places and gain the freedom to move on the conveyor belt in 40 seconds. In the following subsubsection, the controlling push force scenario is introduced.

6.1.4. Controlling Push Force Scenario

In this scenario, the push force of the Push Agent is dynamically arranged using the membership degrees according to the bricks' colours and expected values in a fuzzified manner. The membership degrees are multiplied by the pre-defined push force set by the domain experiment. Specifically, either red or green membership degrees were considered based on the triggered plan/rule (recall Table 4) of the colour groups (recall Figure 9 and 10) and multiplied by the maximum push force of the motor controlled by the Push Agent.

It was assumed that darker colours, such as spoiled red and dark green, represent rotten products that can be soft and more prone to be defective because of the applied force. In other words, we considered their rigidness and durability. For example, if it is a fresh tomato, it should be durable and red. However, if it is spoiled, then the colour becomes purple. Therefore, spoiled products should be hit using less force

[33]`https://youtu.be/mt5CiGStacI`
[34]`https://github.com/micss-lab/FuzzyBDIAgents/blob/main/src/asl/sortAgent.asl#L113-L114`

to avoid fragmentation and shattering. Accordingly, the membership degrees can set the pushing force for the product groups in a fuzzified execution. The excerpt demonstration can be accessed on[35].

In the classical BDI approach, the push forces for different bricks were kept constant. In other words, the Push Agent applied fixed pushing force to the bricks to send them outside the conveyor belt. At first, the red and spoiled red brick group had been (recall Figure 9), and then the dark, light and middle green bricks were inputted into the system (recall Figure 10).

Both the Classical BDI agent and Fuzzy BDI versions of the Push Agent were compared with the assumption of the Ideal Agent version that fulfils the expected push forces for five different colours without any deviation. In the next subsubsection, the end-to-end scenario is noted.

6.1.5. End-to-end Scenario

In this scenario, end-to-end regular product processing was followed, starting from the Drop Agent to the Press Agent (see Figure 5). Specifically, it was aimed that the system should push the Dark Green and Spoiled Red into the buckets, allowing the pairing of any Red, Middle Green, or Light Green combination in the Press phase.

For instance, a series can be *Spoiled Red, Red, Dark Green, Middle Green, Red, Spoiled Red, Red, Dark Green, Light Green, Middle Green, Red, Spoiled Red, Red, Spoiled Red, Dark Green, Middle Green, Red, Spoiled Red, Spoiled Red, Middle Green.* In this case, the ideal sequence as pairs is *Red-Middle Green, Red-Red, Light Green-Middle Green, Red-Red, Middle Green-Red, Middle Green (no pair as sequence ends)* and expected exclusions, i.e., expected pushed products, is *six Spoiled Red and three Dark Green.*

The production line processes these products (bricks) and presses two products to combine them as pairs. In this regard, each series has its ideal sequence that can be compared with the produced products by the system, both using the fuzzy BDI and classical BDI agents. A representative demo of the pressing action can be reached on[36] and a representative demo regarding the end-to-end operation can be accessed on[37]. The experimental setups of the scenarios are mentioned in the following subsection.

6.2. Experimental Setup

To evaluate our approach, we have created six experiments. At first, we aimed to be sure that any environmental variable may affect our experiments. We then carried out our experiments in a modular manner, which can be grouped as the first and second conveyor belts, resulting in the end-to-end evaluation as well.

*6.2.1. **First Conveyor Belt***

We established mindful setups for the emergency control and stuck brick recovery experiments.

Emergency control experiment uses the ultrasonic sensor and a weighted brick on the first conveyor belt side. The ultrasonic sensor gathers the distance to the object approaching the conveyor belt. The operation range was measured using a ruler on the physical system, and its corresponding values were saved on the cyber side. The operation range was constrained to 5.5 cm at most, considering the ultrasonic sensor returns the maximum integer value when the distance is quite far.

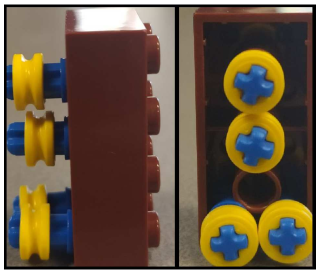
According to the closeness of the approaching object, the classical BDI agent halves the speed of the conveyor belt if the object's length is between 4 and 5.5 cm. Similarly, if the object approaches further and goes below 4 cm, then the conveyor belt is stopped. On the fuzzy-BDI agent side, it slows down the

[35]`https://youtu.be/CRyQvjKg4bQ`
[36]`https://youtu.be/WPk96bD9cl4`
[37]`https://youtu.be/dRUyXYuDPlY`

conveyor belt according to the defined membership functions, membership degrees, and their multiplication with the speed of the conveyor belt (recall Shred Agent code[38] and Figure 6).

The fuzzy-BDI agent thus provides smooth stopping for the conveyor belt. Moreover, at first, an ordinary LEGO brick was tried to evaluate the emergency control effectiveness of both approaches. However, as the regular brick was quite a light product for this goal, a particular brick was designed by adding additional LEGO parts for a more concrete entity. In this way, it could stand on when the conveyor belt was idle and perform minimal movements initiated by the conveyor motor 1. Figure 13 depicts the brick used for experimenting with the emergency control scenario both for the fuzzy-BDI and classical-BDI agents. Figure **??** illustrates the mentioned components, such as the weighted brick, the ultrasonic sensor, and the conveyor belt 1, from the sensor viewpoint. Moreover, Figure **??** depicts the conveyor motor one and the weighted brick from the motor viewpoint.

Figure 13 The weighted brick used for experimenting with the emergency control scenario.



Stuck brick recovery required multiple ordinary bricks and stuck points to carry out this experiment in a controlled manner. Figure 15 depicts the stuck brick (bottleneck) locations on the first conveyor belt. Initially, these bricks are at least 5 cm away from each other. These bottlenecks were created by constraining the conveyor belt's wideness from 4 cm up to 2.5 cm using the LEGO parts, considering that the length of the bricks is 3 cm. The reverse actions are then performed by the Sort agent using Conveyor Motor 1. In the following subsubsection, the experimental setup regarding the second conveyor belt is introduced.

*6.2.2. **Second Conveyor Belt***

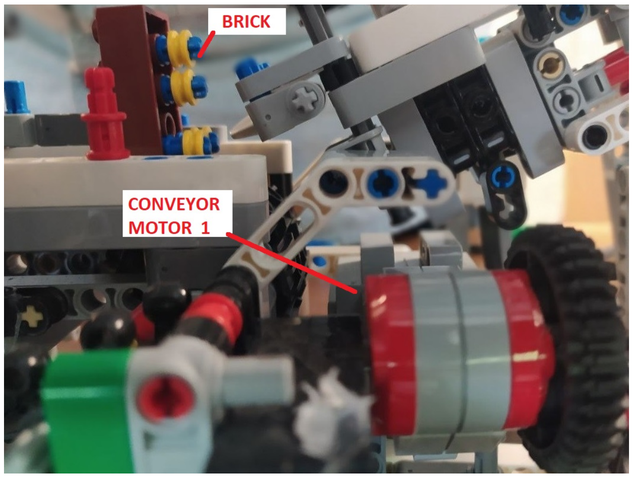
Two cautious setups were established for colour distinguishing and controlling the push force experiments on the second conveyor belt. Figure 16 depicts the conveyor belt 2 we carried out the mentioned experiments.

Colour distinguishing setup required the consideration of environmental influences. Therefore, we checked whether the sensor was operational or any component that may create a significant effect on our cause[39]. We also inspected whether light intensity affects the colour sensor's red, green and blue values[40]. Ultimately, we concluded that light intensity does not affect our setup. Moreover, as the bricks' wideness is 1.5 cm and the conveyor belt's wideness is 4 cm, we constrained the brick distance to the sensor. Otherwise, in case of a brick arrives at the Sort agent's phase and the furthest point, it becomes 2.5 cm away from the sensor. This exceeds the sensor's reading frame (out of scope), which should be less than 2 cm. Therefore, as Figure 17 illustrates, we used distance limiters to guide the incoming bricks to a specific distance interval `[0.5cm,3cm]`. We also measured the sensor's related *null* value in RGB mode. In other words, the corresponding and randomly altering R, G, and B values where there is not

[38]`https://github.com/micss-lab/FuzzyBDIAgents/blob/main/src/asl/shredAgent.asl#L43-L48`
[39]http://www.legoengineering.com/ev3-sensors
[40]https://education.lego.com/v3/assets/blt293eea581807678a/bltc5b2d4e8f6b6ccdc/5f8806d2f6a0a50f825b03da/ev3_user_guide_us.pdf

Figure 14 Viewpoints of the Conveyor Belt 1 and its components.

(**a**) Sensor Side



(**b**) Motor side

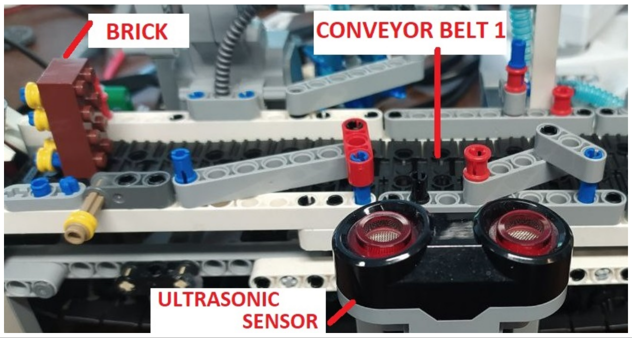
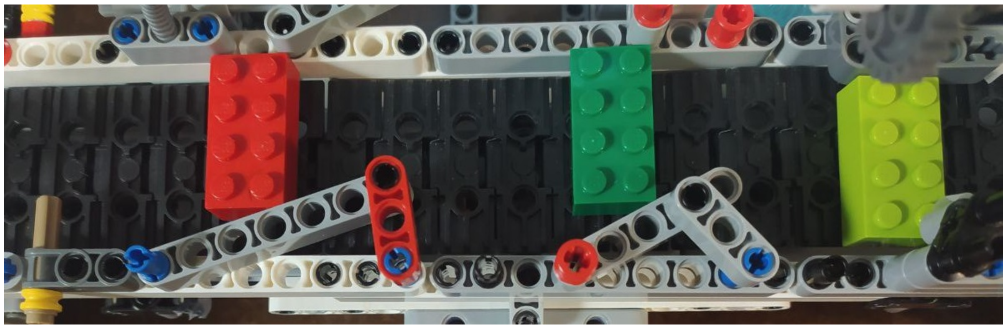


Figure 15 The sample bricks and stuck locations used for experimenting with the brick recovery scenario.



any product in front of the Sort agent's sensor are given as their means and standard deviations `[7.86,± 1,1913]`, `[10,7241,± 1,4605]`, `[7.4655,± 1.2871]`, respectively. In the following subsection, the colour-distinguishing experiment is mentioned.

Controlling push force required calibrating the push motor controlled by the Push agent based on the rotation angle and speed. As can be seen from Figure 18, the calibration of the push mechanism was done using the trial and error approach in three steps. Considering Figure 18, firstly, the push rod was set to the initial position with zero angles. Secondly, the correct angle was tried to be found by increasing the rotation angle of the push motor. Lastly, the configuration parameter (angle) that moves the push mechanism from the zero point to the border of the conveyor belt was found. When the fixed angle was

Figure 16 Conveyor belt 2.

