Model-Based Dependability and Performance Analysis for Satellite Systems with Collaborative Maintenance Maneuvers via Stochastic Games

Abdelhakim Baouya[a,*], Brahim Hamid[a], Otmane Ait Mohamed[b], Saddek Bensalem[c]

[a] *IRIT, Université de Toulouse, CNRS, UT2, 118 Route de Narbonne, 31062 Toulouse Cedex 9, France*
[b] *Concordia University, Montréal, Canada*
[c] *Université Grenoble Alpes, VERIMAG, CNRS, Grenoble, France*

Abstract

GPS Standard Positioning Service (SPS) relies on orbiting satellites to provide accurate time, location, and altitude information under all weather conditions, day or night, anywhere in the world. The lifespan of these satellites can vary depending on the specific version and dependability reference parameters. Engineers rely on dependability references to assess Reliability, Availability, and Maintainability (RAM) during the design phase, aiming to maximize a satellite's mean time between failures (MTBF). Furthermore, Integrity and Continuity are crucial performance ingredients for SPS. This paper proposes a formal parametrizable model based on concurrent stochastic games (CSG) to represent satellite constellations. The model incorporates formal specifications of dependability and performance, expressed in rPATL. Model parameters are derived from SPS standard characteristics established by the Space Operations Squadron to ensure the health and status of the operational constellation. Through the PRISM-games model checker, we conduct a quantitative analysis of collaborative behaviors between players in orbit and on the ground. We demonstrate the advantages of the CSG model in learning previous experiences and attempts to achieve efficient maintenance. Additionally, our analysis offers insights into the balance between maintenance efforts for orbital and ground-based personnel.

Keywords: Standard Positioning Service, Dependability, Performance, Concurrent Stochastic Games

Contents

*Corresponding author at : IRIT, Université de Toulouse, CNRS, UT2, 118 Route de Narbonne, 31062 Toulouse Cedex 9, France

Email addresses: abdelhakim.baouya@irit.fr (Abdelhakim Baouya), brahim.hamid@irit.fr (Brahim Hamid), otmane.aitmohamed@concordia.ca (Otmane Ait Mohamed), saddek.bensalem@univ-grenoble-alpes.fr (Saddek Bensalem)

1. Introduction

Satellite systems have become indispensable in our daily lives. The growing need for global communication has spurred innovation in the space industry, leading to increased competition among new space companies such as Starlink [1] and Amazon [2]. Satellite constellations are crucial in various applications, including internet connectivity, remote sensing (e.g., IoT and weather forecasting), and critical military operations. To ensure the success of these missions, dependability assurance, encompassing Reliability, Availability, and Maintainability (RAM) [3], is essential during the design phase. Prioritizing RAM helps minimize costs and complexities associated with potential repairs. Furthermore, ensuring communication integrity and continuity is indispensable for accomplishing successfully satellite mission goals.

The GPS Standard Positioning Service (SPS) is a widely accessible positioning and timing service used for civil, commercial, and scientific purposes. It employs specific signal codes and adheres to standardized definitions outlined in [4]. The GPS Space Segment (SPS) is controlled by a Control Segment (CS) consisting of a Master Control Station (MCS), ground antennas (GAs), and monitoring stations (MSs). The GPS constellation typically consists of 24 orbital slots, each containing at least one operational satellite. Each satellite communicates with a network of dedicated GAs and monitoring MSs (see Figure 1) located worldwide to ensure that GPS performance and reliability consistently meet the requirements of civilian users. The MCS coordinates information from multiple sources, ensuring alignment with performance standards set by qualified personnel monitoring both on-orbit and ground-based operations, as well as the timing commands generated by GAs and received by MSs.

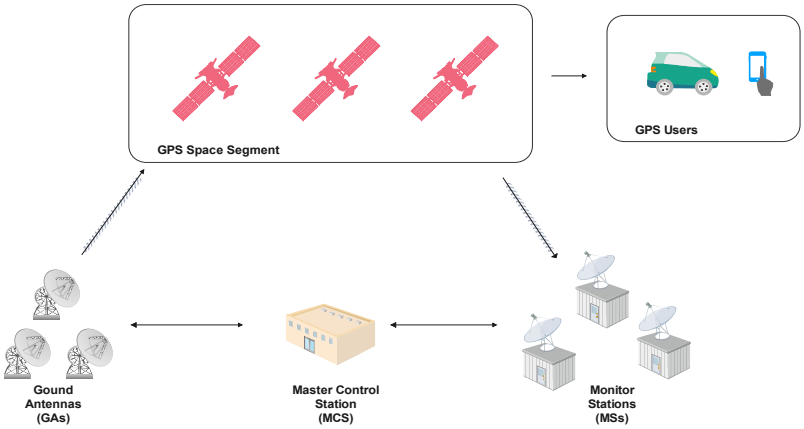


Figure 1: The GPS Control Segment (CS).

Satellite systems may be subject to various failure modes, impacting their performance and availability. SPS has identified two distinct error types [4], soft GPS failures and hard GPS failures. A soft GPS failure occurs when the SPS remains operational but experiences errors that could compromise performance, including communication accuracy, integrity, and continuity. In contrast, a hard GPS failure renders the SPS untrackable. Extensive research has been conducted on various techniques, including formal methods [5, 6, 7] and simulation [8, 9], to address dependability and performance issues. The majority of simulation results are documented in [4]. While this reference identifies numerous common errors, a foundational model is essential for predicting the long-term performance of orbital exercises, such as GPS satellite systems, which are expected to operate for 20 years. Formal models incorporating probabilistic failures can serve as a basis for such evaluations, while verification provides valuable insights into long-term execution.

