JOT JOURNAL OF OBJECT TECHNOLOGY

Journal of Object Technology | **RESEARCH ARTICLE**

Module-based Modelling and Assessment of Modular Robots w.r.t. Energy Efficiency

Antonios Naguib[*], **Olga Kouchnarenko**[*], **and Frédéric Lassabe**[*]
[*]Université Marie et Louis Pasteur, FEMTO-ST Institute, UMR 6174 CNRS, France

ABSTRACT Modular robots are cyber-physical systems composed of elements similar in nature that collaborate for a common global goal. This paper reports on a formal modelling of modular robots using Markov Decision Processes (MDPs) supported by the PRISM tools. The novelty of our modelling approach consists in incorporating empirical data and feedback from energy-dependent components collected thanks to an instrumentation of real modular robots called Blinky Blocks. While using PRISM modules for specifying individual robots or motifs composed of Blinky Blocks in their environment, the collected data are used for probabilistic transitions and non-deterministic choice, as well as for rewards to estimate energy availability that impacts fallback behaviours and actuation success. Afterwards, PRISM allows performing model-based analysis of safety, recovery, and energy-boundness under non-deterministic choice of the environment, and a model-based simulation on scenarios of interest. Comparing the obtained simulation results to those observed on real individual robots and their motifs provides useful insights for the developers, such as anomalous consumption patterns or inter-module variability. The designed trustworthy models can then be used for designing and validating adaptation policies for modular robots.

KEYWORDS Model-based engineering, Modular robots, Energy efficiency, Probabilistic model checking, Markov decision processes

1. Introduction

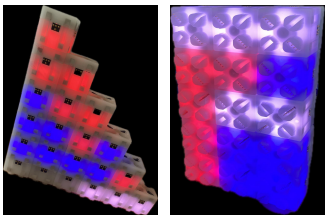
The fast development of distributed and modular systems has enabled them to be a central building block in the development of scalable, adaptive, fault-tolerant cyber-physical systems (CPS). Out of these, modifiable self adaptive systems are distinct with their ability to adapt and adjust in real-time due to a change in the internal or environmental context. This dynamic reactivity is crucial for applications as diverse as smart infrastructure, robots, autonomous mobility, or critical power distribution networks. Nevertheless, providing correctness and performance guarantees of these systems is non trivial because of their complexity, non-deterministic behaviours, and distribution in decision-making (de Lemos & et al. 2013).

Energy management is a major challenge for cyber-physical systems (Monostori 2014) as their behaviour is tightly coupled with physical resource constraints. Power limitations directly affect actuation reliability and can lead to failures, motivating the need for energy-aware modelling and analysis. Particularly, in large-scale deployments where power distribution, propagation, and optimization must be carefully controlled. Hence the CPS development requires improved design methodologies and tools to support specification, modelling and analysis of both cyber and physical aspects (Bliudze et al. 2019). Recent work (Päßler et al. 2025) describes a methodology and a tool support to address this question. However, there are methodological difficulties and lack of data for building CPS' models, which must be trustworthy to allow a meaningful model-based analysis. Moreover, the CPS' 5C architecture in (Bagheri et al. 2015) contains a data-to-information conversion layer with the purpose to help the developers design better and more robust systems and equipment by leveraging data from operational performance.

Blinky Blocks (BBs), one of several modular robotic systems, serve as a promising real-word platform to study these phenomena. Each BB can act independently or in coordination, offering a programmable interface, sensing, and actuating capabilities.

JOT reference format:
Antonios Naguib, Olga Kouchnarenko, and Frédéric Lassabe. *Module-based Modelling and Assessment of Modular Robots w.r.t. Energy Efficiency*. Journal of Object Technology. Vol. 25, No. 3, 2026.
http://dx.doi.org/10.5381/jot.2026.25.3.a15

Their modularity allows prototyping real-world adaptive systems and scalable experiments. The aim behind modelling BBs is to understand and formalize the behaviour of modular systems that could be scaled to larger smart infrastructure—such as distributed power plant modules that supply energy to variable consumption nodes like homes, gas stations, or city blocks. Despite having the same input configurations, these distributed systems may have similar architectures or utility functions, but the disparity in power demands between regions puts unequal strain on individual modules, resulting in a probabilistic distribution of success and failure states requiring hierarchical abstraction, communication and coordination in between for the global goal.



(a) Half-pyramid motif **(b)** Rectangle motif

Figure 1 BB motifs.

To formally reason about such stochastic behaviours and to ensure properties like safety, liveness and resilience, we utilize the PRISM model checker. We focus on modelling using Markov Decision Processes (MDPs), allowing us to capture both the non-deterministic task selection and probabilistic transitions observed in BBs. Unlike deterministic simulators, PRISM model checker allows us to rigorously verify some temporal properties under uncertainty, enabling deeper analysis of failure propagation, fallback strategies, and energy-aware adaptation (Kwiatkowska et al. 2011).

This paper presents a model-based workflow that bridges physical experimentation and formal analysis and simulation. As illustrated in Fig. 2, empirical measurements collected from instrumented Blinky Blocks are used to calibrate probabilistic transition models capturing energy-dependent failures and fallback behaviours. These calibrated parameters instantiate MDPs in the PRISM model checker, enabling formal verification, behavioural exploration by model animation, and strategy analysis under uncertainty. This paper organisation corresponds to the overview in Fig. 2, with the focus on system modeling with a parameter calibration based on empirical data collected, and on comparison of the observed artefacts vs. the results obtained on system model side (confrontations).

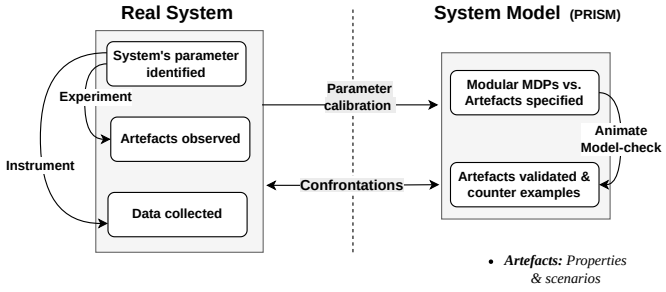


Figure 2 End-to-end workflow: From empirical instrumentation and parameter calibration to modular MDP modelling, formal verification, and design-time insights

2. Real System and Motivations

2.1. Blinky Block Overview

Systems composed of Blinky Blocks can be seen as CPSs structured in motifs (lines, stars, etc.), as illustrated in Fig. 1. A Blinky Block is a cube shaped robot of 4 cm-sized length (cf. Fig. 3), which executes a program, defining its actions depending on events it receives from its sensors. It may provide a visual rendering (colour, luminosity) by means of RGB LEDs, and may play sounds of a needed volume by means of a loudspeaker. The face magnets and connectors allow putting them together to form 2D and 3D structures.

More precisely, each Blinky Block includes an ARM microcontroller (MCU), a power supply (ALIM), multi-color LEDs, a speaker, an accelerometer (ACC), a microphone (MIC), and 6 USART[1] communication interfaces for inter-module coordination. These components achieve sensing, processing, and actuating tasks, making energy management a critical factor in ensuring system operations. The power supply requires a dedicated description: Indeed, for limiting cost, Blinky Blocks have no own power source. They are plugged to the electrical grid through one to many distributed AC/DC converters (230V AC to 5V DC), the number of which depends on the number of Blinky Blocks to supply. By distributed, we mean that Blinky Blocks have a parallel wired network, collocated with the USART interfaces, whose task is to connect to the grid through neighbors.

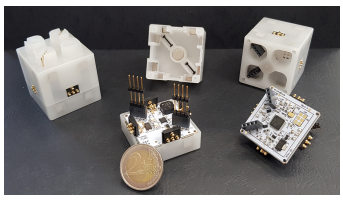


Figure 3 Blinky Block

Since the number of Blinky Blocks one power supply can serve varies from 8 to 20, based on past observations, it is complicated to build large systems. For instance, the largest MR composed of 1824 Blinky Blocks (Piranda et al. 2025) took about 9 work.days to complete, of whose more than half were for fixing incorrect connections and power transmissions.

2.2. BB-based Systems: Motivating Issues

Blinky Blocks systems are very useful for developing MRs algorithms and protocols. Indeed, while simulation (Thalamy et al. 2021) allows implementing distributed algorithms on millions of virtual MRs, and usual MRs physical implementations rarely exceed dozens, or a few hundreds, Blinky Blocks have been designed to be simple enough so that one may manufacture thousands of them. Thus, with 2000 Blinky Blocks it makes a good intermediate solution to test real-life MRs at a scale in-between what is usually achieved with physical robots and what can be simulated. Thus, it enables us to test distributed algorithms with real hardware and software, which will underline hardware-related issues that cannot be simulated precisely, while still having a size sufficient to make large scale Modular robots (MRs) systems observations.

Blinky Blocks also come with their flaws, since they have

[1] universal synchronous/asynchronous receiver/transmitter

been designed to be based on COTS[2] components and a simplified architecture to be cost effective. As such, they have limited capabilities, especially when it comes to self-awareness. In the scope of energy management and optimization, this translates to robots unable to measure the quality of their power supply (current and voltage), which leads to setups with power supplies count and location based on (more or less) educated guesses. Those already cost losses in our Blinky Blocks set, some completely burnt, others with one or two melted network interfaces. Thus, two main issues identified with the distributed power supplies are: *(i) Malfunctions:* This issue is benign since it just prevents the set of robots of performing its task. Usual symptom is a subset of Blinky Blocks not performing correctly (no light or no sound). Adding power supplies and checking connectors alignment helps solving this kind of issue. *(ii) Blinky Blocks destruction:* This issue, fortunately very uncommon, is critical since it may destroy hardware. It happens when too much current is served through a path, leading to a high Joule's effect that may melt components of the Blinky Block (up to the MCU). Usual symptoms are smoke and burning circuit smell. Solving the issue requires checking everything, add power supplies and replace the destroyed BBs. The larger and more complex the system is, the more likely the issue occurs.