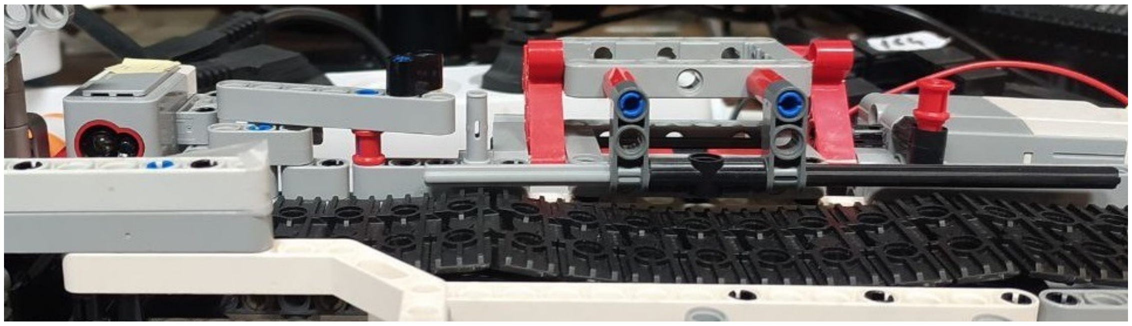
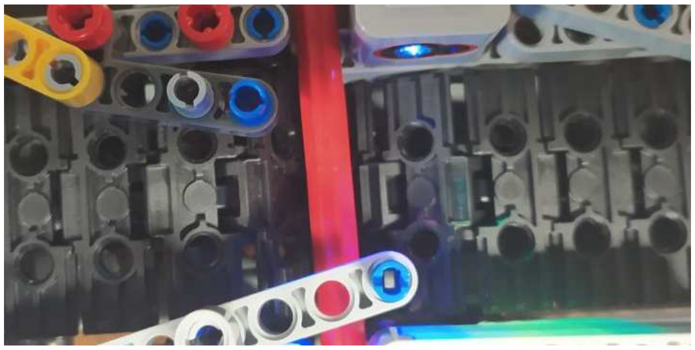


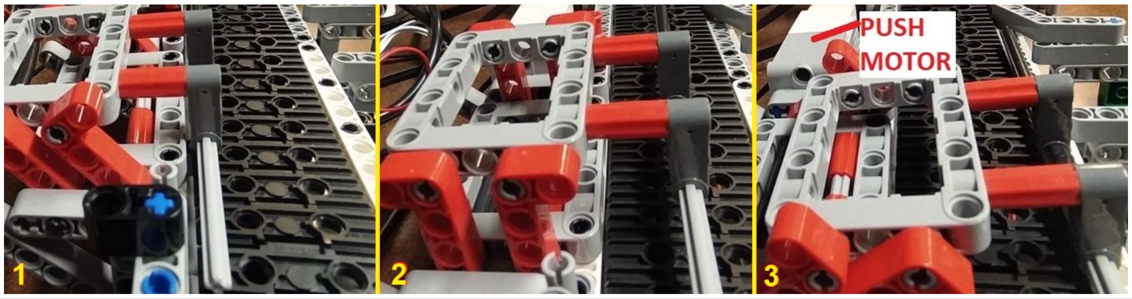
Figure 17 Distance limiter parts and the colour sensor.



set as the first parameter, then the rotation speed was arranged by the Push agent.

Lastly, it should also be noted that, for the end-to-end scenario, situations such as emergency control and stuck brick recovery were not considered as the aim is to measure the system's performance and process time based on regular operation.

Figure 18 Calibration of the push mechanism.



6.3. Experimental Results and Analyses

In this subsection, colour distinguishing, emergency control, stuck brick recovery, controlling push force, execution time, and end-to-end experiments are introduced, considering the performance and execution time evaluation. Their analysis results are given as well. Generally, our focus was evaluating the system performance and time consumption. Specifically, performance evaluation considers the system performance evaluation based on the experiments such as colour distinguishing, emergency control, stuck brick recovery, controlling push force and end-to-end process performance. Moreover, time evaluation focuses on evaluating the process and execution time aspects of the system based on the experiments, such as execution time evaluation and end-to-end process time. The results are then highlighted in Section 7.

*6.3.1. **Colour Distinguishing Experiment***

As mentioned, the colour-distinguishing effectiveness of the fuzzy BDI agent was evaluated and compared with the conventional BDI agent. Therefore, in this experiment, every five different colours, dark green, green, light green, red and spoiled red, were inputted 40 times to compare colours determining capabilities between two agents. Lastly, the chi-square test was carried out to indicate whether there was a significant difference between the fuzzy BDI and traditional BDI agents.

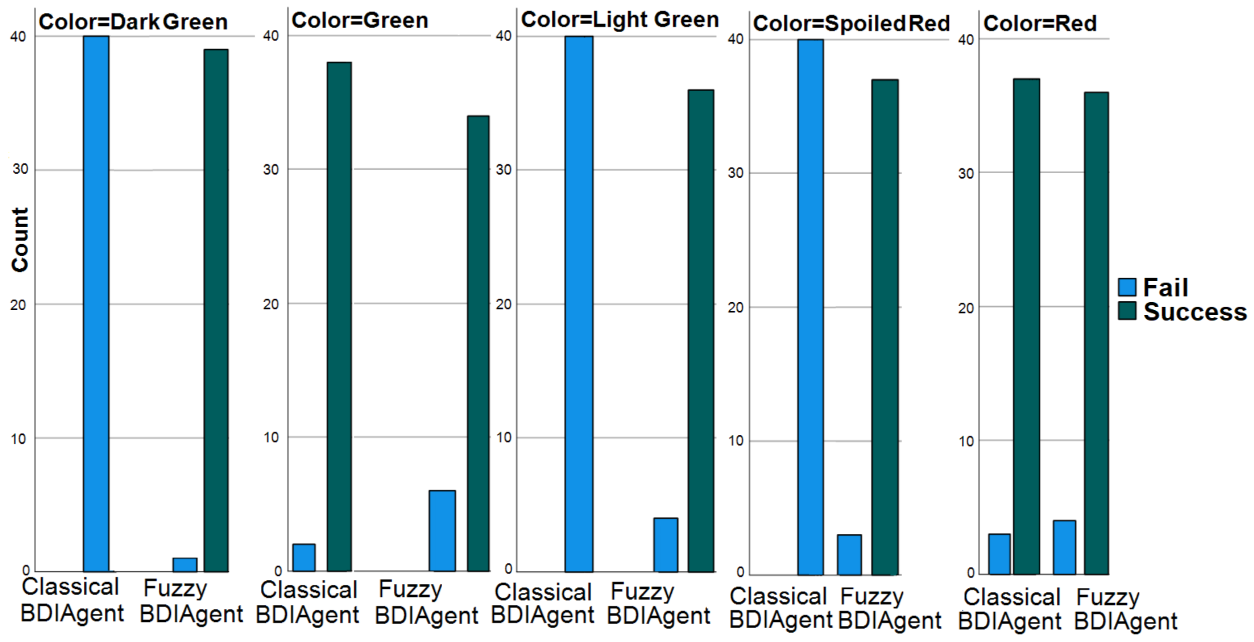
Table 5 Chi-Square Test results of the five different colours of the colour distinguishing experiment.

Chi-Square Tests						
Colour		Value	df	Asymptotic Significance (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
Dark Green	Pearson Chi-Square	76.098c	1	<.001		
	Continuity Correctionb	72.245	1	<.001		
	Likelihood Ratio	101.501	1	<.001		
	Fisher's Exact Test				<.001	<.001
	N of Valid Cases	80				
Green	Pearson Chi-Square	2.222d	1	.136		
	Continuity Correctionb	1.250	1	.264		
	Likelihood Ratio	2.315	1	.128		
	Fisher's Exact Test				.263	.132
	N of Valid Cases	80				
Light Green	Pearson Chi-Square	65.455e	1	<.001		
	Continuity Correctionb	61.869	1	<.001		
	Likelihood Ratio	84.096	1	<.001		
	Fisher's Exact Test				<.001	<.001
	N of Valid Cases	80				
Red	Pearson Chi-Square	.157f	1	.692		
	Continuity Correctionb	.000	1	1.000		
	Likelihood Ratio	.157	1	.692		
	Fisher's Exact Test				1.000	.500
	N of Valid Cases	80				
Spoiled Red	Pearson Chi-Square	68.837g	1	<.001		
	Continuity Correction	65.167	1	<.001		
	Likelihood Ratio	89.142	1	<.001		
	Fisher's Exact Test				<.001	<.001
	N of Valid Cases	80				
Total	Pearson Chi-Square	124.612a	1	<.001		
	Continuity Correction	122.293	1	<.001		
	Likelihood Ratio	135.933	1	<.001		
	Fisher's Exact Test				<.001	<.001
	N of Valid Cases	400				

Table 5 displays the results of the chi-square test for the five different colours, as well as the overall results. The classical BDI agent performs slightly better in primary colours like red and green. However, when considering side colours such as spoiled red, dark green, and light green, the fuzzy BDI agent outperforms the classical agent not only in side colours but also in the overall results. The null hypothesis, which states that there is no significant difference between the two samples, can be rejected with a confidence level of 99% for dark green, light green, spoiled red, and the total.

Figure 19 illustrates fail and success counts of the classical and fuzzy BDI agents for dark green, green, light green, spoiled red and red, respectively. In the following subsection, the emergency control experiment is mentioned.

Figure 19 Failure and success counts of the Classical and Fuzzy BDI Agents for the dark green, green, light green, spoiled red, and red of the colour distinguishing experiment.



6.3.2. ***Emergency Control Experiment***

In the experiment, the brick was put on the conveyor belt's predefined location for each iteration. After 1 second, the agents' emergency plan was triggered. In the end, if the brick was kept up, this counted as a success, and if it lost its balance and fell, this phenomenon was counted as a failure. If the brick hit the corners and dropped or stayed still, that iteration was redone as it could affect the brick's state.

40 different hand (object) movements with 70 samples were gathered to provide fairness for comparing both approaches. For each different hand movement, various conveyor belt speeds starting from 350 units to 1000 were experimented with, increased by 50 units. Table 6 displays the failure and success counts of the classical and fuzzy BDI agents. As described by Table 7, a chi-square test was carried out based on this data. The results of the Pearson Chi-Square test indicate a significant difference between the fuzzy BDI and classical BDI agents with a confidence level of 95%.

The bar chart in Figure 20 illustrates the failure and success counts of the classical and fuzzy agents. The classical BDI agent encountered 106 failures and 454 successes, while the fuzzy BDI agent had 77 failures and 483 successes out of 560 iterations. Therefore, the fuzzy BDI agent performs slightly better than the classical one in this experiment. In the following subsection, the stuck brick recovery experiment is mentioned.

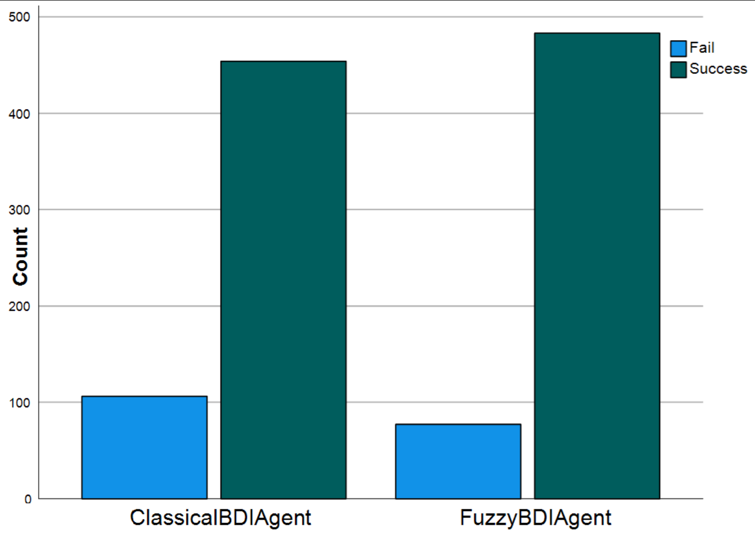
Table 6 Fail and success counts of the classical BDI and the fuzzy BDI agents of the emergency control experiment.

AgentType	Test Fail	Test Success	Total
ClassicalBDIAgent	106	454	560
FuzzyBDIAgent	77	483	560
Total	183	937	1120

Table 7 Chi-Square Test results of the Classical BDI and the Fuzzy BDI agents of the emergency control experiment.

	Value	df	Asymptotic Significance (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
Pearson Chi-Square	5.493	1	.019		
Continuity Correction	5.121	1	.024		
Likelihood Ratio	5.513	1	.019		
Fisher's Exact Test				.023	.012
N of Valid Cases	1120				

Figure 20 Failure and success counts of the classical and the fuzzy agents of the emergency control experiment.



6.3.3. ***Stuck Brick Recovery Experiment***

In order to carry out this experiment, 40 iterations were conducted for each brick sequence of both approaches, leading to 240 endeavours.