Formal verification [10] is a powerful technique for ensuring the correctness and reliability of complex systems. It utilizes various formalisms, each suited to specific use case. Some common formalisms include Markov Decision Processes (MDPs), Continuous-Time Markov Chains (CTMCs), and Concurrent Stochastic Games (CSGs). Stochastic games verification [11] allows for the generation of quantitative correctness assertions about a system's behavior (e.g. "The object recognition system can correctly identify pedestrians

with a probability of at least 99%, even in challenging lighting conditions".), where the required behavioral properties are expressed in quantitative extensions of temporal logic. The problem of strategy synthesis constructs an optimal strategy for a player, or coalition of players, to ensure a desired outcome (property) is achieved. The formalism of Concurrent stochastic multi-player games (CSGs) [11, 12] permits players to choose their actions concurrently in each state of the model. This approach captures the true essence of concurrent interaction, where agents make independent choices simultaneously without perfect knowledge of others' actions. However, although algorithms for verification and strategy synthesis of CSGs have been implemented in PRISM-games[13], their adoption for RAM analysis has not been investigated.

Contribution. This paper extends our previous work presented at SEAA Conference 2024 [14] by demonstrating how to accurately model satellite systems and verify their dependability and performance properties using the PRISM-games model checker. The previous paper comprehensively addresses key dependability characteristics, drawing on research insights from [5, 6, 7]. By leveraging the documentation provided by the GPS SPS standards (fifth edition) [4], we incorporate performance-related metrics such as continuity, integrity, and additional failure modes probabilities. To implement dependability and performance metrics, we leverage the PRISM-games model checker. The PRISM-games tool extends the capabilities of classical probabilistic model checkers, which have been widely applied to verify the correctness and effectiveness of hardware and software designs [15]. However, classical models in probabilistic automata (PA) often focus on a single perspective. Our approach leverages Concurrent Stochastic Games (CSGs) to capture the collaborative maintenance and repair of satellite systems [5, 6, 7]. This allows us to model multiple players involved in the system's operation (which introduces failures and planned/unplanned interruptions), on-orbit maintenance maneuvers (responsible for software update moving to a redundant satellite), and ground control (responsible for satellite preparation and launch). CSGs are particularly well-suited for this task, effectively representing failure management, collaborative maintenance, and personnel learning from past experiences (as previously adapted strategies).

Outline. The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 provides the necessary background on Concurrent Stochastic Games (CSGs) and the PRISM-games language. Section 4 presents our proposed modeling approach for satellite systems. We then perform a quantitative analysis of the satellite system's behavior under failure scenarios in Section 5. Finally, Section 6 concludes the paper and suggests directions for future research.

2. Related work

Dependability and performance analysis of Cyber-Physical Systems (CPS) necessitates a comprehensive approach, encompassing simulation and verification activities. The specific methods employed for these tasks can vary based on the individuals' expertise. In addition, many CPS systems rely on specific standards that influence the requirements for various activities. For example, automotive safety standards like ISO26262 [16, 17, 18] often prioritize simulation while placing less emphasis on verification. In transportation systems, the European Railway Traffic Management System (ERTMS) [19] (based on the Global System for Mobile Communications - Railway standard) requires rigorous testing [20] and validation to ensure its safety and reliability. Formal verification techniques are increasingly being used to assess the correctness and robustness of ERTMS components as identified through ERTMS conferences [21, 22, 23]. In avionics systems, the DO-178C standard [24] is primarily concerned with the software aspects of avionics systems. It defines a process for developing and verifying software that is used in airborne systems, including flight control systems, navigation systems, and communication systems. Although the standard emphasizes verification, the nuance regarding the use of formal methods is to support the qualification of software tools while testing is also recommended [25]. Previous standards primarily emphasize the anticipation of critical failures in sensitive and crucial components, along with the planning of subsequent maintenance. Satellite systems rely on international communication consortia like the European Telecommunications Standards Institute (ETSI), the International Telecommunication Union (ITU), and the Federal Communications Commission (FCC) to regulate satellite communication and radio frequencies. While these organizations establish standards,

governments often prioritize technologies that prioritize accuracy, performance, and continuity, such as GPS and the Russian Global Navigation Satellite System (GLONASS). This emphasis on specific technologies can lead to a divide-dependent approach. The GPS Standard Positioning Service (SPS), maintained by the U.S. government, has documented the evolution of its satellite systems in a reference document [4] titled "*GPS Standard Positioning Service (SPS) Performance Standard*." This document outlines key dependability and performance metrics. Although the reference document outlines the collection of key metrics through operational design monitoring, it lacks a reported quantification tool for verification. Therefore, the quantification analysis presented in this research paper demonstrates the value of such an approach for strategy adaptation and maintenance planning.