2.3. Energy Metrics and Parameters

In this section, we outline the key parameters and assumptions for energy consumption we have identified.

The vital metrics which apparently influence the energy profiling towards our MRs applications are (A) Motif: The specific shape to structure the BBs; (B) Number of the feeding blocks: The number of blocks providing power supply; (C) Feeding block place: The position of the feeding block within the overall configuration; (D) Number of BBs: The total number of BBs involved in the experiment; (E) Commands of the components: The commands of individual components such as MIC, ACC, SPK, and LEDs; and (F) Command order towards the power supply: The sequence in which commands are issued to the power supply affecting the BB.

Current regime Concerning the assumptions, with the recent experiments, the power rails are acting as an ideal parallel bus or almost negligible. All BBs in the system are considered with the same bus voltage. Consequently, for every assigned command (metric E), the total current scales linearly with the number of blocks N (metric D).

Based on the description of the system and its power issues, one can easily understand why modelling and optimizing energy management in systems composed of Blinky Blocks is of interest.

3. Background

For modelling self-adaptive systems, we make use of *Markov Decision Processes* (MDPs), a standard formalism for combining probabilistic behaviour with non-deterministic choices (Baier & Katoen 2008).

[2] commercial-off-the-shelf

3.1. MDPs

We use MDPs labelled with atomic propositions and a reward structure.

Definition 1 (Labelled MDP with rewards)**.** *A labelled MDP with rewards is a tuple*

$$\mathcal{M} = (S, A, P, R, \mathcal{AP}, L),$$

where S is a finite non-empty set of states, A is a finite set of actions, where $A(s) \subseteq A$ denotes the non-empty set of actions available in state s; $P : S \times A \times S \to [0,1]$ is the probabilistic transition function such that, for all $s \in S$ and $a \in A(s)$(the enabled actions in s), $\sum_{s' \in S} P(s,a,s') = 1$; $R = (r_S, r_A)$ is a reward structure with $r_S : S \to \mathbb{R}_{\geq 0}$ and $r_A : S \times A \to \mathbb{R}_{\geq 0}$, $\mathcal{AP}$ is a finite set of atomic propositions, and $L : S \to 2^{\mathcal{AP}}$ is an interpretation function to label states with atomic propositions.

In the rest of the paper, we often take the state names as atomic propositions, with $L(s) = \{s\}$. In particular, we denote `finish` and `failure` $\in S$ the absorbing success and failure states, respectively. The paths in $\mathcal{M}$ describe the potential runs resulting from both the non-deterministic and probabilistic choices resolution. Paths in an MDP are defined as alternating sequences of states and actions: An infinite path π in an MDP $\mathcal{M}$ is an infinite sequence $s_0\alpha_1 s_1 \alpha_2 s_2 \alpha_3 \ldots \in (S \times A)^\omega$, such that $P(s_i, \alpha_{i+1}, s_{i+1}) > 0$ for all $i \geq 0$. Let $Paths(s)$ denote the set of infinite paths that start in state s, and $Paths(\mathcal{M}) = \bigcup_{s \in S} Paths(s)$ denote the set of infinite paths of $\mathcal{M}$.

For any state $s \in S$, scheduler (or policy) σ induces a probability measure $\Pr_s^\sigma$ over $Paths(s)$ to resolve nondeterminism. A scheduler for $\mathcal{M}$ is a function $\sigma : S^+ \to A$ such that $\sigma(s_0 s_1 \ldots s_n) \in A(s_n)$ for all $s_0 s_1 \ldots s_n \in S^+$. The path π is called a σ-path if $\alpha_i = \sigma(s_0 \ldots s_{i-1})$ for all $i > 0$.

We consider memoryless schedulers, either deterministic (MD), or randomized (MR) ones, which depend only on the current state:

$$\Pi^{\mathrm{MD}} = \{\, \sigma : S \to A \mid \sigma(s) \in A(s) \,\},$$
$$\Pi^{\mathrm{MR}} = \{\, \sigma : S \to \mathcal{D}(A) \mid \mathrm{supp}(\sigma(s)) \subseteq A(s) \,\}.$$

Here, $\mathcal{D}(A)$ denotes the set of discrete probability distributions over A, and supp denotes the support condition ensures that a randomized scheduler assigns non-zero probability only to actions that are enabled in the current state, preserving the operational semantics of the MDP. For the objectives used in this paper, memoryless schedulers are sufficient to attain optimal probabilities (Baier & Katoen 2008).

3.2. Probabilistic Computational Tree Logic (PCTL)

To formally specify behavioural and quantitative properties over $\mathcal{M}$, we adopt *Probabilistic Computational Tree Logic (PCTL)*, which is widely used for stochastic systems model-checking (Hansson & Jonsson 1994; Baier & Katoen 2008).

Definition 2 (PCTL State and Path Formulas Syntax)**.** *State formulas φ are defined by:*

$$\varphi ::= \mathit{true} \mid a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathsf{P}_{\bowtie p}[\chi],$$

where a is an atomic proposition, $p \in [0,1]$*, and* $\bowtie \in \{\leq, \geq\}$*. Path formulas* χ *are given by:*

$$\chi ::= \mathsf{X}\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2.$$

We refer to (Baier & Katoen 2008) for the PCTL semantics. It is defined over states and paths of MDP under non-deterministic choice schedulers. The propositional formulas semantics is standard over states. In addition, a state s satisfies $\mathsf{P}_{\bowtie p}[\chi]$ if and only if $\inf_\sigma \Pr_s^\sigma(\{\pi \in Paths(s) \mid \pi \models \chi\}) \bowtie p$ or $\sup_\sigma \Pr_s^\sigma(\{\pi \in Paths(s) \mid \pi \models \chi\}) \bowtie p$, depending on whether the minimum or maximum probability is considered.

Path formula satisfaction $\pi \models \chi$ is defined inductively over the structure of χ in the standard way for CTL with the semantics of the X and U operators unchanged. Here, inf and sup range over all schedulers σ. Ranging over all schedulers and considering minimal or maximal probabilities corresponds to a worst-case analysis. This is due to the fact that the full class of schedulers covers all possible resolutions of the nondeterminism that is present.

Examples include safety requirements such as bounding the probability of reaching a failure state, where F stands for the "eventually" operator[3] $\mathsf{P}_{\leq p}[\,\mathsf{F}\, failed\,]$, and liveness properties ensuring successful recovery: $\mathsf{P}_{\geq p}[\,\mathsf{F}\, adapted\,]$.

3.3. Self-Adaptive System Architecture

A self-adaptive system (SAS) reference model (IBM Corporation 2006) commonly structures the system using a two-layered approach which decomposes the system into a managed and a managing subsystems interacting with an environment. Overall, this architecture supports a layered and feedback-driven adaptation process: telemetry flows upward the managing subsystem, decisions propagate downward towards managed subsystem, and environmental influences modulate sensing and energy dynamics. This design aligns with established principles of self-adaptive systems (de Lemos & et al. 2013; Weyns 2019), and extends them to resource-constrained modular robotics as illustrated in a recent work (Päßler et al. 2025).

Following (Päßler et al. 2025), the behaviour of a *family* of systems that differ in their features, e.g., in the managed subsystem, can be specified. Then a feature controller can activate and deactivate the features at run-time, and thus change the behaviour of the system. It can be used, e.g., by the managing subsystem of an SAS to change the configuration of the managed subsystem. Furthermore, the environment can be specified as a separate module that interacts with the managed and managing subsystems. The resulting architecture combines stochastic behaviour arising from hardware and environment interactions with non-deterministic control decisions at the managing layer.

We follow this methodology for developing an MDP model of a single BB, and a multi-module MDP for BB-based systems presented in the paper. Analyzing such systems requires a formalism that supports both probabilistic and adversarial reasoning across interacting components.

[3] $F\varphi = true\,\mathsf{U}\,\varphi$

3.4. Tool Support

PRISM (Kwiatkowska et al. 2011) is the core modelling and analysis framework to support complex systems development while using MDPs. It enables the integration of real-world uncertainty, empirical behaviour, and adaptive fallback logic in unified and analyzable models. However, the state space exploration is limited due to scalability limitations in PRISM's explicit-state engine; hence the modelling choices at different abstraction levels. For our BBs case study, the system is composed of Blinky Blocks (BBs), each modelled as an agent with local states and transitions governed by empirical failure probabilities and energy-awareness policies, as described in the following sections.

4. Single Blinky Block Model

Following the method described in Sect. 3.3, the modular robots architecture is separated into managed and managing subsystems. For BB-systems executing sensing, actuation, and communication, the managed subsystem comprises the physical entities—LEDs, speaker, microphone, and accelerometer—as well as the firmware-level logic that selects colour, intensity, pitch, and simultaneous blink–buzz modes. Communication among BBs is performed through USART links, and all BBs rely on a shared constrained power supply, which significantly influences actuation success and sensing reliability. The design favors modularity, scalability, and fidelity to real-world behaviour, representing internal hardware entities (e.g., RGB LEDs, speakers, accelerometers, microphones) by distinct, interpretable states within each module.

The BB managed subsystem is displayed in Fig. 4 illustrating a modular state-/transition-based representation through a combination of non-deterministic and probabilistic transitions. At this stage the design intuitively prioritizes energy-efficiency and self-adaptive behaviour by assigning higher success probabilities to lower-energy actuation states and by redirecting likely failure transitions toward energy-saving fallback modes.