Table 8 presents the failure and success counts for both approaches in the brick recovery experiment. Table 9 provides the chi-square test results for the failure and success counts of the fuzzy BDI and classical BDI agents in this experiment.

Figure 21 displays the bar charts representing the results of the brick recovery experiment. It can be observed that the classical BDI agent failed to recover half of the stuck bricks in the one, two, and three-brick trials. The statistical tests conducted indicate that there is no significant difference between the classical and fuzzy BDI approaches in the one and two-brick trials. However, a significant difference exists between the approaches in the three-brick trial.

Table 8 Failure and success counts of the fuzzy BDI and the classical BDI agents for the brick recovery of the stuck brick recovery experiment.

Agent Type	Fail	Success	Total
Classical BDI Agent	19	101	120
Fuzzy BDI Agent	4	116	120
Total	23	217	240

Table 9 Chi-square test results of the brick recovery plan of the stuck brick recovery experiment.

Brick Count		Value	df	Asymptotic Significance (2-sided)	Exact Sig. (2-sided)
	Pearson Chi-Square	1.013	1	.314	
	Continuity Correction	.000	1	1.000	
1.00	Likelihood Ratio	1.399	1	.237	
	Fisher's Exact Test				1.000
	N of Valid Cases	80			
	Pearson Chi-Square	1.013	1	.314	
	Continuity Correction	.000	1	1.000	
2.00	Likelihood Ratio	1.399	1	.237	
	Fisher's Exact Test				1.000
	N of Valid Cases	80			
	Pearson Chi-Square	18.660	1	<.001	
	Continuity Correction	16.529	1	<.001	
3.00	Likelihood Ratio	20.872	1	<.001	
	Fisher's Exact Test				<.001
	N of Valid Cases	80			
	Pearson Chi-Square	10.819	1	.001	
	Continuity Correction	9.425	1	.002	
Total	Likelihood Ratio	11.669	1	<.001	
	Fisher's Exact Test				.002
	N of Valid Cases	240			

Thus, in order to evaluate the three stuck brick trial's joint success iterations for a fair comparison of both methods in terms of time, the Paired Simple T-test was also applied.

In other words, the successful trials corresponding to 97 trials in both approaches were considered for our evaluation. Table 10 pre-describes the means, and Table 11 shows the mean difference of approximately 11 seconds. The standard deviation of the differences is 7.74 seconds with a 95% confidence interval of 9.87 and 12.88. As a result of the paired sample t-test, the classical BDI agent is ahead of the fuzzy BDI agent regarding recovery time.

Table 10 The Statistics of Paired Samples Test of the Stuck Brick Experiment for recovery time.

	Mean	**N**	**Std. Deviation**	**Std. Error Mean**
FuzzyBDIAgent	19.1546	97	4.86985	.49446
ClassicalBDIAgent	7.8247	97	6.08620	.61796

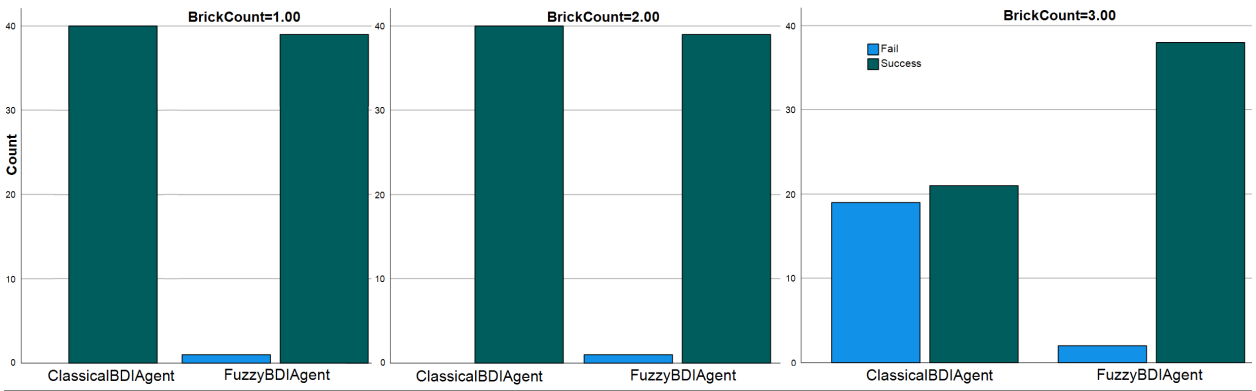
Table 11 The Paired Samples Test Results for the Stuck Brick Experiment for Recovery time.

		Paired Differences					t	df	Significance	
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Diff.	95% Confidence Interval of the Diff.			One-Sided p	Two-Sided p
					Lower	Upper				
Pair	FuzzyBDIAgent - ClassicalBDIAgent	11.32990	7.74021	.78590	9.76990	12.88989	14.416	96	<.001	<.001

The classical BDI agent saves by one and by two bricks faster than the fuzzy BDI agent by pulling the conveyor belt back sequentially. The reason is that the fuzzy BDI agent begins by making the conveyor belt minimal back-and-forth movements by smoothly switching to maximal ones. These initial slow and minimal movements were our implementation choice and not the effect of the computation time. Howbeit, we also measured the computation time, which will be mentioned in subsection 6.3.7. Therefore, such a movement was required when the number of bricks was increased as the bricks cause to keep stuck by colliding with each other as a part of the solution. When the classical BDI agent begins instant reversing the conveyor belt, the same movement was not able to perform better success in recovering the multiple bricks. On the other hand, the fuzzy BDI agent created variable displacements among the bricks that made them resolve from their stuck state, even though the initial minimal movements of the conveyor belt caused a bit of time loss.

Consequently, when the difference between the two methods is inspected by one and by two bricks, it can be said that there is no significant difference between the classical BDI and fuzzy BDI agents. However, if we examine the three stuck bricks iteration, it can be seen that there is a significant difference between the two approaches where the fuzzy BDI performs better recovery. In other words, the fuzzy BDI provides superior performance when the number of stuck bricks is increased. In the following subsection, the controlling push force experiment is mentioned.

Figure 21 Representation of the failure and success counts of the stuck brick recovery experiment.



6.3.4. ***Controlling Push Force Experiment***

In this experiment, it was tried to comprehend whether there is a significant difference between the expected behaviour of the Ideal Agent and the behaviour of the fuzzy and classical BDI agents using the ANOVA method. To assess the homogeneity of variances, Levene's tests were conducted as indicated by Table 13. Descriptive statistics for this experiment can be found in Table 12. Table 14 shows the ANOVA test results.

Upon examining the p-values, significant differences between the groups are observed. Therefore, multiple comparisons need to be conducted to understand the source of these differences, as shown in Table 15. According to the results of multiple comparisons, there is a significant difference between the Ideal Agent and the Classical BDI agent but no significant difference between the Ideal Agent and the Fuzzy BDI agent. Lastly, Figure 22 illustrates the means plot, which describes the distance to the Ideal Agent. The Classical BDI Agent deviates significantly from the Ideal Agent, while the Fuzzy BDI Agent is slightly closer to the Ideal Agent. In the following subsection, the execution time measurements are mentioned.

Table 12 The statistical descriptives for the Push Force experiment.

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean Lower Bound	95% Confidence Interval for Mean Upper Bound	Minimum	Maximum
ClassicalBDIAgent	200	540.0000	280.70264	19.84867	500.8593	579.1407	200.00	800.00
ExpectedValue	200	680.0000	75.02094	5.30478	669.5392	690.4608	600.00	800.00
FuzzyBDIAgent	200	687.2604	166.83330	11.79690	663.9974	710.5234	250.00	1000.00
Total	600	635.7535	204.68114	8.35607	619.3427	652.1642	200.00	1000.00

Table 13 Test results of Homogeneity of Variances for the Push Force experiment.

	Levene Statistic	**df1**	**df2**	**Sig.**
Based on Mean	489.205	2	597	<.001
Based on Median	81.753	2	597	<.001
Based on Median and with adjusted df	81.753	2	310.148	<.001
Based on trimmed mean	463.008	2	597	<.001

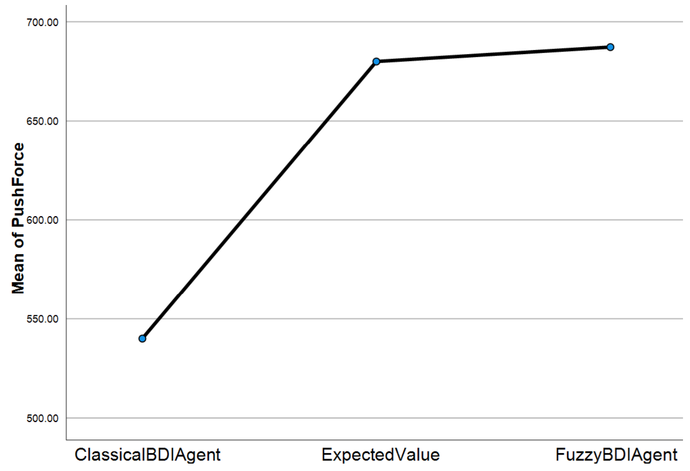
Table 14 The ANOVA test for the Push Force experiment.

	Sum of Squares	**df**	**Mean Square**	**F**	**Sig.**
Between Groups	2755889.357	2	1377944.679	36.825	<.001
Within Groups	22338836.872	597	37418.487		
Total	25094726.229	599			

Table 15 Tukey HSD Multiple Comparision for the Push Force experiment.

	(I) AgentType	**(J) AgentType**	**Mean Difference (I-J)**	**Std. Error**	**Sig.**
Tukey HSD	**ClassicalBDIAgent**	**ExpectedValue**	-140.00000*	19.34386	<.001
		FuzzyBDIAgent	-147.26041*	19.34386	<.001
	ExpectedValue	**ClassicalBDIAgent**	140.00000*	19.34386	<.001
		FuzzyBDIAgent	-7.26041	19.34386	.925
	FuzzyBDIAgent	**ClassicalBDIAgent**	147.26041*	19.34386	<.001
		ExpectedValue	7.26041	19.34386	.925

Figure 22 Means Plot that shows distances for the compared types.



6.3.5. ***End-to-End Process Performance Experiment***

In this experiment, the classical and fuzzy BDI approaches were evaluated based on the end-to-end processes and their impact on product success. 30 series that consist of 20 product sequences were randomly generated, and each 30 series was used to evaluate both approaches end-to-end. Each series was repeated thrice by measuring the end-to-end process time.

Table 16 present the group statistics of the end-to-end process performance experiment. The collected data were analyzed using an independent samples t-test to evaluate the end-to-end process performance experiment, and the results are displayed in Table 17. There is a significant difference between the fuzzy and classical BDI with the 99% confidence interval.

Figure 23 illustrates the mean count of accepted products by sequence and agent type, while Figure 24 displays the rejected product count by sequence and agent type.

Considering the statistics, it can be concluded that there is a significant difference between the fuzzy and classical BDI agents. The fuzzy BDI agent performs a superior outcome compared to the classical one. In the following subsubsections, time evaluations, which are execution time and end-to-end process time, are mentioned.

Table 16 The group statistics of the end-to-end process performance experiment.

	Agent Type	**N**	**Mean**	**Std. Deviation**	**Std. Error Mean**
Accepted Products Count	ClassicalBDIAgent	90	3.0556	1.49427	.15751
	FuzzyBDIAgent	90	6.1000	1.17129	.12346
Rejected Product Count	ClassicalBDIAgent	90	2.4222	1.48383	.15641
	FuzzyBDIAgent	90	.0667	.25084	.02644

Table 17 The Independent Samples t-test results of the end-to-end process performance experiment.

		t-test for Equality of Means			**t-test for Equality of Means**		**t-test for Equality of Means**	
			Significance				**95% Confidence Interval of the Diff.**	
		df	**One-Sided p**	**Two-Sided p**	**Mean Diff.**	**Std. Error Diff.**	**Lower**	**Upper**
Accepted Products Count	**Equal variances assumed**	178	<.001	<.001	-3.04444	.20013	-3.43938	-2.64951
	Equal variances not assumed	168.395	<.001	<.001	-3.04444	.20013	-3.43953	-2.64935
Rejected Product Count	**Equal variances assumed**	178	<.001	<.001	2.35556	.15863	2.04252	2.66859
	Equal variances not assumed	94.083	<.001	<.001	2.35556	.15863	2.04060	2.67051

6.3.6. ***End-to-End Time Experiment***

In this experiment, the end-to-end processes of both the classical and fuzzy BDI approaches were experimented with regarding time success. 30 series that consist of 20 product sequences were randomly generated, and each 30 series was used to evaluate both approaches end-to-end. Each series was repeated thrice by measuring the end-to-end process time.