Formal methods provide a robust solution for mathematically analyzing critical issues within CPSs[26, 27, 28, 29]. These methods have been widely adopted in European projects to assess the fidelity of requirements, primarily expressed in temporal logic and supported by formal proof [30]. The choice of formal methods depends on user capabilities and preferences. Graphical tools like UPPAAL for real-time model checking and Autofocus AF3 [31]for component-based architecture are well-suited for architect users. While experts in formal representation often favor textual expressivity tools like PRISM, it is important to note that PRISM can effectively capture a wide range of Markovian models, including Markov Decision Processes, Continuous-time Markov chains (CTMCs), and more recent formalisms for stochastic games. For further information on Markovian formalisms and stochastic games, readers can refer to the references [32] and [33]. The shift in stochastic games [12, 13, 34, 35] supported by PRISM-games model checker [36] based on a game-theoretic approach enables the design of protocols that utilize penalties or incentives to mitigate the impact of selfish players [13]. Within this semantic framework, the model can be conceptualized as a collection of players employing strategies to determine their actions based on the current state of the execution. This phenomenon is more prevalent in collaborative actions where multiple players strive to ensure the correct execution of a task [37, 38, 39], such as autonomous robots [40] and satellite system maintenance [14] where humans are playing a central role.

Human intervention plays a pivotal role in CPS maintenance. As emphasized by Cámara et al. in their research [41, 42], collaborative human maneuvers can significantly optimize system performance. Notably, their work in [42] incorporates the human factor as a state of availability within the model. In our previous research [14], we explored the scenario of two operational staff responsible for maintaining satellite systems both in orbit and on the ground, with their strategic plan consisting of a set of tasks. Efficient reliability, availability, and maintainability are critical game goals that staff operations have to achieve. While comparing to the added value of our research other papers have investigated the maintenance of satellite systems [5, 6, 7]. In [7], Peng et al. propose modeling a satellite system using Continuous-Time Markov Chains (CTMCs). This approach allows them to portray the impact of various factors on satellite reliability, including failures related to solar radiation and maintenance. Hoque et al. in [5] build upon the model presented in [7] by incorporating Erlang distributions instead of the exponential distributions supported by CTMCs. This change leads to more accurate results when comparing qualitative findings. Lu et al. in [6] model the system using Markov Decision Processes (MDPs) to account for communication between the satellite system and ground stations. Additionally, they utilize the $\pi - calculus$ to model the system's semantics. Building on the findings of [7], Peng et al. in [43] model the reliability of a satellite constellation using CTMCs. However, the impact of human interaction on maintenance costs is not addressed in any of the contributions as a game model. Farias et al. in [3] provide stochastic Petri nets to analyze their operational efficiency. By evaluating reliability, availability, and maintainability, this approach aids satellite designers in making informed decisions about constellation configurations. While the approach builds upon the work of [7, 43], it does not emphasize the human as a central factor in system performance. Unlike previous studies, the work in [3] quantitatively demonstrates redundant satellites' significant impact on reliability and availability, as highlighted in [5]. The user-friendly graphical tool in [3] is designed to assist architects, rather than those with formal backgrounds, in understanding and addressing these critical aspects of system design.

While previous research has focused on using PRISM with MDP, CTMC and PetriNet formalism to model satellite systems, our approach extends the work in [14] by incorporating a more comprehensive representation of the system's behavior, including failures and human satellite interactions. This enables

us to conduct a more in-depth analysis of system performance and forecast dependability while considering human maintenance attempts, as the results can be effectively synthesized into actionable maintenance strategies.

3. Background on Concurrent Stochastic Games

Concurrent stochastic multi-player games (CSGs) [11, 12] is an extension of Probabilistic Automata (PA) PA [44] by introducing concurrent decision-making, where players simultaneously choose actions in each state. In CSGs, each player controls one or more modules, and the actions associated with commands within a player's modules are exclusive to that player.

To express the coalition game, we rely on PRISM [10]. The PRISM model consists of modular components that synchronize with one another based on module actions. Each module is characterized by its variables, commands, and the state transitions they induce. A command takes the form:

$$[a_1, \ldots, a_n]\ g\ \rightarrow\ p_1 : u_1 + \ldots + p_n : u_n$$

or,

$$[a_1, \ldots, a_n]\ g\ \rightarrow\ u$$

This means that if the guard g is true, then an update u_i is enabled with a probability p_i for the conjunction of actions $a_i,\ i = 1, \ldots, n$. A guard is a boolean formula constructed from the variables of the module. The update u_i is an evaluation of variables expressed as a conjunction of assignments: $v_i' = val_i + \ldots + v_n' = val_n$ where $v_i \in V$, with V being a set of local and global variables, and val_i are values evaluated via expressions denoted by θ such that $\theta : V \rightarrow \mathbb{D}$, where $\mathbb{D}$ is the domain of the variables. CSGs are augmented with reward structures [11, 12]. The action reward function, denoted as $r_A : S \times A \longrightarrow \mathbb{R}$, assigns a real value to each state-action pair $(s \in S, a \in A)$. This value is accumulated when the action a is selected in state s. Additionally, the state reward function, denoted as $r_S : S \longrightarrow \mathbb{R}$, assigns a real value to each state s. This value is accumulated when the state s is reached.

The properties related to CSGs are expressed using rPATL (Reward Probabilistic Alternating Temporal Logic) [45], an extension of CTL (Computation Tree Logic) [46] with the coalition operator $\langle\langle C \rangle\rangle$ from ATL (Alternating-time Temporal Logic) [47] and the probabilistic operator P from PCTL (Probabilistic Computation Tree Logic) [48]. The property grammar is based on CTL, augmented with these additional operators.