Initialization (s_0)[4]: The system begins in an idle state where all sensors and actuators are inactive. This state corresponds to the baseline for triggering operational transitions. The deterministic transition to INIT state is used to wake the system devices up. **Color selection (s_1–s_3)**: Upon activation, each BB can non-deterministically transition into one of the colour states. W.l.o.g., we represent three colour states—red, blue, or white—which mainly is user-defined, mapped to empirical energy profiles as lightning colours can be more or less energy consuming. Setting up subsequent actuation behaviours depends on these states.

Sensing states (s_4–s_6) capture microphone and accelerometer inputs. These inputs may be given together or separately. The modelling includes probabilistic transitions to success or failure depending on environment triggers and sensing accuracy.

Pitch and LED intensity states (s_7–s_9, s_{14}–s_{16}) model speaker and blinking outputs at varying intensity levels, namely low, medium, and high. High levels consume more power and are

[4] The states correspond to the SingleBB model available in our git repository.

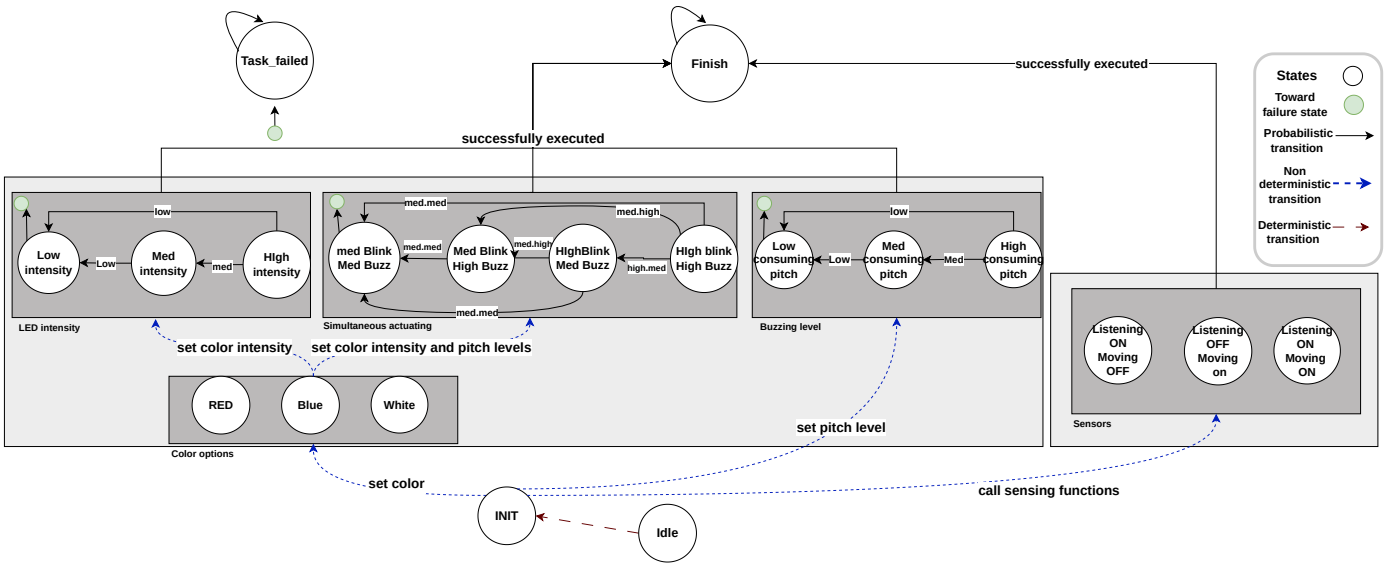


Figure 4 Managed subsystem state diagram (omitting state numbers)

more failure-prone, whereas lower levels improve reliability and energy efficiency.

Simultaneous actuation (s_{10}–s_{13}) allow concurrent LED and speaker activation. They support transitions down to less consuming states depending on runtime constraints, environmental conditions, or hardware feedback. Transitions that reduce energy demand, reflect adaptation logic chosen.

To track system reliability, two more operational outcome states are added:

Task_failed (s_{17}) to represent unsuccessful operations. However, in alignment with adaptive strategies aiming to avoid a blackout, failure state is reachable only from low states, and such cases increment a *task_counter* corresponding to E parameter from Sect.2.3. This modelling choice is represented by fallback transitions to less consuming alternatives (e.g., from high-pitch to medium or low pitch) based on predefined failure distributions.

Success (s_{18}) represents completed tasks and the *task_counter* is incremented as well. Accessible by probabilistic transitions from actuators and sensing states, this state represents various successful behaviours.

The BB managing subsystem implements the BB adaptation logic, which corresponds to activating and deactivating the features of the managed subsystem, following the method in (Päßler et al. 2025). Figure 5 illustrates an excerpt of a managing subsystem dealing with LEDs intensities. The grey area includes the transitions that can be taken during the task execution, and the white area contains the transitions once the task executed successfully. Each transition contains a guard. Guards are written in black as in Fig. 5. For example, in the high intensity state, if the failure probability (FP) is high, the managing subsystem selects next state of low intensity. Note that the transitions in the grey area implicitly carry the guard $s! = finish$.

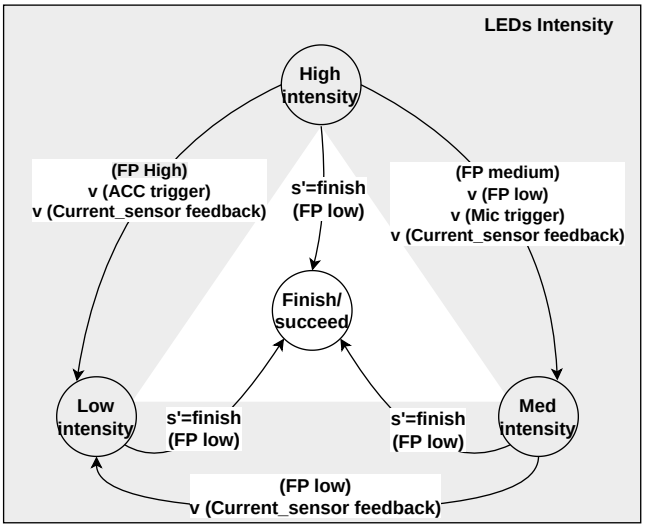


Figure 5 Managing subsystem for LEDs intensities

When assigning tasks for the BB, i.e., in the grey area for the managing schema, the managing subsystem activates and deactivates the features with low, med and high intensity according to the failure probabilities. It is clear that this is empirical data driven modelling. Activating a feature enables a transition of the corresponding action in the managed subsystem schema. Adaptation is triggered by sensing functions such as the accelerometer or microphone, as well as by current sensor feedback indicating the need to shift the module toward lower-intensity state.

5. Multi-Module Modelling

Building on the single-BB managed subsystem (Fig. 4), the collective behaviour of multiple BBs can be modelled by an MDP with interactions among BBs modules. Unfortunately, this approach quickly faces the state explosion with only a few

BBs modelled (depending on Prism engines used). This is why the modelling is performed at a more abstracted level from the BB point of view. Instead, the focus is now on BBs interactions.

In this section we describe the model for BB-systems composed of two interacting clusters.[5] We consider a cluster to be a group of adjacent BBs that execute exactly the same assigned commands (parameter E in Sect. 2.3). A task example is given in Listing 1.

```
#define TASK_COUNT              3
#define CLUSTER_COUNT           2
//omitted code
/* Task commands per cluster */
static const CommandType
    TASK_COMMAND[TASK_COUNT][CLUSTER_COUNT] = {
    /* Task 0 commands for each cluster */
    { CMD_LED_BLUE_HIGH,  CMD_LED_RED_HIGH },
    /* Task 1 commands for each cluster */
    { CMD_LED_WHITE_MED,  CMD_LED_BLUE_HIGH },
    /* Task 2 commands for each cluster */
    { CMD_LED_WHITE_LOW,  CMD_LED_BLUE_HIGH }};
//omitted code
```

Listing 1 Task-to-cluster command mapping in the BB firmware

Each cluster is treated as a macro-agent representing a group of N_1 and N_2 BBs, respectively. Internally, at the cluster level, the detailed per-block behaviour is merged to be observed at the task level with different hardware actuation modes. In particular, let's mention MDP states such as `idle`, `start_task`, `finish`, `task_failed`, colour selection (`choose_Red`, `choose_Blue`, `choose_white`), pitch levels (`high_pitch`, `med_pitch`, `low_pitch`), LED intensities (`high_Led`, `med_Led`, `low_Led`), combined blink–buzz modes, and sensing states for MIC/ACC (`listen_on_moving_on`, `listen_on_moving_off`, `listen_off_moving_on`). This way, the representation allows capturing the salient choices and outcomes of a set of BBs executing the same program commands.

Figure 6 shows the global MDP architecture. The model follows a Globally Asynchronous, Locally Synchronous (GALS) style (Lee & Messerschmitt 2000). In each cluster (c_1 and c_2), BBs are assumed to evolve synchronously following the direct state-transition models product, while the two clusters progress in a globally asynchronous manner. A shared synchronisation label `[step]` is used when both clusters evolve in lockstep, and an alternative label `[stag1]` combined with a `turn` variable is used to realise staggered execution. This mechanism allows us to model, and then to compare fully synchronous task executions vs. asynchronous operation with task interleaving, under the same abstract task logic.