Table 18 represents the group statistics of the end-to-end experiment in terms of process time. The collected data were subjected to an independent samples t-test to evaluate the experiment, and the results are revealed in Table 19. The results point out that there is a significant difference between the fuzzy and classical BDI approaches with a 99% confidence interval.

Figure 25 represents the multiple timelines by sequence by agent type. When assessing the statistical results, it can be inferred that there is a significant difference between the two approaches. Specifically, the classical BDI is a bit faster than the fuzzy BDI agent as it is approximately 30 seconds quicker.

In the following subsection, the execution time experiment is mentioned.

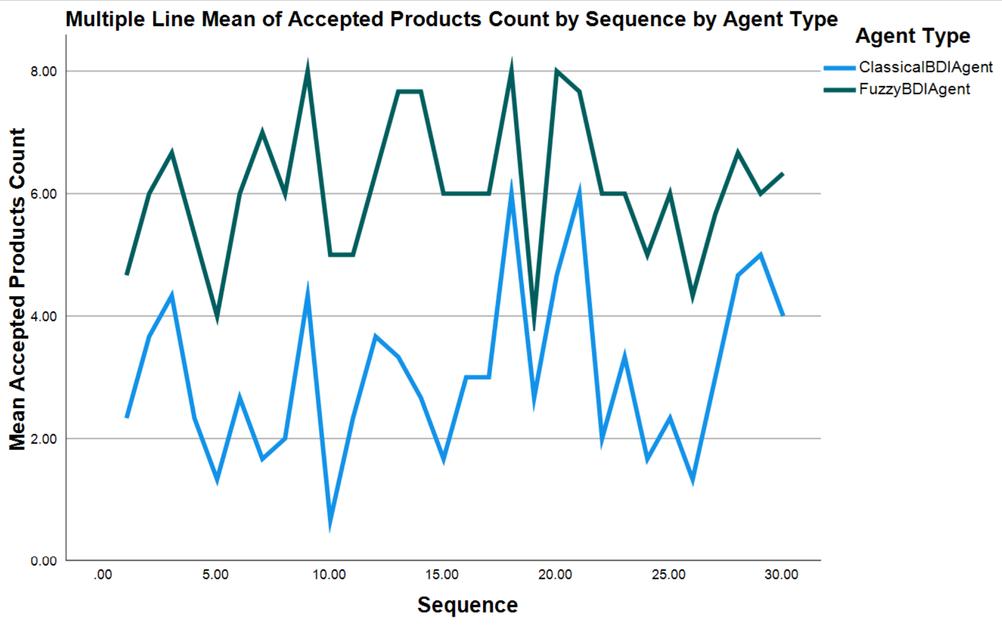
Table 18 The group statistics of the end-to-end time experiment.

	Agent Type	N	Mean	Std. Deviation	Std. Error Mean
Time	ClassicalBDIAgent	90	277.8778	39.33601	4.14638
	FuzzyBDIAgent	90	305.9000	27.89106	2.93998

Table 19 The Independent Samples t-test results of the end-to-end time experiment.

		t-test for Equality of Means			t-test for Equality of Means		t-test for Equality of Means	
		df	Significance		Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
			One-Sided p	Two-Sided p			Lower	Upper
Time	Equal variances assumed	178	<.001	<.001	-28.02222	5.08291	-38.05273	-17.99171
	Equal variances not assumed	160.434	<.001	<.001	-28.02222	5.08291	-38.06025	-17.98419

Figure 23 Multiple Line Mean of Accepted Products Count by Sequence by Agent Type



6.3.7. ***Execution Time Evaluation***

As mentioned in subsection 6.3.3 and to explore that the membership functions do not cause computation costs for the fuzzification of the BDI agents, computation costs of the classic and fuzzy BDI agents' colour reading methods were measured 40 times with different sample sizes, such as 1, 5, 10 and 15. This part was selected as it has the most complex fuzzy computation and is often called the data sampling method by the *!+samplecolour* plan (recall Sort Agent's code[41]). As mentioned, the software is run by the RaspberryPI 3 B+ board.

[41]https://github.com/micss-lab/FuzzyBDIAgents/blob/main/src/asl/sortAgent.asl#L43

Figure 24 Multiple Line of Rejected Product Count by Sequence by Agent Type

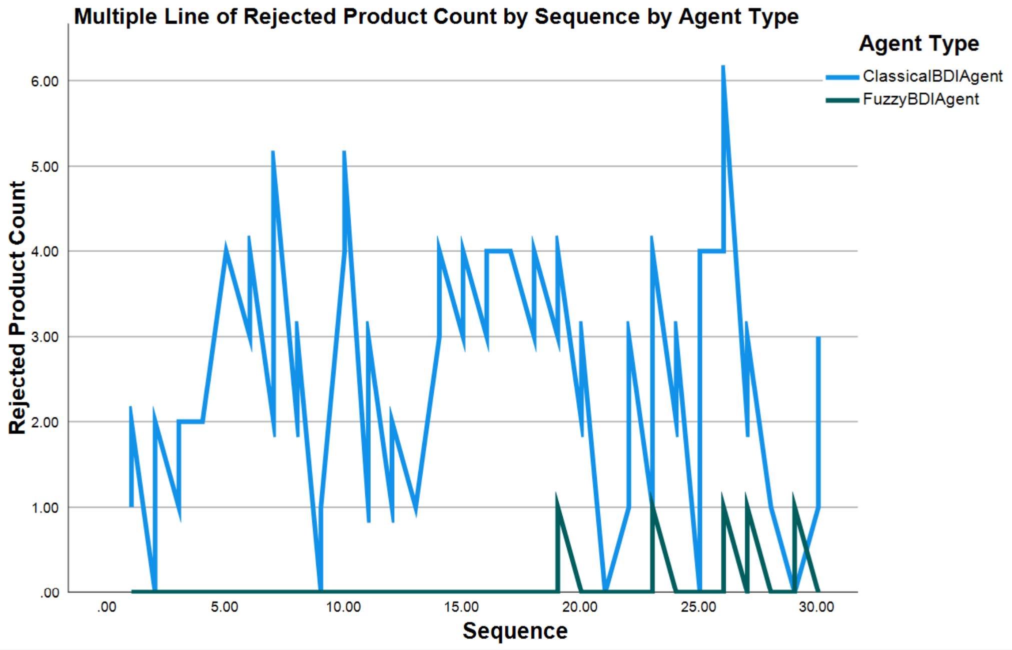
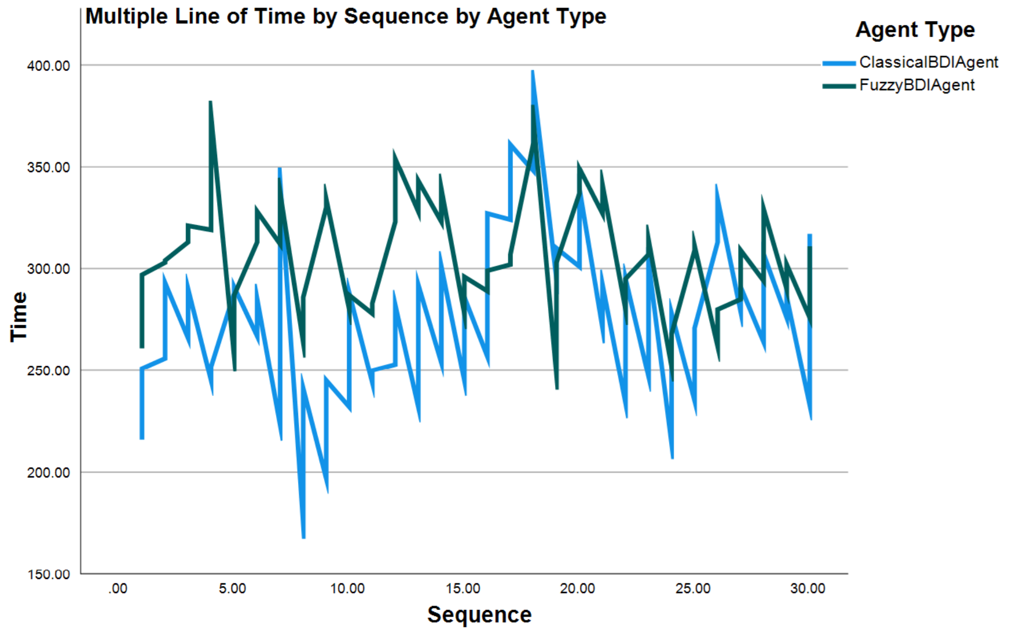


Figure 25 Multiple Line of Time by Sequence by Agent Type

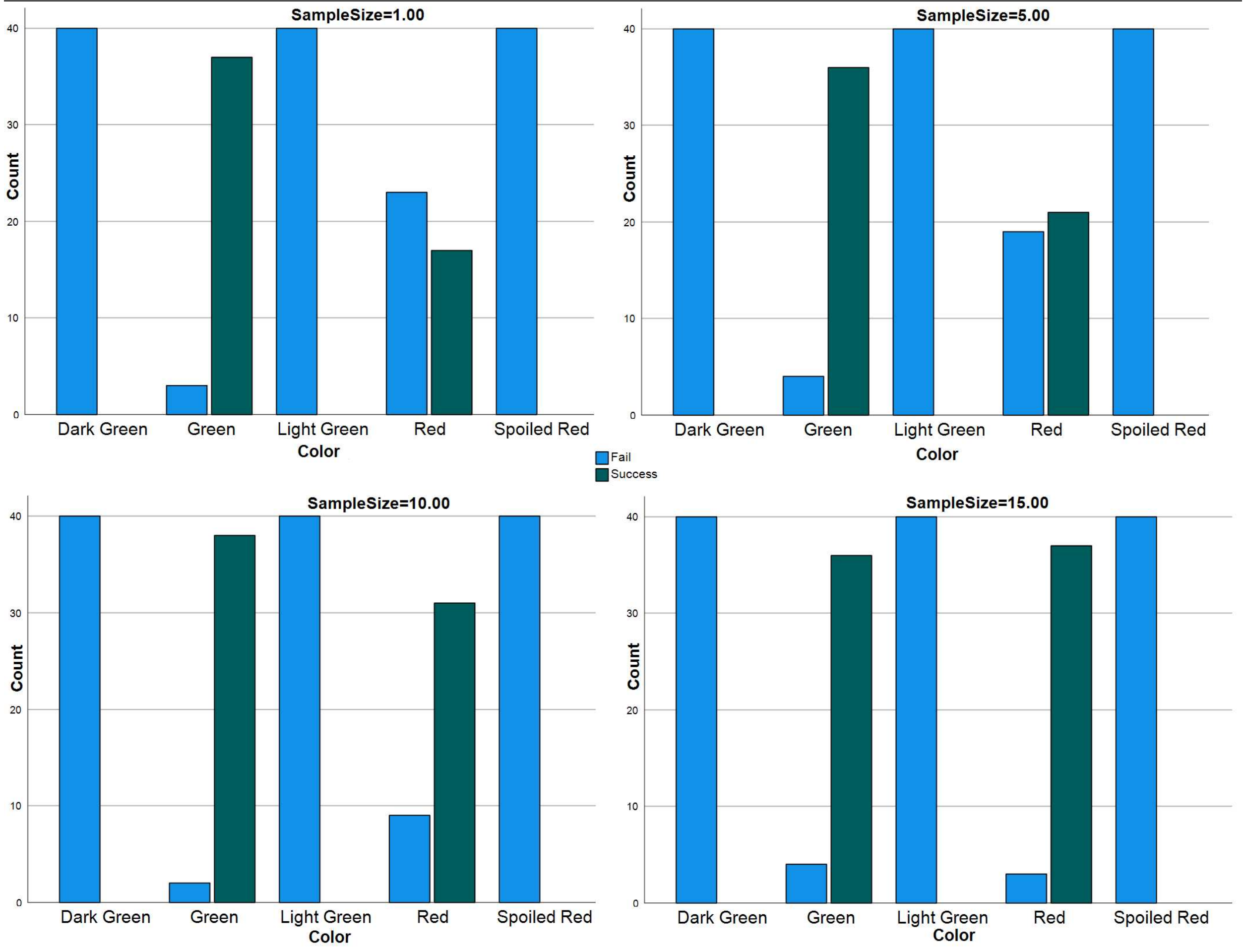


Initially, the success rate of the sampling sizes in deciding on the correct colour was compared. Secondly, it was observed that the best result was achieved when using a sample size of 15 on the classical BDI agent's side.