For instance, the following property, expressed in natural language, can be formally defined using rPATL: *Two communicating entities have a strategy to ensure that the probability of payload tampering within k rounds is less than 0.001, regardless of the strategy of attackers*:

$$\langle\langle 1, 2 \rangle\rangle \mathtt{P}_{<0.001}[\mathtt{F}\ (\mathtt{tampering}\ \&\ \mathtt{rounds} = \mathtt{k})],\ \mathtt{k} = 1 : 1 : 100$$

Here, `tampering` is the label that refers to the states of modified data in transit. Regarding the reward structure, the property expressed in natural language: *What is the reward r within k rounds to reach* `tampering` *for both communicating entities (Players 1 and 2) for a selected strategy?* is expressed in rPATL as:

$$\langle\langle 1, 2 \rangle\rangle \mathtt{R} = ?[\mathtt{F}\ (\mathtt{tampering}\ \&\ \mathtt{rounds} = \mathtt{k})],\ \mathtt{k} = 1 : 1 : 100$$

Where k refers to the round number during model construction, with increments of 1 unit and a maximum round of 100 units.

Example 1. *Consider a CSG model of a client-server pattern, as presented in [14, 37], where a player attempts to read and write an object asynchronously using a buffer size of 1. The client has a 50% probability of performing a read operation and a 50% probability of performing a write operation. A player who successfully writes an object is considered the winner and receives a payoff of 1, while the other player receives a payoff of -1. If both players attempt to read or write simultaneously, no winner is declared. In the PRISM language, the arbiter monitors the game, using player actions to label its transitions and record the winner*

and loser. $A = \{(unlock_1 unlock_2)(r_1 r_2), (w_1 w_2), (w_2 r_1), (r_2 w_2), (reset_1 reset_2)\}$. *The monitor requests that both players unlock to perform read/write actions. After these actions are completed, the players perform a reset to prepare for the next round. The CSG starts in state* s_0, *and states* s_1, s_2, *and* s_3 *are labeled with*

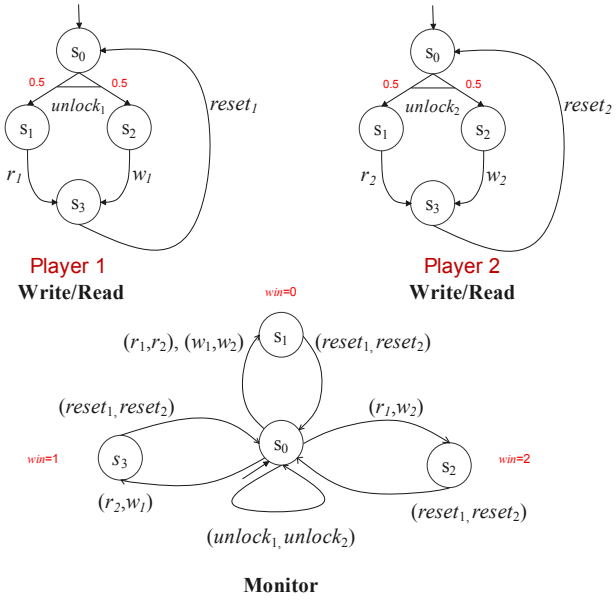


Figure 2: Read and Write Game Model in CSG [37].

atomic propositions corresponding to a player winning. Considering the modeled system in Figure 2, when Player 1 initiates the game and emerges victorious by writing, the property is expressed as:

$$\langle\langle 1 \rangle\rangle \mathtt{P}_{>0.99} =? [\mathtt{F\ win} = 1]$$

The model and properties associated with the example are available in [49]. The monitor which is a non-player module orchestrates read and write operations in the PRISM code shown in Listing 1. All commands in the monitor are labeled with at least two actions, corresponding to the players responsible for triggering the internal write and read operations. The **`win`** *variable defines the player's success in writing, taking values 1 or 2. The initial command, recorded on line 5, represents the unlock action, triggering write or read operations by the players. The subsequent commands on lines 6-7 depict unscheduled write (or read) operations. Once these operations are executed, a reset command is introduced on line 9 to indicate an idle state. Finally, the commands on lines 10-11 record the write/read actions performed by the players.*

To model the payoff, we introduce two reward structures, one for each player, reflecting their utility in each round of read/write operations. These rewards can be assigned as follows: 1 for winning, 0 for drawing, and -1 for losing. To represent these choices, we again use action lists corresponding to the different options available to the players as in Listing 1 lines 13-25. To calculate the payoff for the first player, we can express the reward in the following form :

$$\langle\langle 1 \rangle\rangle \mathtt{R} =? [\mathtt{F\ win} = 1]$$

4. Formal Modeling of Satellite Systems

PRISM-games is a probabilistic model checker for analyzing Concurrent Stochastic Games (CSGs) involving multiple players. It verifies properties in rPATL (an extension of PCTL), enabling reasoning about probabilities and rewards. This allows for the creation of state-based models, such as the single satellite system shown in Figure 3. The new maintenance plan extends the version presented in [14].

Listing 1: PRISM Code for Read/Write of Figure 2

```
module Orchestrator
 win : [0..2] init 0;
 s : [0..3] init 0;

 [unlock1, unlock2] s=0 -> (s'=0) & (win'=0);
 [w1, w2] s=0 -> (s'=1) & (win'=0);
 [r1, r2] s=0 -> (s'=1) & (win'=0);

 [reset1, reset2] s=0 |s=1 | s=2 | s=3 -> (s'=0) & (win'=0);
 [r1, w2] s=1 -> (s'=2) & (win'=2);
 [w1, r2] s=1 -> (s'=3) & (win'=1);
endmodule
rewards "utility1" // utility of player 1
        [r1,r2] true : 0;
        [r1,w2] true : -1;
        [w1,r2] true : 1;
        [w1,w2] true : 0;
endrewards

rewards "utility2" // utility of player 2
        [r1,r2] true : 0;
        [r1,w2] true : 1;
        [w1,r2] true : -1;
        [w1,w2] true : -1;
endrewards
```

4.1. The system model

This paper leverages a previously established satellite model described in [5, 6, 7] with extended parameters stemming from the standard documentation [4]. The model integrates performance aspects related to continuity, integrity, availability, and human-in-the-loop considerations. All performance standards in the GPS SPS Performance Standard are expressed at a 95% probability level, in accordance with[4].