Cluster activation and scheduling are governed by a separate *Task_controller* module. It maintains a task counter, activates or deactivates each cluster via Boolean flags (`active1`, `active2`), and, when staggered execution is enabled, controls which cluster is allowed to move at each step. An *Operator* module exposes a binary flag `staggered` that switches the system between purely synchronous execution (`staggered = false`) and staggered execution (`staggered = true`), without altering the internal semantics of the cluster modules. This separation between cluster logic and scheduling policy is essential for exploring different coordination patterns over the same underlying behaviour.

Two additional modules capture the cyber-physical context of the clusters. The *Hardware* module provides a coarse-grained abstraction of current-sensor feedback with a discrete state `hw_state` $\in \{0,1,2\}$ denoting nominal, medium-load, and high-load conditions. Hardware feedback is given highest priority in transition resolution: When `hw_state` indicates overload, the model deterministically enforces fallback from high- to medium- or low-consumption states (e.g., from `high_Led` to `med_Led`/`low_Led`), thereby encoding energy-safety constraints. The *Environment* module models MIC/ACC sensing modes for each cluster (`env_state1`, `env_state2`) and Boolean flags `sound` and `move` describing external activity. These variables influence probabilistic transitions from sensing and actuation states, capturing the impact of ambient noise and motion on success and failure.

A *pattern* module models the physical motif in which each cluster operates. The possible 2D motifs include lines, stars, rectangles, hollowed rectangles, etc. In the case study, Flags `Rec_motif1` and `Rec_motif2` distinguish between a line motif (baseline) and a rectangle motif. For the rest of the paper, we introduce coefficients described in Table 1. In this section, only their intuition is given from the model designer point of view. A coefficient θ scales success probabilities when clusters operate in a rectangle, reflecting the improved power distribution and load sharing observed empirically as compared to a single-supply line configuration. In addition, a coefficient γ adjusts probabilities in staggered mode, and a parameter α_2 is reserved to capture the higher failure risk of a distant cluster due to increased resistance and voltage drop.

The transition probabilities in the model are expressed symbolically using constants such as $q\_w\_high\_c1$, $q\_b\_high\_c1$ or $q\_high\_pitch\_c1$, which denote base success probabilities under nominal conditions. These are combined with coefficients (β_1, β_2, θ, γ) and guards over motif, execution mode, hardware and environment state to capture conditional behaviour. For instance, moving from a high-intensity, line-motif state to `finish` under nominal hardware and quiet environment uses the base probability; operating in a rectangle motif multiplies this probability by θ; and fallback from high to medium or low intensity scales it by β_1 or β_2. Joint success of combined blink–buzz states is approximated as the product of LED and buzzer success probabilities, assuming conditional independence. Actual values for these parameters are obtained from the empirical analysis described in Sect. 6, which calibrates the model against measured success rates and energy consumption.

Overall, the two-cluster MDP provides a compact but expressive representation of the system's collective behaviour. It preserves the essential structure of individual BBs, incorporates environment and hardware feedback, supports different motifs and scheduling modes, and remains parameterised, so that different experimental scenarios can be played and analysed thanks to a single, reusable model.

[5] Model available in our git repository.

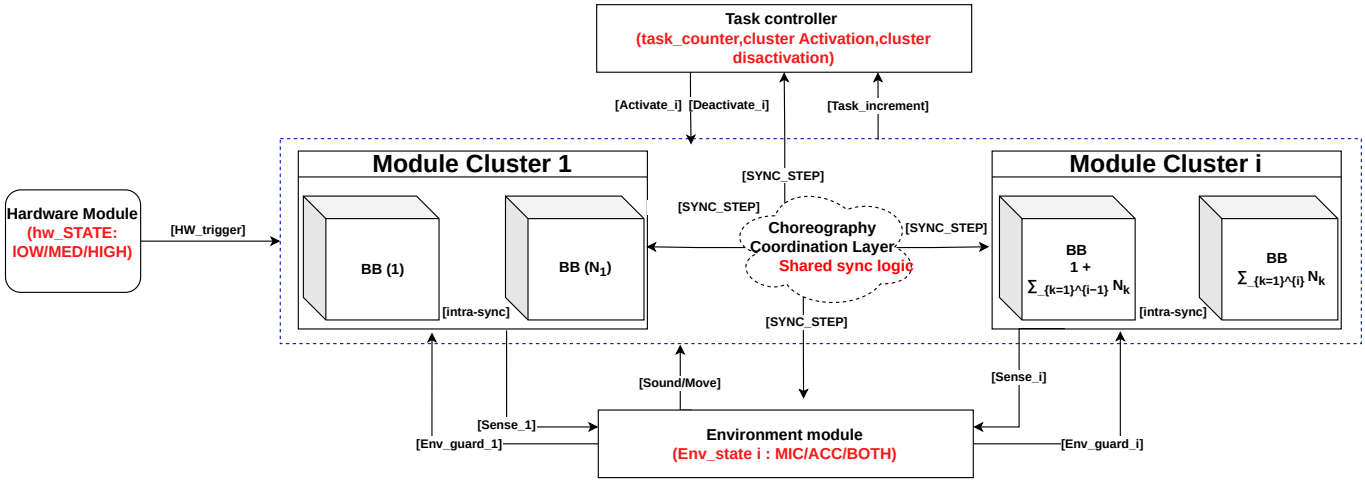


Figure 6 Two-cluster PRISM model schema with task controller, hardware and environment modules.

6. Empirical Data for MDPs

6.1. Empirical Transitions and Scheduler

Each cluster is modelled as an MDP with finite state space S. A local state $l \in S$ is written as $l = (s, h, e, c)$, where s denotes the current *control state* of the cluster (i.e. the active actuation or sensing mode), $h \in \{0, 1, 2\}$ is the hardware feedback (nominal/med/high), $e = (\text{env}, \text{sound}, \text{move})$ is the environment feedback with $\text{env} \in \{\text{idle}, \text{MIC}, \text{ACC}, \text{BOTH}\}$ and $\text{sound}, \text{move} \in \{0, 1\}$, and $c \in \{\text{Red}, \text{Blue}, \text{White}\}$ is the selected colour.

Scheduling with Guarded Transitions The behaviour of the managed subsystem is defined by a set of guarded probabilitic/deterministic transitions, following the standard MDP semantics. Nondeterminism between enabled transitions is resolved by a *memoryless scheduler*, as defined in Sect. 3.1. Once a transition is selected, its associated probability distribution fully determines the successor state. The transition set is thus partitioned into three families of guarded transitions, corresponding to hardware safety (T_{hw}), environment-driven adaptation (T_{env}), and baseline behaviour (T_{base}). We expect the scheduler obeying the following constraints: Hardware feedback deterministically overrides all other decisions, environment-driven adaptation is applied when hardware conditions are nominal, and baseline behaviour is expected otherwise.

(1) Hardware–safety T_{hw}. Hardware feedback enforces deterministic downgrades. This is modelled using Dirac distributions δ:

$$T_{\text{hw}}(s,h) = \begin{cases} \delta_{\text{LowPitch}} & \text{if } s = \text{HighPitch},\ h = 2, \\ \delta_{\text{MedPitch}} & \text{if } s = \text{HighPitch},\ h = 1, \\ \delta_{\text{LowLED}} & \text{if } s = \text{HighLED},\ h = 2, \\ \delta_{\text{MedLED}} & \text{if } s = \text{HighLED},\ h = 1, \\ \delta_{\text{MedBlinkMedBuzz}} & \text{if } s = \text{HighBlinkHighBuzz}, \\ & h = 2. \end{cases}$$

If a hardware guard holds, lower-priority transition guards are ignored.

(2) Environment T_{env}. When $h = 0$ and environmental stress is detected, transitions are biased probabilistically.

For example, the pitch family transitions contain

$$T_{\text{env}}(s,h,e,c) = \begin{cases} 0.25\,\delta_{\text{MedPitch}} + 0.75\,\delta_{\text{LowPitch}} & s{=}\text{HighPitch}, \\ & \text{env}_i{=}\text{ACC},\ \text{move}{=}1' \\ 0.75\,\delta_{\text{MedPitch}} + 0.25\,\delta_{\text{LowPitch}} & s{=}\text{HighPitch}, \\ & \text{env}_i{=}\text{MIC},\ \text{sound}{=}1. \end{cases}$$

(3) Empirical baseline T_{base}. When hardware and environment are nominal ($\text{env}_i = \text{idle}$, $\text{sound} = \text{move} = 0$), we expect empirical distributions derived from measurements.

LED family (colour–dependent, no stress):

$$T_{\text{base}}(\text{HighLED}, c) = \begin{cases} 0.70\,\delta_{\text{FIN}} + 0.15\,\delta_{\text{MedLED}} + 0.15\,\delta_{\text{LowLED}} \\ c = \text{White}, \\ 0.80\,\delta_{\text{FIN}} + 0.10\,\delta_{\text{MedLED}} + 0.10\,\delta_{\text{LowLED}} \\ c = \text{Blue}, \\ 0.90\,\delta_{\text{FIN}} + 0.05\,\delta_{\text{MedLED}} + 0.05\,\delta_{\text{LowLED}} \\ c = \text{Red}. \end{cases}$$

In summary, the overall guarded transition order is

$$T(s,h,e,c) = \begin{cases} T_{\text{hw}}(s,h) & \text{if a hardware guard holds,} \\ T_{\text{env}}(s,h,e,c) & \text{else if an environment guard holds,} \\ T_{\text{base}}(s,c) & \text{otherwise.} \end{cases} \tag{1}$$

Listing 2 displays one command line. Each actuation command is annotated with a calibrated conditional success probability $\Pr(\text{succ} \mid c, \iota, k, m, o)$, whose value depends on colour c, intensity ι, cluster index k (position or distance w.r.t. the feeding block), motif m, and the current operation mode o (either synchronous or staggered one).

```
[step] s1=high_Led  &color1=3& hw_state=0 &
  env_state1=0 & (sound=0 & move=0) &
  Rec_motif1=false & staggered=false->
  q_w_high_c1 : (s1'=finish) + (1-q_w_high_c1)
  /2 : (s1'=med_Led) + (1-q_w_high_c1)/2 : (s1
  '=low_Led); // default (empirical)

//... omitted code ...
```

Listing 2 An excerpt of the BB model: Line motif, synchronous cluster execution step with a conditional probability. Conditions are set on the left-hand side to guard this action.