Subsequently, the execution time of both sizes was compared, considering the sufficient performance of both sides for detection, where the fuzzy BDI agent has one sample size, and the classical BDI agent has 15 sample sizes.

At first, the sample for the classical BDI agent was set to 1 and increased to 5, 10 and 15. As seen in Table 20, the increase in sample size created a significant difference. Moreover, Figure 26 shows that the most significant and favourable outcome was obtained with a sample size of 15, particularly for successfully detecting red bricks. In other words, increasing the sample size positively affected colour detection success. Eventually, 15 sample size selection for the classical BDI agent was compared with the one sample size operation of the fuzzy BDI agent. Since it was already observed that increasing the sample size on the fuzzy BDI side did not yield any significant difference, the decision was made to use a single sample size for the fuzzy BDI agent.

Figure 26 Failure and success counts of the Classical and Fuzzy BDI Agents based on the sampling time the dark green, green, light green, spoiled red, and red of the execution time experiment.



Then, the independent samples t-test, a parametric test, was applied as a statistical analysis to compare the fuzzy BDI agent with one sample size and the classical BDI agent with 15 sample size. Table 21 represents the sample size, mean standard deviation, and standard error mean. Table 22 provides the

Table 20 Chi-Square Tests results of the execution time experiment using different sample sizes for the classical BDI agent.

Chi-Square Tests				
SampleSize		**Value**	**df**	**Asymptotic Significance (2-sided)**
1.00	**Pearson Chi-Square**	136.327b	4	<.001
	Likelihood Ratio	157.444	4	<.001
	N of Valid Cases	200		
5.00	**Pearson Chi-Square**	133.382c	4	<.001
	Likelihood Ratio	157.687	4	<.001
	N of Valid Cases	200		
10.00	**Pearson Chi-Square**	160.726d	4	<.001
	Likelihood Ratio	199.184	4	<.001
	N of Valid Cases	200		
15.00	**Pearson Chi-Square**	172.495e	4	<.001
	Likelihood Ratio	215.179	4	<.001
	N of Valid Cases	200		
Total	**Pearson Chi-Square**	579.321a	4	<.001
	Likelihood Ratio	703.641	4	<.001
	N of Valid Cases	800		

t-test results.

Based on the t-test results in Table 22, no significant difference was found between the fuzzy BDI agent with one sample size and the classical BDI agent with 15. This indicates that the execution times of the two approaches were not significantly different. Therefore, from a statistical standpoint, the choice of a single sample size for the fuzzy BDI agent and 15 sample sizes for the classical BDI agent did not result in a significant difference in terms of execution time.

In the following subsection, the mentioned experiments' results will be spotlighted. The research questions will be answered according to the research findings, and threats to the validity will be given, considering the implications and limitations of the experiments. Technical spottings and potential enhancements/extensions will be discussed for further research.

Table 21 Group Statistics of the execution time measurement for both approaches of the execution time experiment.

	AgentType	**N**	**Mean**	**Std. Deviation**	**Std. Error Mean**
Time	**ClassicalBDIAgent**	40	.0433	.03430	.00542
	FuzzyBDIAgent	40	.0441	.01212	.00192

Table 22 The Independent Samples t-test for execution time experiment.

		Levene's Test for Equality of Variances		t-test for Equality of Means							
		F	Sig.	t	df	Significance		Mean Diff.	Std. Error Difference	95% Confidence Interval of the Diff.	
						One-Sided p	Two-Sided p			Lower	Upper
Time	Equal variances assumed	13.047	<.001	-.135	78	.446	.893	-.00078	.00575	-.01223	.01067
	Equal variances not assumed			-.135	48.589	.447	.893	-.00078	.00575	-.01234	.01078

7. Discussion

In this section, the experimental results are highlighted, the research questions are examined, threats to the validity are mentioned and future directions are discussed.

7.1. Highlighting the Results

In this subsection, the experimental results mentioned in subsection 6.3 are highlighted. Begin with the colour-distinguishing experiment; at first, it aimed to increase the distinguishing ability of the BDI agent. The classical BDI approach was not enough to approximately precisely separate the different colours within the same type. In addition, the classical BDI agent is not only to distinguish the colours, but it also detects them as another type of colour that can lead to unwanted results both from the system and human perspectives. The precision was tried to be improved by increasing the sampling size to 1, 5, 10 and 15, especially for detecting the red-coloured bricks. Even though 15 sample sizes enabled us to successfully detect the primary colours, such as Red and Green, it did not provide the desired success on the side colours, such as separating the Red and the Spoiled Red products. Moreover, the green colours are also diverged as Light Green, Middle Green and Dark Green. Labelling all of them as only Green may not be the desired case for different scenarios and process phases. In this regard, we needed an approach that generalizes, specifies and harnesses these values by allowing logical statements despite the noise in the data. Therefore, the R, G and B values were gathered from the sensor API, and their fuzzy functions were created based on our expert opinion and manual interpretation of the data, which was achieved using one sample size. The slight drop in the fuzzy BDI agent to detect the primary colours might be because of the variable reading distance between `[0.5cm,3cm]`. This variable reading distance puts additional shifts in the data that require deep knowledge. Therefore, the fuzzy functions could be improved using machine learning techniques by aggregation and classification methods to detect better the boundary cases among the fuzzy sets and fine-tuning.

In the emergency control experiment, it was devised that the conveyor belt might need to be stopped or slowed down during the run-time. The ultrasonic characteristic may not allow linearly slowing down the conveyor belt. Moreover, the dynamic operation may harm both products and people nearby. The instant change in the speed may not be tolerable using the traditional logic as the speed is halved, which is a sharp change. Therefore, fuzzified actions can be preferred according to the distance change to the object approaching the conveyor belt. As another parameter, the conveyor belt's speed can be set to different values ranging from 350 to 1000 units. The fuzzy functions may require to be tuned according to various speed settings. At 1000 and 850 speed levels, the classic BDI approach showed a bit more success but under-performed in total. This could be because of our initial fuzzy functions setting for 400 units. The success of the fuzzy-BDI can be improved by fine-tuning the membership functions for specific settings or using different fuzzy function forms more suitable for motor control, such as Gaussian or bell-shaped. Alternatively, the fuzzy hedges can also be used (Zadeh, 1972).

A temporal approach was followed in the stuck brick recovery experiment to detect the stuck bricks state. Firstly, one brick-stuck state was experimented with, and a shorter recovery time compared to the classical BDI approach resulted compared to the fuzzy BDI one. Secondly, two bricks stuck state experimented on the classic BDI agent approach, and a slight drop but still faster recovery time was observed compared to the fuzzy BDI agent. Lastly, in the case of the three bricks stuck experiment, the classical BDI agent approach became ineffective as the number of stuck bricks increased. The bricks started colliding with each other, and reversing the conveyor belt instantly provided a kind of deterministic movement that led the system to deadlock states, sometimes resulting in time exceeds. On the other hand, the fuzzy BDI begins with minimal back-and-fort fuzzified actions as our flexible design choice is given thanks to fuzzy logic. This action creates a wave trend movement (recall Figure 12) that causes sequential and variable displacements among the bricks. In other words, the bricks' position and rotation were often influenced, increasing the speed smoothly. As a drawback, the necessity of minimal to maximal movements for creating sequential and variable position changes created a bit of time loss. As another design choice, the frequency of the minimal movements and transition from minimal to maximal ones can be increased by modifying the boundaries of the fuzzy sets and creating more overlapped regions. However, this also depends on the system requirements and product conditions. As a result, the fuzzy BDI provided superior performance when the number of stuck bricks was increased.

In the controlling push force experiment, the Push agent dynamically arranged the push mechanism's hitting speed on the fuzzy BDI approach. At the same time, it was static on the classical BDI agent approach. The membership degree that belongs to the highest linguistic set was selected and multiplied by the pre-defined speed. As this value depends on the colour-distinguishing performance of the Sort agent, few hitting force differences among the fuzzified actions resulted. As discussed for the colour distinguishing experiment, this might occur because of the variable reading distance between `[0.5cm,3cm]`. However, as an improvement, type-2 fuzzy sets (Mendel, 2007) for further enhancement of the perceptions and/or different fuzzy operators (Cordón et al., 1997) to approximate the ideal expected values for further enhancement of the fuzzified actions. Lastly, we compared our classical BDI and fuzzy BDI agents according to the ideal agent that always satisfies the expected value. As a result, the fuzzy BDI agent showed a closer outcome with respect to the ideal agent.

As previously discussed in the stuck brick recovery experiment, we were motivated to measure the execution time. Therefore, we compared the execution times of the Sort Agent for the colour sampling plan. Despite the fuzzy logic enhancements and extensions on the different layers, there was no significant difference between the execution overhead between the two methods. The fuzzy BDI logic approach only puts approximately %0,2 overhead (recall Table 21) compared to the classical BDI method. However, this minimal difference may cause considerable delays for larger systems. In this regard, we also carried out the end-to-end experiment. In the end-to-end experiment, the system's performance and time metrics were evaluated. As mentioned, the classical BDI agent is a bit faster than the fuzzy BDI agent, around 30 seconds. This time delay also depends on the selected embedded hardware's computation capability. The time delay can be reduced by selecting industrial-level sensors, actuators and hardware. Moreover, optimal device-specific APIs and device libraries can also be considered.

Despite the short time overhead, the end-to-end performance of the fuzzy BDI approach performs a superior outcome compared to the classical BDI method. When the product quality, cost, and people's health are considered, this may be a considerable option to improve the efficiency of the complex systems using a logical approach based on the domain experts' opinions. Using the fuzzy-BDI method, expert and domain knowledge can be extracted from the domain specialists and then integrated into the system. In this way, the system's performance can be improved using a tangible method in the short term. The system can then gradually gain mid- and long-term enhancements and extensions using different symbolic and sub-symbolic approaches. In the subsequent subsection, the research questions are examined.

7.2. Examining the Research Questions

R.Q.1 (Feasibility) How can we enhance and deploy the BDI agents using fuzzy logic to increase their capacity to deal with run-time uncertainty?

R.A.1: As mentioned, fuzzy logic was selected as it is an enhancement method to deal with uncertainty. Usually, the BDI agents use traditional logic. As traditional logic is a subset of fuzzy logic, the fuzzy sets needed to be defined in a feasible method. Since fuzzy logic is a platform-independent mathematical model that makes it an inherently feasible enhancement, it was represented using triangular functions. In the design time, these membership functions were implemented using Equation 1 from scratch based on expert knowledge. In our study, we used Jason and Java BDI agent development environment, but the fuzzy logic can also be applied to other frameworks. Lastly, the reference architecture's corresponding layers are then extended one by one.

R.Q.2 (Integrability) How agents' sensing, reasoning, plan selection and execution phases can be *fuzzified* and bind to the device-specific software?

R.A.2: In order to achieve BDI agents and fuzzy logic integration, considering the impact on the device-level sensing and actuation, the designed membership functions were integrated into the middleware layer and wrapped. The fuzzy membership functions were added to the perceptions as an intermediate step to convert the crisp numbers to membership degrees. This way, the beliefs were fuzzified and added to the belief base. The reasoning cycle then uses the fuzzified beliefs and selects the plan with the highest triggering degree (i.e., maximum membership degree) in a fuzzified manner using the Prolog rules. The selected plan's triggered degree is used within the plan's actions to create fuzzified actions. In this way, the fuzzy logic and Jason's BDI agents were integrated. Lastly, the Java API allows the BDI agents to manipulate the input/output ports of the selected device.