4.1.1. Continuity

The model considers the system's vulnerability to both scheduled and unscheduled interruptions throughout its lifecycle. Scheduled interruptions occur due to maintenance or software updates, leading to temporary signal unavailability for a fixed duration of t_α =4,320 hours. Scheduled interruptions, announced at least 48 hours in advance, do not contribute to loss of continuity. Two types of scheduled interruptions can lead to integrity risks: soft failures and satellite O&M activities. Soft failures, such as inaccurate Earth orientation predictions or errors in clock frequency, can compromise system integrity. Certain routine O&M activities, such as satellite maneuvers and atomic clock maintenance, while necessary, can also temporarily degrade system performance and lead to integrity failures. The expected Mean Time Between Failures (MTBF) for soft failures is 7.5 years. Additionally, the estimated duration of Satellite Operations and Maintenance (O&M) activities is 0.5 years.

Unscheduled interruptions, such as those caused by solar radiation, can induce a Single Event Upset (SEU) in the satellite's signal. Unlike scheduled events, SEUs are unpredictable but also self-correcting, resolving automatically. Permanent failures (referred to as hard failures), however, necessitate maneuvers in orbit or on the ground. Hard failures are subdivided into short (ST) and long-term (LT) failures (Detailed in section 4.1.4). Ground and orbit control evaluate the best course of action upon satellite failure. In some cases, the issue might be resolved remotely by sending software commands to the satellite (SatZap command). If the problem persists for LT failures, deploying a redundant satellite from orbit as a replacement may be necessary. However, if no backup satellite is available, a new one will need to be manufactured and launched, introducing the risk of a launch failure. The probability of a satellite failure is $1 - r$, where r is its reliability calculated from the failure rate and Mean Time Between Failures (MTBF). Both the unscheduled and scheduled interruption times are $t_\alpha = 4320h$. The expected MTBF for LT hard failures is approximately 15 years, while the expected MTBF for ST hard failures is around 0.5 years.