This way, only relevant probability is computed and associated with the action labeling transition from the configuration considered, that satisfies the guard. The resulting transitions directly encode empirical behaviour, with success and fallback branches proportionally adjusted to the calibrated conditional probabilities.

Calibrating Probabilistic Failure Transitions from Empirical Data Empirical measurements over the real BB-systems have been performed thanks to the BB instrumentation and data visualisation support developed.[6] The collected data are used to assign values to the probabilities used onto the MDP transitions, and to probabilistic constants and coefficients described in Sect. 5. The goal is to identify a compact, reusable set of model's parameters that summarise systems' behaviour while varying colours, intensities, spatial motifs, execution modes, and other parameters influencing energy consumption. The resulting constants or ranges are used for assigning the baseline model part described in Sect. 6.1.

For example, given a line motif composed of 20 BBs, our experimental campaign provides colour-specific failure frequencies for 3 clusters (white, blue, and red) depending on the increasing distance from the power supply. More specifically, failures occur almost exclusively in the far clusters, and the failure probability ordering $p_{\text{White}} > p_{\text{Blue}} > p_{\text{Red}}$ is consistent across all tests.

A literal fit would set the proximal cluster's failure probability to zero, but this would not generalise to larger or denser formations. We, therefore, adopt a conservative scaling:

$$p_{\text{near}}(c) = \alpha_{clusterdistance}\, p_{\text{far}}(c)$$

$$\alpha_{clusterdistance} \approx \frac{P_{\text{farSuccess}}}{p_{\text{nearSuccess}}} \approx 0.21_{(white)}$$

For the values extracted experimentally, one has:

$$\begin{aligned} p_{\text{far}}(\text{Red}) &\approx 0.23, & p_{\text{near}}(\text{Red}) &\approx 0.10, \\ p_{\text{far}}(\text{Blue}) &\approx 0.45, & p_{\text{near}}(\text{Blue}) &\approx 0.20, \\ p_{\text{far}}(\text{White}) &\approx 0.85, & p_{\text{near}}(\text{White}) &\approx 0.30, \end{aligned}$$

preserving the empirical fragility ordering and avoiding unrealistic perfect-reliability zones.

These values parameterise the MDP transitions: Each colour-cluster activation includes a probabilistic branch to a failure state using $p_{\text{near}}(c)$ (cluster 1) or $p_{\text{far}}(c)$ (cluster 3) above. The abstraction proposed here is conservative and ensures the developed controller complies with the real system.

[6] This part of work is not detailed in the present paper.

6.2. Overview of the Parameter Set

Let us now consider real MR executions that are specified by the BB-system designer/developer, also called scenarios. Cluster 1 is treated as the reference cluster, meaning that all other clusters' probabilities are empirically computed with respect to this cluster. In general, speaking empirically, cluster 1 has the highest probabilities to execute successfully. Table 1 summarises the constants used by the MDP model under scheduler. For each constant, we compute scenario-specific candidate values[7] and then obtain: (i) a single global constant used in the MDP, and (ii) an empirical range given by the minimum and maximum candidate values across all calibration scenarios. When insufficient or inconsistent data is available, the corresponding constant is conservatively set to 1 (neutral).

Cluster 2 colour-specific probabilities are not defined independently. Instead, we have

$$q_{c,high,c2} = \alpha_2 \cdot q_{c,high,c1},$$

and the same ratio applies for medium and low intensities. This reduces overfitting and preserves colour ordering. The pessimistic dominance of the white LED motivates using white-based degradation to bound all cluster 2 colours.

Intensity–dependent success is derived from experiments on cluster 1:

$$q_{c,med} = \min(1, \beta_1 \cdot q_{c,high,c1}),\ q_{c,low} = \min(1, \beta_2 \cdot q_{c,high,c1}).$$

6.3. Calibration Protocol

The calibration procedure operates over each experimental scenario as follows:

- For a task involving a colour–intensity pair (c, i), count the number n of issued commands and of failures f. Compute the empirical failure rate: $\hat{p}_{\text{fail}} = \frac{f}{n}, \quad \hat{p}_{\text{succ}} = 1 - \hat{p}_{\text{fail}}$.
- Apply Hoeffding' correction (Hoeffding 1963) to bound sampling variability: $p^*_{\text{fail}} = \min\big(1, \hat{p}_{\text{fail}} + \sqrt{\frac{\ln(2/\delta)}{2n}}\big), \quad \delta = 0.05$.
- Extract base cluster 1 high–intensity constants $(q_{w\_high\_c1}, q_{b\_high\_c1}, q_{r\_high\_c1})$. Red, blue, and white preserve their empirical ordering.
- Compute α_2 as the min ratio between cluster 2 and cluster 1 success. White LED provides the upper (pessimistic) bound, and red the lower bound.
- Derive global intensity coefficients β_1 and β_2 by comparing medium– and low–intensity results against high–intensity in any representative colour. These are applied uniformly across colours.
- Estimate θ using pairs of line vs. rectangular motifs under identical tasks. Set $\theta = 1$ when no motif correction is desired.
- Estimate γ from sequential vs. parallel tasks executions. Set $\gamma = 1$ when the scheduling difference should be neutral.
- Clamp probabilities to $[0, 1]$ to avoid invalid transition weights. *(if needed)*

This procedure can easily be extended for several clusters and adapted to other types of modular robots.

[7] in Table 1; $p^{*(sc)}$: calibrated or inferred parameter (not a raw empirical output) for scenario *sc*

Table 1 Probabilistic constants and their empirical interpretation.

Constant	Meaning	Empirical range (over scenarios)
$q_{w\_high\_c1}$	Base success probability for white LED, high intensity, cluster 1. Obtained as corrected success rates $p^{*(sc)}_{\text{succ}}(W, \text{high}, c1)$ over all scenarios sc.	$\left[\min_{sc} p^{*(sc)}_{\text{succ}}(W, \text{high}, c1), \max_{sc} p^{*(sc)}_{\text{succ}}(W, \text{high}, c1)\right]$
$q_{b\_high\_c1}$	Analogous base success for blue LED, high intensity, cluster 1.	$\left[\min_{sc} p^{*(sc)}_{\text{succ}}(B, \text{high}, c1), \max_{sc} p^{*(sc)}_{\text{succ}}(B, \text{high}, c1)\right]$
$q_{r\_high\_c1}$	Analogous base success for red LED, high intensity, cluster 1.	$\left[\min_{sc} p^{*(sc)}_{\text{succ}}(R, \text{high}, c1), \max_{sc} p^{*(sc)}_{\text{succ}}(R, \text{high}, c1)\right]$
$q_{high\_pitch\_c1}$	Success probability for high–pitch actuation in cluster 1, derived from scenarios where sound is dominant.	$\left[\min_{sc} p^{*(sc)}_{\text{succ}}(\text{pitch}, c1), \max_{sc} p^{*(sc)}_{\text{succ}}(\text{pitch}, c1)\right]$
α_2	Distance-related reliability ratio for cluster 2. For each scenario with measurements on both clusters, compute $\alpha_2^{(sc)} = p^{*(sc)}_{\text{succ}}(c2) / p^{*(sc)}_{\text{succ}}(c1)$. White LED, which is empirically the most fragile, provides a pessimistic upper bound; red provides the optimistic lower bound. The global α_2 used in the MDP is the mean or median of all $\alpha_2^{(sc)}$.	$\left[\min_{sc} \alpha_2^{(sc)}, \max_{sc} \alpha_2^{(sc)}\right]$
β_1	Intensity coefficient from high to medium. For each scenario and colour where both intensities are measured, compute $\beta_1^{(sc)} = p^{*(sc)}_{\text{succ}}(\text{med}) / p^{*(sc)}_{\text{succ}}(\text{high})$. The same β_1 is then applied to all colours in the model.	$\left[\min_{sc} \beta_1^{(sc)}, \max_{sc} \beta_1^{(sc)}\right]$
β_2	Intensity coefficient from high to low, computed analogously as $\beta_2^{(sc)} = p^{*(sc)}_{\text{succ}}(\text{low}) / p^{*(sc)}_{\text{succ}}(\text{high})$. Again, a single global β_2 is applied to all colours.	$\left[\min_{sc} \beta_2^{(sc)}, \max_{sc} \beta_2^{(sc)}\right]$
θ	Motif coefficient for rectangular vs. line motifs. For each scenario with both shapes, compute $\theta^{(sc)} = p^{*(sc)}_{\text{succ}}(\text{rect}) / p^{*(sc)}_{\text{succ}}(\text{line})$. $\theta = 1$ corresponds to a neutral choice with no effect on motifs.	$\left[\min_{sc} \theta^{(sc)}, \max_{sc} \theta^{(s)}\right]$
γ	Scheduling coefficient for staggered (or sequential) vs. fully parallel execution. For each scenario sc, compute $\gamma^{(sc)} = p^{*(sc)}_{\text{succ}}(\text{staggered}) / p^{*(sc)}_{\text{succ}}(\text{parallel})$. $\gamma = 1$ is used when scheduling differences are ignored.	$\left[\min_{sc} \gamma^{(sc)}, \max_{sc} \gamma^{(sc)}\right]$

6.4. Worked Example from Scenarios

Consider scenario 1 which involves two line-shaped clusters (15 blocks each) executing tasks in parallel.[8] The relevant measurement for calibration is Task 1: white LED, high intensity on cluster 2.