R.Q.3 (Applicability) How the proposed approach can be implemented and deployed on a concrete physical setup?

R.A.3: The application of the fuzzy-BDI approach was performed using Jason and its Java environment. The fuzzy logic enhanced BDI agents perform fuzzified sensing, reasoning and actions on a concrete case study both for the regular operation and in case of uncertainty according to the context. The entire complex system is autonomously controlled and coordinated by a network of fuzzy-BDI agents, collaborating with each other throughout the run-time. Moreover, as also demonstrated in subsubsection 6.1, the fuzzy-BDI method has been applied to modular and end-to-end scenarios.

R.Q.4 (Effectiveness) Are fuzzy-BDI agents effective for the uncertainty of agent-based CPS?

As mentioned in Section 6, multiple modular and end-to-end experiments showed that there is a significant difference between the fuzzy-logic and traditional logic BDI agent approaches. Generally, the fuzzy BDI agent performs approximately 3 times better than the classical one despite around 10% more computation time. When the modular experiments and end-to-end ones are considered, mostly the fuzzy BDI agent approach outperforms the classical BDI agent method. In other words, the fuzzy logic and BDI agent-based operation impact a complex CPS. At the same time, a bit of additional computation costs can be expected or tolerated, considering the continuous evolution of embedded technologies. Therefore, it can be concluded that fuzzy-BDI agents offer several advantages for the development of complex and heterogeneous systems. In the subsequent subsection, we discuss the threats to the validity that need to be considered in this study.

7.3. Threats to the Validity

The section discussing threats to validity addresses potential limitations or factors that could affect the validity of the experiments conducted and the results obtained. These threats highlight areas of concern that could introduce biases or inaccuracies in the study's findings. By acknowledging these threats, researchers aim to provide a transparent assessment of the study's limitations and potential impact on the validity of the results.

7.3.1. Conclusion Validity

In the context of conclusion validity, the focus is on examining the relationship between the subjects (experimental units, participants) and the obtained results. The goal is to establish a statistical relationship that supports the observations and hypotheses with a meaningful level of significance. In order to achieve this, a range of scenarios were derived from different experiments conducted on the system. The statistical tests used in these experiments were carefully selected to ensure reliable and valid results.

Additionally, a complex, modular, and heterogeneous case study was established to provide a versatile platform for conducting various experiments at different stages. This comprehensive setup allows for the exploration of different aspects and variables of the system, enabling researchers to perform in-depth analyses and draw meaningful conclusions.

By carefully designing the experiments, selecting appropriate statistical tests, and utilizing a robust case study, the study ensures a strong foundation for establishing the relationship between the subjects and the obtained results. This contributes to the conclusion validity of the research and enhances the reliability of the findings.

7.3.2. Internal Validity

Internal validity refers to the extent to which a causal relationship between the treatment (independent variable) and the outcome (dependent variable) can be established while minimizing the influence of uncontrolled events or entities. In order to ensure internal validity in our study, we took various measures to account for external effects and potential confounding factors.

One potential external effect we considered was light, as it could impact the performance of the sensors used in the experiment. To address this, we examined the technical documents of the sensors [42] and specifically assessed the noise data collected by the colour sensor in dark environments with a light source. We found that the difference in the sensor readings under these conditions did not show a significant variation, indicating that the presence of light did not significantly affect our experimental results.

Furthermore, the choice of the case study was not incidental. In line with previous research conducted by (Barbosa et al., 2016; Vieira et al., 2020; Leitão et al., 2016; Leitão & Barbosa, 2016), we decided to utilize an industrial-like case study for our analyses.

These studies also focused on conveyor belts and provided insights into particular scales and complexities. In our case study, we extended the functionality by incorporating additional operations such as push and press actions. This allowed us to assess a broader range of functionalities and evaluate the performance of the system under different scenarios.

The case study itself was carefully designed to have multi-stage, multi-component, and modular features, providing a controlled environment for conducting experiments. This design enabled us to isolate and examine specific aspects of the system while maintaining internal validity.

To enhance the understanding of our study, demonstration videos were provided, offering visual insights into the experimental setup and results. These videos serve as supplementary evidence and contribute to the transparency and credibility of the research.

By considering external effects, selecting a relevant case study, conducting controlled experiments, and providing demonstration videos, we aimed to establish a strong internal validity in our study and ensure the reliability and integrity of the observed causal relationships.

7.3.3. Construct Validity

Construct validity concerns the generalization of results based on the conducted experiments. In this study, both the fuzzy-BDI agent approach and the comparison methods are logical approaches that have been extensively studied for many years. They operate at the same level of abstraction, using BDI agent programming and the same API, hardware, sensors, and actuators. To mitigate the possibility of obtaining results by chance, we conducted multiple repetitions of our experiments. The measurements were based on high-quality sampling within the system, which we also aimed to enhance. Therefore, generalization is not expected to have a negative impact on construct validity.

[42] $https://www.mikrocontroller.net/attachment/338591/hardware\_developer\_kit.pdf$

7.3.4. External Validity

External validity refers to the extent to which the findings of a study can be generalized to real-world situations or other contexts beyond specific experimental conditions. In our study, we took measures to address potential threats to external validity and ensure the relevance and applicability of our experiment results to industrial techniques.

To mitigate this threat, we carefully selected a case study that closely resembles an industrial production line system. In making our decision to focus on an industrial-like case study, we relied on the insights and findings from various studies in the field, including the works of (Alves et al., 2019; Sakurada et al., 2019; Barbosa et al., 2018; Rodrigues et al., 2013; Peres et al., 2018; Ribeiro et al., 2018; Vieira et al., 2020). These studies provided insights into industrial-oriented scenarios that closely aligned with our study objectives.

Additionally, we opted to use LEGO as the physical platform for our case study due to its flexibility and adaptability to meet the physical requirements of the system (Karaduman et al., 2023a). This allowed us to modify and customize the case study according to the specific needs and constraints of industrial techniques.

Through our experimental observations and analysis, we did not identify any significant threats that would undermine the validity of the experiment results in terms of their generalizability to industrial techniques. This suggests that the chosen case study, combined with the use of LEGO as a flexible platform, provided a solid foundation for drawing meaningful conclusions that can be applied in industrial contexts. In the next subsection, future directions are discussed.

7.4. Future Directions

As uncertainty is a fact that should be considered even during the design time, we believe that multi-logical modelling (MLM) techniques (Calegari et al., 2020, 2021) supported with learning methods (Bosello & Ricci, 2020) should be used in a combinatorial manner for short-, medium- and long-term objectives at the system level. The modelling term refers to the mathematical formulation (modelling for enhancement) to leverage the systems' smartness. In this way, the CPS can become more sustainable to uncertainties. Moreover, as these systems are highly complex because of their structural and behavioural aspects, they should be tackled using the multi-paradigm modelling (MPM) techniques (Challenger & Vangheluwe, 2020) (modelling for abstraction). In this regard, *the MLM suggests combining proper logic paradigms to mitigate uncertainty by providing system-level enhancements and benefiting from the MPM philosophy of modelling the system in an appropriate level of abstraction employing well-fit formalisms to reduce the complexity.*

However, there is still a need for research on the integration of BDI agents and CPS. In their work, (Menegol et al., 2018b) underscore the lack of comprehensive evaluations for agent-oriented programming approaches that integrate well-established practices for developing systems with embedded technologies. Moreover, (Leitao et al., 2016) suggest that in order to demonstrate the effectiveness of Multi-Agent Systems (MAS) in the domain of CPS, it is binding to carry out evaluations through the implementation of complex systems. Furthermore, comprehensive systematic literature reviews (SLRs) such as (Calegari et al., 2021; Tepjit et al., 2019) indicate the significant potential of logic-based approaches, such as fuzzy logic, for addressing uncertainty in MAS. However, most of the surveyed research works in (Calegari et al., 2021) did not translate their theoretical contributions into tangible MAS and CPS integration technologies. Based on the unaddressed aspects of prior studies, it is apparent that there is a significant need to explore methodologies that facilitate the deployment of agents in embedded systems, leveraging a range of APIs, development boards, and agent-oriented languages. This requires to lead us to design the proposed architecture, as described in (Karaduman et al., 2023a), and enhance the corresponding layers by incorporating and integrating fuzzy logic to ensure run-time feasibility and effectiveness. Therefore, our study focuses on addressing the identified gaps in the literature and the research needs identified by

SLR studies and related works. These gaps include the challenges of establishing smart and distributed CPS, deploying BDI agents on resource-constrained devices, and reducing run-time uncertainty. To address these needs, we propose enhancing BDI agents with fuzzy logic and providing a fuzzy-logic-based architecture for deploying these enriched software entities on embedded boards.

In this regard, our study provides a concrete case study that is agent-based, fuzzy-enhanced and has BDI-based reasoning. As also mentioned in (Esfahani & Malek, 2013), the uncertainties of the CPS mostly emerge from a lack of knowledge. This requires an additional logic-based paradigm that can provide reasoning on the run-time uncertainties, distributed deployment for instrumenting the complex, multi-stage processes and autonomy for self-sufficiency. Therefore, software agents can present more coverage and information sharing among the system components to reduce uncertainty. The reasoning capability of the BDI agents allows them to perform rational actions by selecting the adaptation plans when the system encounters unpredictable situations. When the BDI agents are enhanced with AI approaches such as fuzzy logic (Calegari et al., 2020, 2021), their combined abilities may guide the system to behave more intelligently. In pursuit of this objective, the first step involved deploying fuzzy-logic enhanced Jason BDI agents onto the embedded hardware, enabling them to operate the production line system in a distributed and collaborative manner. Most of the studies in the literature have focused on simulation and single CPS instances to evaluate their work. In contrast, we conducted our study on a complex, multi-stage and concrete case study considering the composite process control run by multiple agents. Furthermore, considering the distributed and tightly coupled approach, they have been less interested in providing a fuzzy-enhanced architecture.

As mentioned in (Boissier et al., 2020), incorporating new technologies, including network protocols and simulation technologies or logic-based approaches, into the BDI agents' internal architecture can be accomplished through two methods: using software artifacts or customising the BDI architecture. Our study reported in (Karaduman et al., 2022b) addresses the former method. This study explores the latter method, to assess the implications of integrating fuzzy logic with intelligent BDI agents for CPS. The fuzzy-BDI establishes rules within the agent software by utilizing membership functions and membership degrees to ensure the system's sustainability at runtime. The major concern is to reveal the efficacy of the fuzzy-logic BDI agents. In its forthcoming trajectory, the study desires to advance the BDI agent's architecture by specifically customizing Jason's BDI architecture to facilitate the construction of a versatile and domain-agnostic framework that can be generically applied across different domains. By enhancing the architecture of the BDI agent in this manner, the study intends to enable a more flexible and widespread adoption of the approach, thereby enabling its broad applicability and utilization.

One of the reasons that we benefited from the BDI agents is generally known that embedded programming benefits from high-level languages such as C and Basic, and the outshining abstraction agent-based programming provided a broad umbrella to merge different paradigms and technologies to achieve this study. However, it can be foreseen that there is still a need for higher abstractions and cognitive approaches for developing and programming CPS. Significantly, the cyber-physical system of systems (CPSoS) requires distributed programming where the software agents can be well-fit software entities(Leitao et al., 2016) to program, create and collaborate with multiple CPSs. The fuzzy logic-enhanced integrated architecture can provide integrity and facility towards devising multiple CPS to form a CPSoS. This considers that every CPS has an internal integrated structure with the horizontal extensibility of the software agent's cognitive capabilities. In addition, the middleware layer allows binding the agent actions within the plans to the API at the low-level hardware layer for actuation/sensing. Study (Karaduman et al., 2022b) also draws attention to MAS's social capabilities and distributed features in enabling the collaborative operation of CPS and encouraging interoperability with additional paradigms, notably the IoT.

The middleware layer can also provide fuzzy-logic-based analysis for network uncertainties or information processing/filtering (Leitão et al., 2021). Nevertheless, the concrete examples of deploying the enhanced BDI agents using multi-logic approaches for the short-term and learning abilities for the long-term objectives should be studied and validated on the concrete examples using the embedded technologies conforming to the proposed architecture (Karaduman et al., 2023a). These enhancements and architectural extensions should be diversified and merged regarding the CPS's cyber, physical and network facets.