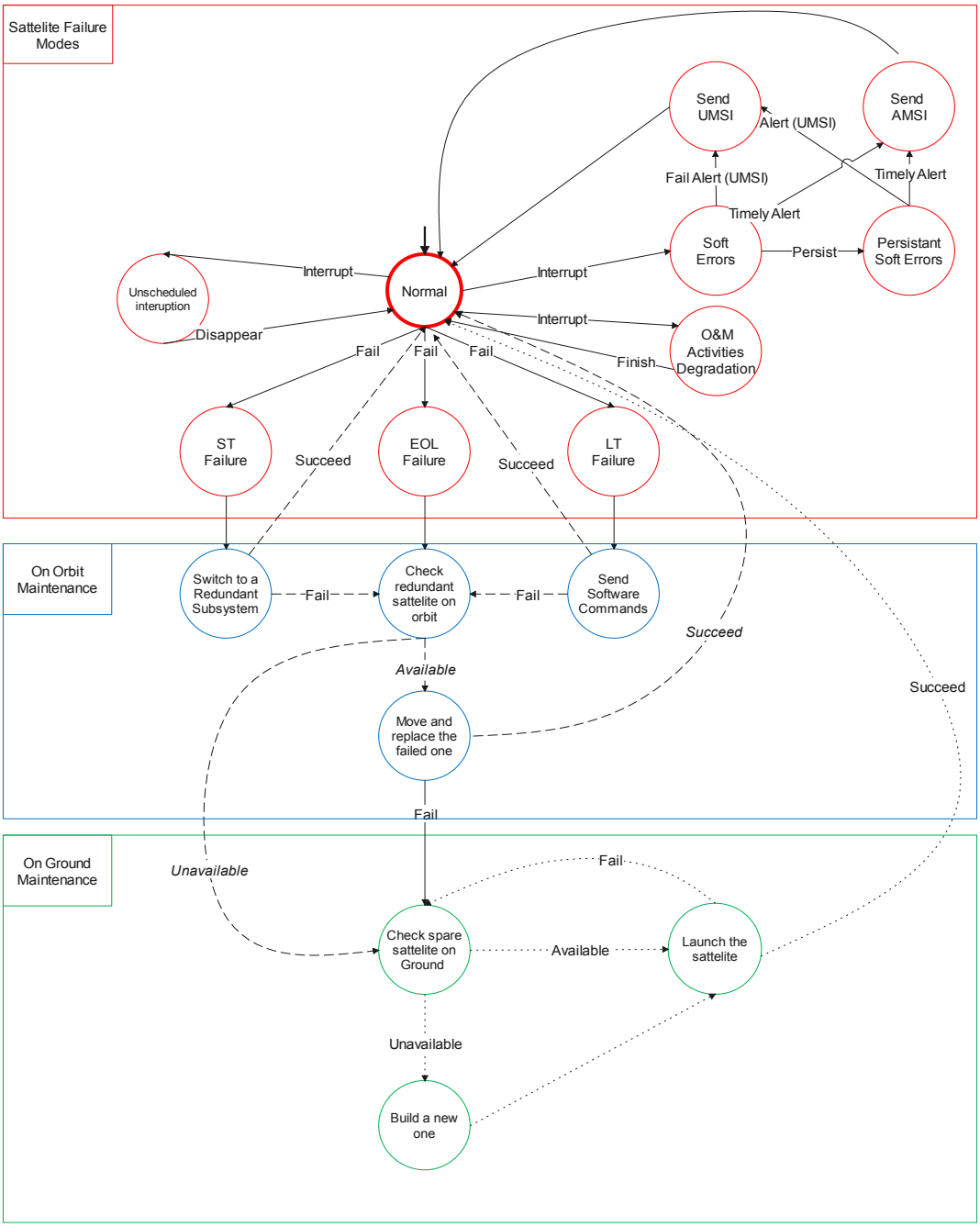


Figure 3: Extended Satellite Maintenance Process Model [14]

4.1.2. Integrity

Integrity refers to the trust in the correctness of the positioning information provided by the system. It also includes giving timely alerts when the system should not be used for positioning in case of soft failure mode. On average (50%), GPS SPS integrity faults persist for approximately 1 hour. A timely alert is defined as one provided no later than 10 seconds after an error exceeds the tolerance. The statistics indicate that, on average, approximately 3 integrity failures occur across the entire constellation per year. Additionally, the probability of a single GPS SPS error exceeding the NTE tolerance without timely alerts is 10^{-5}. In contrast, the likelihood of two or more satellites exceeding the NTE tolerance without timely alerts is 10^{-8}. If an integrity fault occurs and an alert is transmitted within 8 seconds, the misleading signal-in-space information (MSI) is considered to be converted to alerted information (AMSI). However, if the alert is not transmitted within 8 seconds, it constitutes an unalerted MSI (UMSI). In this case, the UMSI is considered a loss of integrity.

4.1.3. Availability

Satellite systems can experience various types of failures, including hard and soft failures. Satellites may reach their End-of-Life (EOL) and require replacement. It is assumed that each satellite has a 60% probability of reaching EOL and a 40% probability of experiencing an early failure due to a Long-Term (LT) hard failure. The constellation availability varies depending on the number of occupied slots. As the number of occupied slots increases, the availability decreases. For instance, if 24 slots are occupied, the availability is 72%, while for 21 occupied slots, the availability is 98%. We refer to Table 1 from the standard [4] where the probabilities were collected. The abbreviation is used for model checking.

Constellation Slots	Abbreviation used in the paper	Availability Probability
All 24 Slots	A1	0.720
23 or More Slots	A2	0.890
22 or More Slots	A3	0.954
21 or More Slots	A4	0.980

Table 1: Constellation Availability [4].

4.1.4. 2SOPS personnel operations

Highly skilled 2SOPS personnel (shortened to 2nd Space Operations Squadron) at the Master Control Station (MCS) are responsible for maintaining satellite health, detecting anomalies in the GPS system, and responding to these anomalies promptly. If an LT hard failure occurs, there is an $p_\beta = 80\%$ chance of resolving it on orbit by replacing the faulty satellite with a redundant one. If a satellite experiences an ST hard failure, a relatively rapid process is initiated to switch to a redundant subsystem, mitigating the impact of the failure. If on-orbit repair is impossible, a new satellite needs to be built. The ground control team manages this process. The time taken to decide to build a new satellite and for one to be manufactured is $t_\gamma = 24$ hours and $t_\delta = 24$ hours, respectively. Following a successful launch with $p_n = 90\%$, it takes another $t_k = 24$ hours for the new satellite to reach its operational position.

4.2. On orbit and ground support capabilities

The modeled system involves multiple state transitions driven by the *Satellite Behavior*, *on-orbit Staff*, and *ground Staff*. Each group performs specific tasks:

- *Satellite Behavior*: Triggers scheduled, unscheduled, and failure events (represented by the environment player: $\mathscr{P}_{env}$) shown in red in Figure 3. Their actions are labeled as α.
- *On-Orbit Staff*: These personnel ($\mathscr{P}_o$) handle tasks like sending commands, updating software, and maneuvering the satellite into the position shown in blue in Figure 3. Their actions are labeled as β.

- *On-Ground Staff*: The ground control team (represented by $\mathscr{P}_g$) is responsible for monitoring satellite health, building new satellites when necessary, and performing launches (shown in green in Figure 3). Their actions are labeled as ω.

Regarding the semantics of CSGs, the modeled system involves three players: $\mathscr{P}_{env}$, $\mathscr{P}_o$, $\mathscr{P}_g$. Each player is semantically described as a Probabilistic Automaton (PA). PA [44] is a modeling formalism that exhibits probabilistic and nondeterministic features implemented by PRISM. Definition 1 formally illustrates a PA of *Satellite Behavior*, *on-orbit Staff*, and *ground Staff* where $Dist(S)$ denotes the set of convex distributions over the set of states S and μ is a distribution in $Dist(S)$ that assigns a probability $\mu(s_i) = p_i$ to the state $s_i \in S$.