Across five trials, the failure counts for cluster 2 are $7, 7, 6, 6, 6$ (failures per trial), giving $f = 32$ failures over $n = 75$ issued commands. Thus:

$$\hat{p}_{\text{fail}} = \tfrac{32}{75} = 0.4267, \qquad \hat{p}_{\text{succ}} = 0.5733.$$

Hoeffding correction for $n = 75$ and $\delta = 0.05$ yields $\varepsilon \approx 0.171$. The corrected failure probability becomes:

$$p^*_{\text{fail}} = \min(1, 0.4267 + 0.171) = 0.5977,$$

and the corrected success probability is $p^*_{\text{succ}} = 0.4023$.

Using cluster 1 data for the same colour–intensity pair, α_2 is computed:

$$\alpha_2 = \frac{p^*_{\text{succ},c2}}{p^*_{\text{succ},c1}}.$$

This ratio is then applied uniformly to red and blue LED probabilities due to the pessimistic behaviour of white under load.

Applying the protocol of Section 6.3 to all calibration scenarios yields the numerical constants reported in Table 2. For scenario 1, this results in $\alpha_2 = 0.9$, $\theta = 1.057$, etc.

[8] This corresponds to scenario 15 in our raw experimental log.

6.5. Integration into the MDPs

The calibrated constants instantiate the baseline ψ_{base} as defined in Sect.6.1. For cluster i, colour c, intensity j, motif m, and operation mode o, the baseline success probability is:

$$p_{\text{succ}} = q_{c,high,c1} \cdot (\alpha_2)^{[i=2]} \cdot \theta^{[m=\text{rect}]} \cdot \gamma^{[o=\text{stag}]} \cdot \beta_1^{[j=\text{med}]} \cdot \beta_2^{[j=\text{low}]},$$

where $[\cdot]$ denotes the indicator function, which evaluates to 1 when the condition holds and to 0 otherwise. So, the indicators select the applicable coefficients. The modelling assumption above is expressed by a product of calibrated coefficients (colour, intensity, motif, execution mode). This is a compact empirical abstraction that preserve relative reliability trends, not a strict independence assumption. Afterwards, hardware and environment parts override this value whenever their guards hold. The result is a compact, parameterised MDP that remains stable w.r.t. empirical variability while retaining behavioural fidelity to the real physical system.

7. MDP Execution with PRISM

This section describes how the calibrated multi-cluster MDP model is executed and analysed. The execution considered is performed *at design time* using the probabilistic model checking and strategy synthesis of PRISM; no online controller is deployed on the physical system. The purpose of execution is to explore alternative scheduling and adaptation choices under uncertainty and to evaluate their quantitative impact on system performance.

7.1. Executions under Schedulers

In our MDPs probabilistic transitions encode empirically calibrated success and failure rates, while non-deterministic choices represent alternative execution and adaptation decisions.

During MDP execution with PRISM, non-determinism is resolved by memoryless schedulers (also called policies or strategies) that select one enabled action in each state thereby inducing a fully probabilistic model. This mechanism allows us to reason about both best-case and worst-case behaviours by computing extrema over all admissible schedulers. We observed that the scheduler structure described in Sect. 6.1 is preserved during execution with PRISM: Hardware overrides everything; environment overrides only the baseline; otherwise, empirically calibrated probabilities apply.

7.2. Task Termination and Execution Cycles

Each cluster executes a task cycle that starts in an `idle` or `start_task` state and terminates either in a `finish` state or `task_failed` state. Failures are not modelled as fully absorbing in order to allow recovery through fallback transitions to lower-energy configurations, reflecting the adaptive nature of the system.

Execution across clusters is coordinated through synchronized labels (e.g., `[SYNC_STEP]`) and, when enabled, through a turn-based mechanism that implements staggered execution. This allows the same model to represent both fully synchronous cluster execution and interleaved execution scenarios, which are analysed comparatively in the case studies.

7.3. Rewards and Quantitative Measures

Quantitative analysis is performed using reward structures attached to states and transitions. In particular, the energy-related rewards shown in Sect. 2.3 capture the relative cost of different actuation levels and are scaled by cluster size. Additional rewards are used to count completed tasks or to penalise failures. Measurements showed near-linear growth in consumption with the number of active blocks within a range. Thus, the sum of per-block contributions is used to estimate the proximity to the supply limit rather than the exact current.

Running the MDP with these rewards, we compute properties such as the expected cumulative energy consumption until task completion, the probability of successfully completing a sequence of tasks, and trade-offs between energy usage and reliability. These measures provide the basis for comparing different motifs, execution modes, and adaptation strategies.

7.4. Tool Support and Scalability

All MDP executions are performed using PRISM, and some selected experiments are validated using STORM (Hensel et al. 2022) to benefit from symbolic engines when exploring larger parameter spaces. The use of parameterised clusters and abstract macro-states keeps the state space manageable while preserving the essential interaction between concurrency, environment feedback, and hardware constraints.

Overall, MDP executions in our framework serve as a systematic exploration of design alternatives under uncertainty, bridging empirically grounded modelling and formal quantitative analysis.

8. Back to the Case Study with MDPs

This section presents our case study with the proposed energy-aware multi-module MDP, grounded in empirical measurements obtained from a real BB system. We deliberately focus on a fully calibrated scenario to demonstrate, in a controlled and reproducible manner, how empirical data are integrated into the formal model and how probabilistic analysis reveals non-trivial adaptive behaviours.

8.1. Scenario and Empirically Calibrated Constants

The scenario considered involves two BB clusters that execute coordinated tasks that combine high-intensity LED activation. The system operates under a shared 5 V / 2 A power supply, and is subject to both hardware-level current feedback and environment-level sensing (microphone and accelerometer).

This scenario is selected because: (i) it was repeatedly executed on the physical platform, (ii) it exhibits both nominal and stressed operating conditions, and (iii) it triggers the full adaptation logic encoded in the managing subsystem.

Tens of experimental trials were conducted under different activation orders and actuation combinations. These experiments revealed consistent colour-dependent and energy-dependent failure patterns that were subsequently encoded as probabilistic transitions in the MDP. Table 2 summarises the constants used

Table 2 Empirically calibrated constants for Scenario 1

Parameter	Value
Motif scaling (θ)	1.057
Fallback scaling (β_1)	1.154
Fallback scaling (β_2)	1.154
Staggering factor (γ)	0.9725
Distance scaling (α_2)	0.9
High LED success (Red)	$q_r^{high} = 0.90$
High LED success (Blue)	$q_b^{high} = 0.90$
High LED success (White)	$q_w^{high} = 0.62$
High pitch success	$q_{pitch}^{high} = 0.80$

in the model for the selected scenario. All values are either directly measured or conservatively derived from empirical trends.

As expected, white LEDs exhibit the lowest success probability due to their higher current draw, followed by blue and red. The fallback coefficients β_1 and β_2 encode the empirically observed increase in reliability when transitioning from high-energy to lower-energy actuation modes. In some scenarios, cluster 1 requires a special β initialization β_{i1} whereas $\beta_{i1} \in [0.1, 1]$, which is not as global as β_1 and β_2. This is just because in cluster 1 a severe clamping to the β coefficient led to the model unreliability.

8.2. Verified Properties

We consider a set of PCTL properties that capture the core concerns of energy-aware adaptive executions.

- **Safety** via the probability that the system never reaches a failure state: $P_{\max\,/\,\min} =? [\, G(\neg failed) \,]$.
- **Liveness** via the probability of eventually completing the task: $P_{\max\,/\,\min} =? [\, F(success) \,]$.
- **Safe-until-success** via the probability of reaching success without failure: $P_{\max} =? [\, (\neg failed)\, U\, success \,]$.
- **Robustness under overload** via the success probability conditioned on high current: $filter(max, P_{\max} = ?[F(success)], hw\_state = 2)$.

Table 3 summarises the results obtained using PRISM. The results highlight the role of nondeterminism. Although adversarial schedulers can force early failure (low $P_{\min}$), the system admits strategies that achieve task completion with probability arbitrarily close to one. Importantly, even under hardware overload, the managing subsystem can enforce safe fallback behaviour that preserves eventual success. Figure 7 illustrates the evolution of the success probabilities along a representative execution path, providing an intuitive explanation for the bounds reported in Table 3.

This case study demonstrates three key contributions. First, empirical measurements directly parameterise probabilistic transitions, grounding the model in physical reality. Second, probabilistic model checking exposes rare but critical execution paths that are infeasible to discover experimentally. For example, the only way to deactivate clusters is to de-attach the chosen clusters from the system; Instead, the models allow us to deal with only the chosen activated clusters. Third, verified properties reveal which adaptation strategies already modelled by the MDP

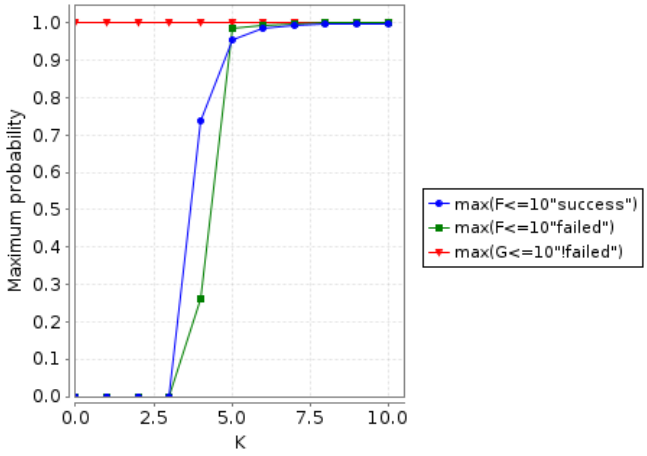


Figure 7 Verification of few properties with optimistic scheduler over bounded executions (here K is the number of steps)

under schedulers are sufficient to guaranty energy-aware task completion under uncertainty.