Moreover, study (Leitão et al., 2022) discusses that the intelligent behaviour of the CPS is characterised by autonomy, cooperation, context awareness, cognitive computation, learning and adaptation. This also motivates and supports our work as the BDI agents provide reasoning mechanisms, cooperative messaging, context awareness, internal/external mental notes, proactivity and planning as cognitive computation. In our study, the fuzzy logic also enhances the adaptation, and our suggested architecture considers the extendability of the machine learning approaches for future works (Bosello & Ricci, 2020).

However, initially, the integrability and interoperability of the logic paradigms should be researched to create a mature infrastructure for future studies. For example, subjective approaches (Petrovska et al., 2021) can be incorporated with the BDI agents. As also pointed out in (Calegari et al., 2021), fuzzy and probabilistic methods can be inspected regarding Bayesian or Markovian approaches to enhance MAS and CPS integration capabilities. In the following section, the conclusion of the paper and planned future works are mentioned.

8. Conclusion & Future Work

This study presents a novel approach to enhancing the BDI agent model with fuzzy logic using an extended multi-logic architecture to address run-time uncertainty in CPS. By combining the reasoning mechanism of BDI agents with the capabilities of fuzzy logic, the proposed approach aims to provide better mitigation of unpredictable events in CPS.

The research questions addressed in this study focus on investigating the impact of fuzzy logic-based agents on CPS. To validate the proposed multi-logic approach, a heterogeneous and complex case study is selected, and a comparison is made with the traditional BDI approach. Various experiments are conducted, and statistical analyses are performed to evaluate the appropriateness and effectiveness of the fuzzy BDI approach.

The results of the experiments show that the fuzzy-based BDI agents are efficient in mitigating uncertainty and outperform the classical BDI agents in most cases. The modular experiments and end-to-end evaluations demonstrate the positive impact of the fuzzy-logic and BDI agent-based operation on complex CPS. It is worth noting that there may be some additional computation costs associated with the fuzzy BDI approach, but these costs can be tolerated considering the continuous evolution of embedded technologies.

This study is limited to ordinary type-1 fuzzy sets used to ignite the primary initialization of fuzzy-BDI application and integration. The membership degrees are used within the plans by multiplying the configuration parameters as an alternative to the defuzzification phase. In further studies, we plan to extend the approach by employing advanced fuzzy sets and learning approaches. Furthermore, our intention is to raise the level of abstraction benefiting from the model-driven development to engineer a modelling language specific to the domain that automates the generation of software artifacts. This approach aims to streamline the software development process by abstracting away low-level implementation details and allowing developers to focus on higher-level modelling concepts. By using this domain-specific modelling language, software artifacts can be automatically generated, reducing the manual effort required for implementation and ensuring consistency and coherence within the system.

Overall, this study contributes to the field of BDI agent-based CPS by introducing a multi-logic approach that combines fuzzy logic and BDI reasoning. The findings highlight the effectiveness of the fuzzy BDI approach in addressing uncertainty and provide directions for further research and improvement in this area.

References

Alaya, N., Dafflon, B., Moalla, N., & Ouzrout, Y. (2017). A cps-agent self-adaptive quality control platform for industry 4.0. In *Proceedings of the 7th International Conference on Information Society and Technology, Kopaonik, Serbia* (pp. 12–15).

Alcalá-Fdez, J., & Alonso, J. M. (2015). A survey of fuzzy systems software: Taxonomy, current research trends, and prospects. *IEEE Transactions on Fuzzy Systems*, *24*, 40–56.

Alves, B. R., Alves, G. V., Borges, A. P., & Leitão, P. (2019). Experimentation of negotiation protocols for consensus problems in smart parking systems. In *International Conference on Industrial Applications of Holonic and Multi-Agent Systems* (pp. 189–202). Springer.

Barbosa, J., Dias, J., Pereira, A., & Leitão, P. (2016). Engineering an adacor based solution into a small-scale production system. In *2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)* (pp. 28–33). IEEE.

Barbosa, J., Leitao, P., Ferreira, A., Queiroz, J., Geraldes, C. A., & Coelho, J. P. (2018). Implementation of a multi-agent system to support zdm strategies in multi-stage environments. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)* (pp. 822–827). IEEE.

Ben Mekki, A., Tounsi, J., & Ben Said, L. (2016). Fuzzy bdi agents for supply chain monitoring in an uncertain environment. *Supply Chain Forum: An International Journal*, *17*, 109–123.

Boissier, O., Bordini, R. H., Hubner, J., & Ricci, A. (2020). *Multi-agent oriented programming: programming multi-agent systems using JaCaMo*. MIT Press.

Bordini, R. H., Hübner, J. F., & Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason* volume 8. John Wiley & Sons.

Bosello, M., & Ricci, A. (2020). From programming agents to educating agents–a jason-based framework for integrating learning in the development of cognitive agents. In *Engineering Multi-Agent Systems: 7th International Workshop, EMAS 2019, Montreal, QC, Canada, May 13–14, 2019, Revised Selected Papers 7* (pp. 175–194). Springer.

Calegari, R., Ciatto, G., Denti, E., & Omicini, A. (2020). Logic-based technologies for intelligent systems: State of the art and perspectives. *Information*, *11*, 167.

Calegari, R., Ciatto, G., Mascardi, V., & Omicini, A. (2021). Logic-based technologies for multi-agent systems: A systematic literature review. *Autonomous Agents and Multi-Agent Systems*, *35*, 1–67.

Calinescu, R., Mirandola, R., Perez-Palacin, D., & Weyns, D. (2020). Understanding uncertainty in self-adaptive systems. In *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)* (pp. 242–251). IEEE.

Can, M. E., Ramkisoen, P., Karaduman, B., Demeyer, S., & Challenger, M. (2022). Enhancing autonomous guided robots using software agents and uwb technology. In *2022 11th Mediterranean Conference on Embedded Computing (MECO)* (pp. 1–6). IEEE.

Castillo, O., Amador-Angulo, L., Castro, J. R., & Garcia-Valdez, M. (2016). A comparative study of type-1 fuzzy logic systems, interval type-2 fuzzy logic systems and generalized type-2 fuzzy logic systems in control problems. *Information Sciences*, *354*, 257–274.

Challenger, M., & Vangheluwe, H. (2020). Towards employing abm and mas integrated with mbse for the lifecycle of scpsos. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings* (pp. 1–7).

Chen, M. (2015). A bdi agents programming language based fuzzy beliefs. In *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics* (pp. 334–336). IEEE volume 1.

Cheng, B. H., Sawyer, P., Bencomo, N., & Whittle, J. (2009). A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In *Model Driven Engineering Languages and Systems: 12th International Conference, MODELS 2009, Denver, CO, USA, October 4-9, 2009. Proceedings 12* (pp. 468–483). Springer.

Cordón, O., Herrera, F., & Peregrín, A. (1997). Applicability of the fuzzy operators in the design of fuzzy logic controllers. *Fuzzy sets and systems*, *86*, 15–41.

Cruz, A., dos Santos, A. V., Santiago, R. H., & Bedregal, B. (2021). A fuzzy semantic for bdi logic. *Fuzzy Information and Engineering*, *13*, 139–153.

Cuevas, F., Castillo, O., & Cortés-Antonio, P. (2022). Generalized type-2 fuzzy parameter adaptation in the marine predator algorithm for fuzzy controller parameterization in mobile robots. *Symmetry*, *14*, 859.

Dawson, S., Ramakrishnan, C. R., & Warren, D. S. (1996). Practical program analysis using general purpose logic programming systems—a case study. In *Proceedings of the ACM SIGPLAN 1996 conference on Programming language design and implementation* (pp. 117–126).

Esfahani, N., & Malek, S. (2013). Uncertainty in self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers* (pp. 214–238). Springer.

Farias, G. P., Dimuro, G. P., & da Rocha Costa, A. C. (2010). Bdi agents with fuzzy perception for simulating decision making in environments with imperfect information. In *MALLOW*.

Fichera, L., Messina, F., Pappalardo, G., & Santoro, C. (2017). A python framework for programming autonomous robots using a declarative approach. *Science of Computer Programming*, *139*, 36–55.

Flake, S., Geiger, C., Lehrenfeld, G., Mueller, W., & Paelke, V. (1999). Agent-based modeling for holonic manufacturing systems with fuzzy control. In *18th International Conference of the North American Fuzzy Information Processing Society-NAFIPS (Cat. No. 99TH8397)* (pp. 273–277). IEEE.

Fredericks, E. M., & Cheng, B. H. (2015). An empirical analysis of providing assurance for self-adaptive systems at different levels of abstraction in the face of uncertainty. In *2015 IEEE/ACM 8th International Workshop on Search-Based Software Testing* (pp. 8–14). IEEE.

Fredericks, E. M., DeVries, B., & Cheng, B. H. (2014). Autorelax: automatically relaxing a goal model to address uncertainty. *Empirical Software Engineering*, *19*, 1466–1501.

Garlan, D. (2010). Software engineering in an uncertain world. In *Proceedings of the FSE/SDP workshop on Future of software engineering research* (pp. 125–128).

Garrab, A., Bouallegue, A., & Bouallegue, R. (2017). An agent based fuzzy control for smart home energy management in smart grid environment. *International Journal of Renewable energy research*, *7*, 599–612.

Georgeff, M., & Ingrand, F. (1989). Decision-making in an embedded reasoning system. In *International Joint Conference on Artificial Intelligence*.

Gheibi, O., Weyns, D., & Quin, F. (2021). On the impact of applying machine learning in the decision-making of self-adaptive systems. In *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)* (pp. 104–110). IEEE.

Gomes, L., Almeida, C., & Vale, Z. (2020). Recommendation of workplaces in a coworking building: a cyber-physical approach supported by a context-aware multi-agent system. *Sensors*, *20*, 3597.

Greer, C., Burns, M., Wollman, D., Griffor, E. et al. (2019). Cyber-physical systems and internet of things.

Gregori, M. E., Cámara, J. P., & Bada, G. A. (2006). A jabber-based multi-agent system platform. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems* (pp. 1282–1284).

Karaduman, B., & Challenger, M. (2022). Smart cyber-physical system-of-systems using intelligent agents and mas. In *Engineering Multi-Agent Systems: 9th International Workshop, EMAS 2021, Virtual Event, May 3–4, 2021, Revised Selected Papers* (pp. 187–197). Springer.

Karaduman, B., David, I., & Challenger, M. (2021a). Modeling the engineering process of an agent-based production system: An exemplar study. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (pp. 296–305). IEEE.

Karaduman, B., Kardas, G., & Challenger, M. (2023a). Development of autonomous cyber-physical systems using intelligent agents and lego technology. In *Cyber-Physical Systems for Industrial Transformation* (pp. 193–211). CRC Press.

Karaduman, B., Tezel, B. T., & Challenger, M. (2021b). Towards applying fuzzy systems in intelligent agent-based cps: A case study. In *2021 6th International Conference on Computer Science and Engineering (UBMK)* (pp. 735–740). IEEE.

Karaduman, B., Tezel, B. T., & Challenger, M. (2022a). Deployment of software agents and application of fuzzy controller on the uwb localization based mobile robots. In *Intelligent and Fuzzy Systems: Digital Acceleration and The New Normal-Proceedings of the INFUS 2022 Conference, Volume 1* (pp. 98–105). Springer.

Karaduman, B., Tezel, B. T., & Challenger, M. (2022b). Enhancing bdi agents using fuzzy logic for cps and iot interoperability using the jaca platform. *Symmetry*, *14*, 1447.

Karaduman, B., Tezel, B. T., & Challenger, M. (2023b). Rational software agents with the bdi reasoning model for cyber–physical systems. *Engineering Applications of Artificial Intelligence*, *123*, 106478.

Karnouskos, S., Leitao, P., Ribeiro, L., & Colombo, A. W. (2020). Industrial agents as a key enabler for realizing industrial cyber-physical systems: multiagent systems entering industry 4.0. *IEEE Industrial Electronics Magazine*, *14*, 18–32.