Definition 1. A PA is a tuple $\mathscr{P} = \langle s_0, S, \Sigma, AP, L, \delta \rangle$:

- s_0 is an initial state, such that $s_0 \in S$,
- S is a set of states,
- A is a finite set of actions to perform synchronization,
- $L : S \to 2^{AP}$ is a labeling function that assigns each state $s \in S$ to a set of atomic propositions taken from the set of atomic propositions (AP), and
- $\delta : S \times A \to Dist(S)$ is a probabilistic transition function assigning for each $s \in S$ ($s = \langle x_i, \theta \rangle$) and $a \in \{\alpha, \beta, \omega\}$ ($\alpha \in A_{env}, \beta \in A_o, \omega \in A_g$) a probabilistic distribution $\mu \in Dist(S)$. The operational semantics rule models the generic local probabilistic transitions:

$$\frac{x_i \xhookrightarrow{g:a}_p x_i' \wedge \theta \models g}{\langle x_i, \theta \rangle \xrightarrow{a}_p \langle x_i' \theta' \rangle}$$

  where $\theta' = \theta[v_i := effect(v_i)]$, The function $effect$ is any mathematical operation that updates the variable v_i. A player's state is defined by its variables' values, as defined in Definition 2.

Definition 2 ($\mathscr{P}$ state). A state of $\mathscr{P}$ is the valuation of a player $\mathscr{P}_i$ variable v_i ($i \in \{env, o, g\}$) in the form $s = \langle x_i, \theta \rangle$ such that $\theta \models g$ and g is the guard over states values that enable the transition from one player state to the next one, where x_i is the valuation of variable v_i, such that $x_i = effect(v_i)$.

To model composability, we introduce a dedicated PRISM module that acts as a non-player. This module encapsulates the environment, on-orbit actions, on-ground actions, and the PRISM commands needed to synchronize their interactions. We define a CSG model, G, as a game modeling the parallel composition of the environment player $\mathscr{P}_{env}$, on-orbit staff player $\mathscr{P}_o$, and ground staff player $\mathscr{P}_g$. This composition is coordinated by the non-player model $\mathscr{P}_{\mathscr{R}}$.

Definition 3. A concurrent stochastic game (CSG) for reasoning on Sattelite system maintenance is a tuple $G = \langle N, S, \bar{S}, A, \delta, AP, L \rangle$:

- $N = \{\mathscr{P}_{env}, \mathscr{P}_o, \mathscr{P}_g\}$ is a finite set of players,
- $S = S_{env} \times S_o \times S_g$ is a set of states (with semantics defined in Definition 2), where S_{env}, S_o, and S_g are states controlled by the system model, the environment model $\mathscr{P}_{env}$, the on-orbit player model $\mathscr{P}_o$, and on-ground model $\mathscr{P}_g$, respectively ($S_{env} \cap S_o \cap S_g = \emptyset$), and $\bar{S} \subseteq S$ is a set of initial states,
- $A = A_{env} \times A_o \times A_g$ where A_{env}, A_o, and A_g are the actions available to the environment model, on-orbit model, and the on-ground model, respectively,
- $\delta : S \times A \longrightarrow Dist(S)$ is a probabilistic transition function. Each player $\mathscr{P}_{env}, \mathscr{P}_o, \mathscr{P}_g$ selects an action α, β, ω, the state of the game is updated according to the distribution $\delta(s, (\alpha, \beta, \omega)) \in Dist(S)$,

- AP is a subset of all predicates that can be built over state variables. AP includes:
  - *goal*, satisfied in the state where a successful operation is reached.
- $L : S \longrightarrow 2^{AP}$ is a labeling function that assigns each state $s \in S$ to a set of atomic propositions (AP).

Following the definition of the CSG players, The non-player commands of $\mathscr{P}_{\mathscr{R}}$ that record the strategy of the CSGs model are expressed through the following transition: $s_m \xrightarrow{\alpha,\beta,\omega} s'_m$, where α represents the label of the environment commands, β denotes the on-orbit command, and ω denotes the on-ground commands. The non-player is modeled by the operation semantics rules *On-Orbit* and *On-Ground*. The *On-Orbit* is achieved by composing the modules $\mathscr{P}_{env}$, $\mathscr{P}_o$, and $\mathscr{P}_g$. The *standby* action refers to the idle position of the player in the CSG model. In this composition, the probability of achieving on-orbit or on-grounds tasks is determined by $\prod_{i=1}^{|N|} p_i$ such that $p_i \in \mathbb{R}$.

$$\frac{[\![\mathscr{P}_{env}]\!] = x_i \xhookrightarrow{\alpha}_{p_1} x'_i \bigwedge_{j=0}^{|A_o|} [\![\mathscr{P}_o]\!] = x_j \xhookrightarrow{\beta_j}_{p_2} x'_j \wedge [\![\mathscr{P}_g]\!] = x_k \xhookrightarrow{\omega}_{p_3} x'_k \wedge [\![\mathscr{P}_{\mathscr{R}}]\!] = x_m \xhookrightarrow{\alpha,\beta,\omega} x'_m}{\langle x_i, \ldots, x_j, \ldots x_k, \ldots x_m, \theta\rangle \xrightarrow{\alpha,\beta,\omega}_{p_1 \cdot p_2 \cdot p_3} \langle x'_i, \ldots, x'_j, \ldots, x'_k, \ldots, x'_m, \theta'\rangle} \quad (\textit{On-Orbit})$$

where $\alpha = Fail \wedge \omega = standby$

$$\frac{[\![\mathscr{P}_{env}]\!] = x_i \xhookrightarrow{\alpha}_{p_1} x'_i \wedge [\![\mathscr{P}_o]\!] = x_j \xhookrightarrow{\beta}_{p_2} x'_j \bigwedge_{k=0}^{|A_g|} [\![\mathscr{P}_g]\!] = x_k \xhookrightarrow{\omega_k}_{p_3} s'_k \wedge [\![\mathscr{P}_{\mathscr{R}}]\!] = x_m \xhookrightarrow{\alpha,\beta,\omega} x'_m}{\langle x_i, \ldots, x_j, \ldots x_k, \ldots x_m, \theta\rangle \xrightarrow{\alpha,\beta,\omega}_{p_1 \cdot p_2 \cdot p_3} \langle x'_i, \ldots, x'_j, \ldots, x'_k, \ldots, x'_m, \theta'\rangle} \quad (\textit{On-Ground})$$

where $\alpha = Fail \wedge \beta = standby$

Figure 4: Human-in-the-Loop Operational Semantics Rules For CSG.