Table 3 Verification results for Scenario 1

Property	Pmin	Pmax
$G(\neg failed)$	1.0×10^{-10}	1.0
$F(success)$	0.0	0.99999997
$(\neg failed)\, U\, success$	0.0	0.9986
$F(success) \mid hw\_state{=}2$	–	1.0

9. Model-based vs. Experimental Validation

9.1. Models vs. Empirical Evaluation

Interactive MDP simulations with PRISM and Monte Carlo sampling allow exploring single or randomized paths, resolving non-determinism uniformly. Meanwhile, model-checking quantifies over all possible resolutions, including also the unfavoured paths that would never occur in real system. A comparison between empirical observations with the behaviour obtained through PRISM simulation for a scenario (parallel execution, line pattern) [9] is in Table 4.

The high-intensity task (T1) shows the strongest discrepancy between nominal execution and fallback behaviour. Empirically, Cluster 2 exhibits approximately 6–7 failures (≈ 0.42 probability), which corresponds to the prediction of the model: The system operates above the energy supply threshold. In the simulation, this configuration yields an accumulated reward of 300 energy units, exceeding the available supply (200 units). The probability of both clusters reaching success in the nominal path is about 0.50, but increases to 0.57 when a fallback to medium or low intensity is considered. For the medium-intensity task (T2), both empirical and simulated results indicate only occasional failures ($\approx$ probability 0.053) with a reward close to the supply limit (210 units). The low-intensity task (T3) remains fully stable in both experiments and simulation, with no observed failures and a reward of 165 units. Overall, these results

[9] Scenario n.15 on https://github.com/TONYSHOKRY/Formal-methods/blob/main/scenario_parameter_extraction.md

Table 4 Comparison between empirical observations and model results.

Metric	T1 (High)	T2 (Med)	T3 (Low)
Empirical failures (Cluster 2)	6–7 / 15	1	0
Empirical failure probability	≈ 0.42	≈ 0.053	0.0
Model reward (energy)	300 units	210 units	165 units
Model success probability	0.50 (high)	$\rightarrow$ 0.57 (med/low)	$\rightarrow$ 0.57 (med/low)

show that the model correctly captures the relationship between energy demand and reliability: As consumption exceeds the supply limit, the probability of fallback transitions increases.

9.2. Experimental Validation of Scenarios

In order to validate predictions derived from simulations with PRISM, experiments were performed on a 16-BB clustered square system with real hardware. One program with two distinct tasks was uploaded, corresponding to the configurations analyzed in a high intensity basic execution, and also with an adaptive execution with lower intensity.

For the first task, we programmed the blocks to run both clusters at a high LED intensity set *set_RGB(100,100,0)* for cluster 1, and *set_RGB(100,0,100)* for cluster 2 with a fixed activation time. The configuration executions resulted in block failures as predicted by the model. This manifested physically by either non responsive modules or failed to run modules, clearly visible in the left image of Fig. 8, where several BBs far and near the power supply in clusters 1 and 2 failed to maintain their intended intensity. This real experiment observation confirms the property verification results, where high intensity paths surpassed the threshold by 40 units, and flagged as risky, with a high probability of failure in such case.

The second task–an adaptive execution with lower intensity–intends to show what happens when a fallback to lower LED intensity occurs. We aim to validate the manual simulation done in the MDP adaptive execution by performing at lower intensities: *set_RGB(25,25,0)* for cluster 1 and *set_RGB(25,0,25)* for cluster 2, with the same fixed activation time on BBs. After an initial high intensity phase, this fallback mimics the adaptive transition proposed on executions with the generated PRISM strategy, where clusters are redirected to lower consuming states once a hardware detects high currents. Therefore, this resulted in stable execution across all BBs, with no failures at all, as shown in the right image of Fig. 8.

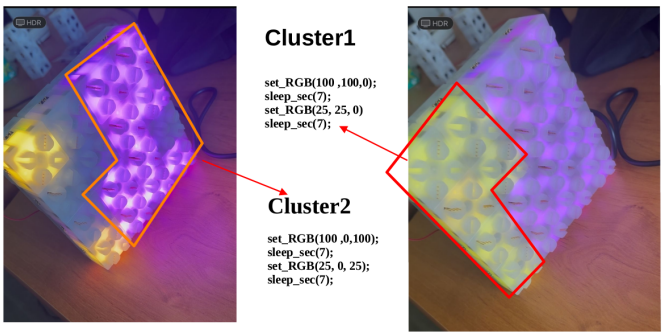


Figure 8 Scenario comparison

These experimental results comply with the PRISM-based predictions. The high-intensity scenario confirmed the model's detection of risky energy accumulation and failures, while the adaptive scenario demonstrated how fallback strategies effectively prevent failures, ensuring the full task execution.

10. Discussion and Related Work

The experimental results presented in Sect. 9 show the interest of our approach. Using PRISM MDPs reveals that a combinations of *baseline* + *environment* + *hardware* would identify worst-case hazards theoretically. These combinations allow simulating and verifying cases undiscovered experimentally. The MDP specification and model-based analysis show adaptation logic improving safety, energy bounds and system successful execution. The formal methods bring a level of abstraction: The full combinations between all metrics and parameters cannot be conceptually afforded, but we keep the model core while varying some of θ coefficients (e.g., LEDs targets, rewards, etc.)

Although PRISM-games support module renaming for replicating agents, the PRISM-MDP engine does not provide an equivalent construct, meaning that a new cluster cannot be instantiated simply by cloning an existing module while changing only its probabilistic constants. In our setting where clusters share structure but not parameters, this limitation unfortunately prevents us from expressing scalable multi-cluster systems directly in the modelling language. As a result, we generate cluster modules programmatically via Python, automatically rewriting constants, actions, guards, and labels to ensure structural consistency without manual duplication. Note that the cluster model could reach up to 10000 Blinky Blocks per cluster, it means the cluster size is not the limiting factor but rather the number of clusters in the system model. This limitation arises when fully building the state space. This workaround exposes a broader scalability constraint: PRISM's symbolic storage and the CSG heap rapidly exhaust memory as the number of clusters grows, whereas STORM (Hensel et al. 2022) can typically accommodate larger families of replicated modules. These observations underline that scalability is not only a modelling concern, but also an engineering one—-MDP engines differ markedly in how far they can scale structurally similar parallel components. In contrast, the over-approximation due to our non-determinism shows limitations, e.g., PRISM policies with adversarial paths that do not adequately represent real system. This could be improved with a feedback loop or real-time adaptation towards physical systems, while expecting PRISM engines capacity scaled up.

Although, when calibrating transition probabilities, we combined empirical measurements with assumed parameters when experiments under the exact same configuration were not available, and we applied clamping to ensure that probability distributions remain valid. We chose Hoeffding's inequality to limit the uncertainty of empirical estimates because, given our (still) small number of trials per scenario, its distribution-free guaranties lead to intervals that are nearly indistinguishable from those obtained by a Beta posterior with a uniform prior. The Beta distribution ("Beta Distribution" 2008) would pro-

vide a principled Bayesian update and credible intervals, but it does not resolve the issue of multiplicative probability inflation—clamping would still be necessary because Beta only constrains single-parameter estimates, not products of multiple calibration factors(Brewer et al. 2002). Hoeffding's approach, therefore, offers a simpler, model-independent method that matches the behaviour we would obtain from Beta-based smoothing while keeping the calibration pipeline consistent and lightweight. Furthermore, considering clusters as macro-agents requires more validation even if this abstraction seems to preserve the dominant relationships between energy demand, topology, and reliability, which are the focus of the analysis. Finally, the empirical calibration also mitigates external factors (heat, BB's misaligment, etc.) by aggregating multiple trials and by using conservative parameter values when instantiating the probabilistic transitions.

Related work. Assuring properties of distributed, multi-agent systems or CPSs under uncertainty often relies on probabilistic model checking against models like MDPs (Clarke et al. 2003). Existing approaches frequently rely on abstract or arbitrary assigned probabilities, separating such formal methods from the physical layer semantics driving failures in real systems. This physical grounding gap does not allow verification results to serve as a reliable basis for safety assurance and certification. Our work comes forward to help deriving the MDP's probabilistic transitions directly from empirically-validated physical threshold rules, ensuring the formal model is trustworthy and the analysis is meaningful with respect to the system's actual energy constraints. Recent work on Collective Adaptive Systems (CAS) (Wirsing et al. 2025-12) requires formal techniques to reason about adaptation and performance guaranties. Our framework makes use of the data collected to help calibrate models, on the one hand, and to analyse performance, predict/anticipate degradation on the other.

The application studied in this paper concerns static MRs. Nevertheless, our modelling and calibration workflow is transferable. For example, other MRs such as Datoms or Catoms (Abdel Ahad et al. 2026), where modular units interact through local actuation and sensing, can be analysed by a similar probabilistic calibration approach. Moreover, in simulation environments, such as VisibleSim, energy estimation could be integrated to enrich simulations before deploying protocols and algorithms on physical hardware. More complex MRs that include locomotion or mechanical coupling, e.g. robots with motors(Saintyves et al. 2024), may exhibit additional dynamics not captured in our model. However, we believe that the modelling principles following the proposals in (Päßler et al. 2025) and our approach with empirical calibration would capture the relationship between energy use, task execution, and likelihood of failure.