Karnouskos, S., Ribeiro, L., Leitão, P., Lüder, A., & Vogel-Heuser, B. (2019). Key directions for industrial agent based cyber-physical production systems. In *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)* (pp. 17–22). IEEE.

Karnouskos, S., Sinha, R., Leitão, P., Ribeiro, L., & Strasser, T. I. (2018). Assessing the integration of software agents and industrial automation systems with iso/iec 25010. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)* (pp. 61–66). IEEE.

Kinay, A. O., & Tezel, B. T. (2022). Modification of the fuzzy analytic hierarchy process via different ranking methods. *International Journal of Intelligent Systems*, *37*, 336–364.

Lee, J., Bagheri, B., & Kao, H.-A. (2015). A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, *3*, 18–23.

Leitão, P., & Barbosa, J. (2016). Building a robotic cyber-physical production component. In *Service Orientation in Holonic and Multi-Agent Manufacturing* (pp. 295–305). Springer.

Leitão, P., Barbosa, J., Geraldes, C. A., & Coelho, J. P. (2018). Multi-agent system architecture for zero defect multi-stage manufacturing. In *Service Orientation in Holonic and Multi-Agent Manufacturing* (pp. 13–26). Springer.

Leitao, P., Karnouskos, S., Ribeiro, L., Lee, J., Strasser, T., & Colombo, A. W. (2016). Smart agents in industrial cyber–physical systems. *Proceedings of the IEEE*, *104*, 1086–1101.

Leitão, P., Queiroz, J., & Sakurada, L. (2022). Collective intelligence in self-organized industrial cyber-physical systems. *Electronics*, *11*, 3213.

Leitão, P., Ribeiro, L., Barata, J., & Vogel-Heuser, B. (2016). Summer school on intelligent agents in automation: Hands-on educational experience on deploying industrial agents. In *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society* (pp. 6602–6607). IEEE.

Leitão, P., Strasser, T. I., Karnouskos, S., Ribeiro, L., Barbosa, J., & Huang, V. (2021). Recommendation of best practices for industrial agent systems based on the ieee 2660.1 standard. In *2021 22nd IEEE International Conference on Industrial Technology (ICIT)* (pp. 1157–1162). IEEE volume 1.

Long, S. A., & Esterline, A. C. (2000). Fuzzy bdi architecture for social agents. In *Proceedings of the IEEE SoutheastCon 2000.'Preparing for The New Millennium'(Cat. No. 00CH37105)* (pp. 68–74). IEEE.

Ltaief, H., Karaduman, B., Boussaid, B., & Challenger, M. (2022). Agent based implementation of a robot arm and smart production line using jade framework. In *2022 11th Mediterranean Conference on Embedded Computing (MECO)* (pp. 1–12). IEEE.

McCarthy, J., & Shannon, C. (1958). *Automata studies.*

Mendel, J. M. (2007). Type-2 fuzzy sets and systems: an overview. *IEEE computational intelligence magazine*, *2*, 20–29.

Menegol, M. S., Hubner, J. F., & Becker, L. B. (2018a). Coordinated uav search and rescue application with jacamo. In *International Conference on Practical Applications of Agents and Multi-Agent Systems* (pp. 335–338). Springer.

Menegol, M. S., Hübner, J. F., & Becker, L. B. (2018b). Evaluation of multi-agent coordination on embedded systems. In *International Conference on Practical Applications of Agents and Multi-Agent Systems* (pp. 212–223). Springer.

Merdan, M., Vallee, M., Lepuschitz, W., & Zoitl, A. (2011). Monitoring and diagnostics of industrial systems using automation agents. *International journal of production research*, *49*, 1497–1509.

Morris, A., & Ulieru, M. (2011). Friend: A human-aware bdi agent architecture. In *2011 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 2413–2418). IEEE.

Muto, T. J., Bolivar, E. B., Serrano, J. E., & González, E. (2021). Multi-agent chans: Bdi farmer intentions and decision making. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Social Good. The PAAMS Collection: 19th International Conference, PAAMS 2021, Salamanca, Spain, October 6–8, 2021, Proceedings 19* (pp. 151–162). Springer.

Othmane, A. B., Tettamanzi, A. G., Villata, S., & Le Thanh, N. (2018). Cars–a spatio-temporal bdi recommender system: Time, space and uncertainty. In *ICAART 2018-10th International Conference on Agents and Artificial Intelligence* (pp. 1–10). SciTePress volume 1.

Pantoja, C. E., Stabile, M. F., Lazarin, N. M., & Sichman, J. S. (2016). Argo: An extended jason architecture that facilitates embedded robotic agents programming. In *International Workshop on Engineering Multi-Agent Systems* (pp. 136–155). Springer.

Peres, R., Rocha, A. D., Matos, J. P., & Barata, J. (2018). Go0dman data model-interoperability in multistage zero defect manufacturing. In *2018 ieee 16th international conference on industrial informatics (indin)* (pp. 815–821). IEEE.

Peres, R. S., Rocha, A. D., Coelho, A., & Barata Oliveira, J. (2017). A highly flexible, distributed data analysis framework for industry 4.0 manufacturing systems. In *Service Orientation in Holonic and Multi-Agent Manufacturing: Proceedings of SOHOMA 2016* (pp. 373–381). Springer.

Petrovska, A., Neuss, M., Gerostathopoulos, I., & Pretschner, A. (2021). Run-time reasoning from uncertain observations with subjective logic in multi-agent self-adaptive cyber-physical systems. In *16th Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS*.

Pico-Valencia, P., & Holgado-Terriza, J. A. (2018). Agentification of the internet of things: A systematic literature review. *International Journal of Distributed Sensor Networks*, *14*, 1550147718805945.

Queiroz, J., Leitão, P., & Oliveira, E. (2022). A fuzzy logic recommendation system to support the design of cloud-edge data analysis in cyber-physical systems. *IEEE Open Journal of the Industrial Electronics Society*, *3*, 174–187.

Ramirez, A. J., Fredericks, E. M., Jensen, A. C., & Cheng, B. H. (2012). Automatically relaxing a goal model to cope with uncertainty. In *Search Based Software Engineering: 4th International Symposium, SSBSE 2012, Riva del Garda, Italy, September 28-30, 2012. Proceedings 4* (pp. 198–212). Springer.

Rao, A. S. (1996). Agentspeak (l): Bdi agents speak out in a logical computable language. In *European workshop on modelling autonomous agents in a multi-agent world* (pp. 42–55). Springer.

Rao, A. S., & Georgeff, M. P. (1998). Decision procedures for bdi logics. *Journal of logic and computation*, *8*, 293–343.

Ribeiro, L., Karnouskos, S., Leitão, P., Barbosa, J., & Hochwallner, M. (2018). Performance assessment of the integration between industrial agents and low-level automation functions. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)* (pp. 121–126). IEEE.

Ricci, A., Piunti, M., & Viroli, M. (2011). Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, *23*, 158–192.

Rocha, A. D., Lima-Monteiro, P., Parreira-Rocha, M., & Barata, J. (2019). Artificial immune systems based multi-agent architecture to perform distributed diagnosis. *Journal of Intelligent Manufacturing*, *30*, 2025–2037.

Rodrigues, N., Pereira, A., & Leitão, P. (2013). Adaptive multi-agent system for a washing machine production line. In *Industrial Applications of Holonic and Multi-Agent Systems* (pp. 212–223). Springer.

Rodriguez-Ubeda, D., Flores, D.-L., Palafox, L., Castanon-Puga, M., Gaxiola-Pacheco, C., & Rosales, R. (2015). Extended reasoning cycle algorithm for bdi agents. *International Journal of Recent Research in Mathematics Computer Science and Information Technology*, *1*, 27–35.

Rosales, R., Castañón-Puga, M., Lara-Rosano, F., Evans, R. D., Osuna-Millan, N., & Flores-Ortiz, M. V. (2017). Modelling the interruption on hci using bdi agents with the fuzzy perceptions approach: An interactive museum case study in mexico. *Applied Sciences*, *7*, 832.

Rosin, F., Forget, P., Lamouri, S., & Pellerin, R. (2022). Enhancing the decision-making process through industry 4.0 technologies. *Sustainability*, *14*, 461.

Sakurada, L., Barbosa, J., Leitão, P., Alves, G., Borges, A. P., & Botelho, P. (2019). Development of agent-based cps for smart parking systems. In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society* (pp. 2964–2969). IEEE volume 1.

Schoofs, E., Kisaakye, J., Karaduman, B., & Challenger, M. (2021). Software agent-based multi-robot development: A case study. In *2021 10th Mediterranean Conference on Embedded Computing (MECO)* (pp. 1–8). IEEE.

Semwal, T., & Nair, S. B. (2016). Agpi: Agents on raspberry pi. *Electronics*, *5*, 72.

Shen, S., O'Hare, G. M., & Collier, R. (2004). Decision-making of bdi agents, a fuzzy approach. In *The Fourth International Conference on Computer and Information Technology, 2004. CIT'04.* (pp. 1022–1027). IEEE.

Shi, Y., & Xu, H. (2009). Research on the bdi-agent learning model based on dynamic fuzzy logic. In *2009 International Conference on Computational Intelligence and Software Engineering* (pp. 1–6). IEEE.

Tepjit, S., Horváth, I., & Rusák, Z. (2019). The state of framework development for implementing reasoning mechanisms in smart cyber-physical systems: A literature review. *Journal of computational design and engineering*, *6*, 527–541.

Tezel, B. T., Kardaş, G., & Uğur, A. (2016). Bulanık mantık tabanlı bdi etmenleri fuzzy logic based bdi agents. In *1st International Conference on Computer Science and Engineering (UBMK 16)*.

Tezel, B. T., & Mert, A. (2021). A cooperative system for metaheuristic algorithms. *Expert Systems with Applications*, *165*, 113976.

Vachtsevanou, D., William, J., Santos, M. M. D., Brito, M. D., Hubner, J. F., Mayer, S., & Gomez, A. (2023). Embedding autonomous agents into low-power wireless sensor networks, 21st international conference on practical applications of agents and multi-agent systems, 12-14 july, 2023. URL: `https://www.alexandria.unisg.ch/handle/20.500.14171/117517`. doi:`https://doi.org/20.500.14171/117517`.

Vieira, G., Barbosa, J., Leitão, P., & Sakurada, L. (2020). Low-cost industrial controller based on the raspberry pi platform. In *2020 IEEE International Conference on Industrial Technology (ICIT)* (pp. 292–297). IEEE.

Vu, T. M., Siebers, P.-O., & Wagner, C. (2013). Comparison of crisp systems and fuzzy systems in agent-based simulation: A case study of soccer penalties. In *2013 13th UK Workshop on Computational Intelligence (UKCI)* (pp. 54–61). IEEE.

Wei, C., & Hindriks, K. V. (2012). An agent-based cognitive robot architecture. In *International Workshop on Programming Multi-Agent Systems* (pp. 54–71). Springer.

Wesz, R. B. (2015). *Integrating robot control into the Agentspeak (L) programming language.* Master's thesis Pontifícia Universidade Católica do Rio Grande do Sul.

Weyns, D. (2020). *An Introduction to Self-adaptive Systems: A Contemporary Software Engineering Perspective.* John Wiley & Sons.

Whittle, J., Sawyer, P., Bencomo, N., Cheng, B. H., & Bruel, J.-M. (2009). Relax: Incorporating uncertainty into the specification of self-adaptive systems. In *2009 17th IEEE International Requirements Engineering Conference* (pp. 79–88). IEEE.

Xiaochao, W., Ying, C., & Longfei, C. (2019). A cgf behavior decision-making model based on fuzzy bdi framework. In *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)* (pp. 1487–1490). IEEE.

Yalcin, M. M., Karaduman, B., Kardas, G., & Challenger, M. (2021). An agent-based cyber-physical production system using lego technology. In *2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS)* (pp. 521–531). IEEE.

Zadeh, L. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on systems, Man, and Cybernetics*, (pp. 28–44).

Zadeh, L. A. (1972). A fuzzy-set-theoretic interpretation of linguistic hedges. *Journal of Cybernetics*, *2*, 4–34.

Zadeh, L. A. (1983). The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy sets and systems*, *11*, 199–227.

Zadeh, L. A. (1996). Fuzzy sets. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh* (pp. 394–432). World Scientific.