Simplified models used for quick analysis fail to capture the complexity, non-linear coupling between motifs shaping, concurrency, and voltage sag observed in hardware, on the one hand. On the other hand, high fidelity modelling of modular robotic systems, particularly concerning physics, is already a mature field. Standard approaches often involve co-simulation environments (e.g., MATLAB/Simulink) or custom solvers that use continuous-time differential equations (ODEs) to capture kinematic, dynamic, and detailed electrical circuit behaviours (Castano et al. 2002) However, these models are intractable for the state-space exploration required by formal verification tools like PRISM. We, therefore, offer a formal multi-module compositional specifications to find such a critical balance.

The combination of empirical data and formal methods is a key focus in software assurance.Approaches such as statistical model-checking (SMC) have successfully integrated runtime data to estimate model parameters or analyze rare events in CPS (Larsen & Legay 2016). Unlike classical probabilistic model checking, SMC provides a complementary analysis technique to estimate property satisfaction by sampling executions of a system or model. The existing hybrid approaches use testing to validate final system behaviour rather than validate the specification. This validated specification then serves as the immutable basis for the MDP's structure and its cost/reward functions, guaranteeing that the synthesized adaptive control policy is not only optimal mathematically, but also physically implementable and safe. Applying SMC to calibrated MDP constitutes a natural future work direction, as it would allow the cross-validation of worst-case results against statistically observed behaviour(Younes et al. 2006) and the evaluation of how conservative the adversarial guaranties are in practice.

Recent research on digital twins (Oo et al. 2025) emphasizes how formal verification techniques and decision-making under uncertainty can be used to evaluate the effectiveness of a digital twin by comparing predicted and observed behaviours and reasoning about perturbations. MDP models can be interpreted as an analytical core of a prospective digital twin for modular robotic systems providing design-time feedback and supporting adaptive control policies derived from model-checking analysis. Our work also aligns with the vision of formal reasoning, which is increasingly recognized as a key component of trustworthy digital twins. Indeed, recent frameworks suggest analysing interactions between digital and physical counterparts through safety and reachability reasoning to quantify trust and detect deviations, e.g., (Azzedin et al. 2026). We intend to develop the integration layer between the physical and digital counterparts in the spirit of (Oo et al. 2025).

11. Conclusion

This paper presented an energy-sensitive modelling approach based on MDPs for self-adaptive modular robotic systems. By combining empirical calibration with formal verification, the approach described in the paper bridges the gap between physical experimentation and model-driven analysis, enabling trustworthy reasoning about adaptive behaviour in modular robotic systems. The resulting MDP model supports adversarial verification, controller synthesis, and systematic exploration of energy-aware adaptation strategies under both probabilistic and non-deterministic uncertainty. The results also reinforce the interest and the advantage of integrating real-time current sensing, which enables the system to autonomously trigger energy-aware adaptations without manual intervention. A better understanding and prediction of the energy aspects of modular Blinky

Blocks systems with the aim of their dynamic adaptation pave the way for robust programmable matter projects.

References

Abdel Ahad, G., Awad, N., Piranda, B., Bassil, J., Werfel, J., & Bourgeois, J. (2026). Quantitative comparison of self-reconfiguration algorithms for 3D Catoms. In *Proc. Int. Symp. on Distr. Autonomous Robotic Systems (DARS)* (pp. 85–99).

Azzedin, F., Alhazmi, T., Abawajy, J., & Hammoudeh, M. (2026). Behavioral trust management in digital twin systems using safe state analysis and virtual validation. *Information and Software Technology*, *194*, 108093. doi: https://doi.org/10.1016/j.infsof.2026.108093

Bagheri, B., Yang, S., Kao, H., & Lee, J. (2015). Cyber-physical systems architecture for self-aware machines in industry 4.0 environment. *IFAC-PapersOnLine*, *48*(3), 1622-1627. doi: https://doi.org/10.1016/j.ifacol.2015.06.318

Baier, C., & Katoen, J.-P. (2008). *Principles of model checking*. MIT Press.

Beta distribution. (2008). In *The concise encyclopedia of statistics* (p. 40). New York, NY: Springer New York. Retrieved from https://doi.org/10.1007/978-0-387-32833-1_29 doi: 10.1007/978-0-387-32833-1_29

Bliudze, S., Furic, S., Sifakis, J., & Viel, A. (2019). Rigorous design of cyber-physical systems - linking physicality and computation. *Softw. Syst. Model.*, *18*(3), 1613–1636. doi: 10.1007/S10270-017-0642-5

Brewer, M. J., Soulsby, C., & Dunn, S. M. (2002). A Bayesian model for compositional data analysis. In *Proc. int. symp. in computational statistics COMPSTAT 2002* (pp. 105–110). Springer. doi: 10.1007/978-3-642-57489-4_10

Castano, A., Behar, A., & Will, P. (2002). The Conro modules for reconfigurable robots. *IEEE/ASME Transactions on Mechatronics*, *7*(4), 403-409. doi: 10.1109/TMECH.2002.806233

Clarke, E. M., Grumberg, O., Jha, S., Lu, Y., & Veith, H. (2003). Counterexample-guided abstraction refinement for Markov decision processes. *Journal of the ACM*, *50*(5), 752–794.

de Lemos, R., & et al. (2013). Software engineering for self-adaptive systems: A second research roadmap. *Dagstuhl Seminar Proceedings*, *10431*.

Hansson, H., & Jonsson, B. (1994). A logic for reasoning about time and reliability. *Int. J. on Formal Aspects of Computing*, *6*(5), 512–535.

Hensel, C., Junges, S., Katoen, J., Quatmann, T., & Volk, M. (2022). The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transf.*, *24*(4), 589–610. doi: 10.1007/S10009-021-00633-Z

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*(301), 13–30. doi: 10.1080/01621459.1963.10500830

IBM Corporation. (2006). An architectural blueprint for autonomic computing. In *IBM white paper.* (MAPE-K reference model)

Kwiatkowska, M., Norman, G., & Parker, D. (2011). Prism 4.0: Verification of probabilistic real-time systems. In *Int. conf. on computer aided verification (CAV 2011)* (Vol. 6806, pp. 585–591). Springer. doi: 10.1007/978-3-642-22110-1_47

Larsen, K. G., & Legay, A. (2016). Statistical model checking: Past, present, and future. In *Leveraging applications of formal methods, verification and validation: Foundational techniques ISoLA 2016, proceedings, part I* (Vol. 9952, pp. 3–15). doi: 10.1007/978-3-319-47166-2_1

Lee, P. A., & Messerschmitt, D. G. (2000). A globally asynchronous, locally synchronous architecture for large high-performance asics. In *IEEE symposium on asynchronous circuits and systems* (pp. 56–64).

Monostori, L. (2014). Cyber-physical production systems: Roots, expectations and R&D challenges. *Procedia CIRP*, *17*, 9-13. doi: https://doi.org/10.1016/j.procir.2014.03.115

Oo, K. H., Koomsap, P., & Ayutthaya, D. H. N. (2025). Digital twin-enabled multi-robot system for collaborative assembly of unorganized parts. *Journal of Industrial Information Integration*, *44*, 100764. doi: https://doi.org/10.1016/j.jii.2024.100764

Päßler, J., ter Beek, M. H., Damiani, F., Johnsen, E. B., & Tarifa, S. L. T. (2025). A configurable software model of a self-adaptive robotic system. *Sci. Comput. Program.*, *240*, 103221. doi: 10.1016/J.SCICO.2024.103221

Piranda, B., Lassabe, F., Tribhout, R., Lasserre, G., & Bourgeois, J. (2025). Creating an artwork with a modular robot composed of 1824 modules. In L. Barolli (Ed.), *Advanced information networking and applications* (pp. 366–377). Springer.

Saintyves, B., Spenko, M., & Jaeger, H. M. (2024). A self-organizing robotic aggregate using solid and liquid-like collective states. *Science Robotics*, *9*(86), eadh4130. doi: 10.1126/scirobotics.adh4130

Thalamy, P., Piranda, B., Naz, A., & Bourgeois, J. (2021). VisibleSim: A behavioral simulation framework for lattice modular robots. *Robotics and Autonomous Systems*, 103913. doi: https://doi.org/10.1016/j.robot.2021.103913

Weyns, D. (2019). Software engineering of self-adaptive systems. In S. Cha, R. N. Taylor, & K. C. Kang (Eds.), *Handbook of software engineering* (pp. 399–443). Springer. doi: 10.1007/978-3-030-00262-6_11

Wirsing, M., De Nicola, R., Jähnichen, S., & Tribastone, M. (2025-12). Rigorous engineering of collective adaptive systems – 3rd special section: part I. *Int. J. Softw. Tools Technol. Transf.*, *27*(6), 533–539.

Younes, H. L., Kwiatkowska, M., Norman, G., & Parker, D. (2006). Numerical vs. statistical probabilistic model checking. *Int. J. Softw. Tools Technol. Transf.*, *8*(3), 216–228.

About the authors

Kouchnarenko Olga is a full professor in computer science at UMLP. You can contact the author at olga.kouchnarenko@umlp.fr.

Lassabe Frédéric is an assistant professor in computer science at UMLP. You can contact the author at frederic.lassabe@umlp.fr.

Naguib Antonios is a PhD student in computer science at UMLP. You can contact the author at antonios.naguib@umlp.fr.