4.3. Evaluating the effectiveness of collaborative maneuvers

Assessing the performance of collaborative maneuvers necessitates developing a strategy for the players $\mathscr{P}_{env}$, $\mathscr{P}_o$, and $\mathscr{P}_g$ that has the objective of reaching a state-satisfying goal and maximizes the value of the reward. The specification for the synthesis of such strategy is given as rPATL property following the pattern :

$$\langle\langle \mathscr{P}_{env}, \mathscr{P}_o, \mathscr{P}_g \rangle\rangle \mathtt{P} =?[\mathtt{F}\ goal]$$

Where $goal = (\text{“}win\text{”} \,\&\, rounds <= k)$ to quantitatively evaluate the efficacy of collaborative maneuvers up to the round k. The label “win” refers to the state satisfying the resolved operation by each player $\mathscr{P}_{env}$, $\mathscr{P}_o$, and $\mathscr{P}_g$ that take the following form: *label* “win” $= (v_i \oplus x_i) \otimes (v_j \oplus x_j) \otimes (v_k \oplus x_k)$ such that $\otimes \in \{\wedge, \vee\}$ is logical operators and $\oplus \in \{=, <, \leq, >, \geq, \neq\}$ is inequality Symbols. However, to calculate the total payoff related to collaborative maneuvers it will take the following pattern:

$$\langle\langle \mathscr{P}_{\mathtt{env}}, \mathscr{P}_{\mathtt{o}}, \mathscr{P}_{\mathtt{g}} \rangle\rangle \mathtt{R}\{\text{“}win\text{”}\} =?[\mathtt{F}\ goal]$$

Where $goal = (rounds <= k)$. In this case, the reward reflects how often the collaborative maneuvers win the game within a specific round k. In PRISM, as discussed in Section 3, dedicated reward blocks are used to associate costs with specific actions or list of actions using *rewards* “win” $\ldots$ *endrewards* constructs.

5. Quantitative Analysis using PRISM-games

The formal model of the satellite system will be divided into three Players, who will act as PAs, with a non-player module coordinating their actions (named *SatteliteInteruptionandFailure*). The model will subsequently be subjected to rigorous assessment to evaluate its performance and dependability metrics.

Experimental setup. We have encoded properties in rPATL formalism. PRISM-games model checker v3.2.1 [12] (based on PRISM 4.6), performs probabilistic verification. These experiments were conducted on a Ubuntu-I7 system equipped with 32GB RAM. Multiple engines can be selected (refer to documentation [50]) offering performance benefits for specific model structures.

Artifacts. The source code for the experiments described in this section is publicly available on a GitHub repository[49]. The PRISM codes are well-commented to ensure comprehensibility and facilitate the replication of the experiments.

5.1. Players and system model

The environment player, denoted by $\mathscr{P}_{env}$, encapsulates the failure states, including unscheduled interruptions, ST failures, LT failures, EOL failures, O&M activity degradations, and soft errors (scheduled interruptions). The model is presented in Listing 2. The environment player encapsulates ten commands: Lines 6 and 9 trigger scheduled and soft error interruptions in the normal state with probabilities pr_{soft} and pr_{uns}, respectively. Line 12 triggers O&M failures with probability pr_{om}. Lines 15-21 trigger LT, ST, and EOL failures with probabilities pr_{lt}, pr_{st}, and pr_{eol}, respectively. All the interruptions transitioned the satellite system back to normal in line 32 when the staff handled all the failures. The command in line 24 triggers AMSI or UMSI alerts with probabilities pr_{amsi}, and $1-pr_{amsi}$, respectively. If a failure persists with a probability of 0.5, an AMSI or UMSI alert is also triggered in this case. The command in line 29 forces the satellite system into a failure mode until it can be resolved by on-orbit and ground-based personnel.

The On-orbit player model, denoted by $\mathscr{P}_o$, is detailed in Listing 3. The model encapsulates four commands that interpret global behavior from Figure 3. The command in line 7 simulates the staff in standby mode, awaiting a potential failure trigger. The command in line 5 models a software update process, transitioning the satellite from standby to update mode. The command in line 13 checks for a redundant subsystem in the event of an ST failure with a probability of $1 - P_\beta$. The command in line 16 shows the case where the action of checking redundant satellite fails with probability P_β. The command in line 19 shows the case where the update fails with probability P_β. In the worst case, ground control checks the satellite's availability and replaces it with probability P_β (line 22). Upon successful update, the satellite resets to standby (line 25).

The On-ground player model, denoted by $\mathscr{P}_g$, is detailed in Listing 4. The model encapsulates five commands that interpret global on-ground behavior from Figure 3. The command in line 7 simulates the staff in standby mode, awaiting a potential failure trigger. The command in line 10 models a staff member transitioning the satellite from standby status to on-ground spare by modifying the variable $s1$. Building a new satellite is modeled by the command in line 13 with probability pr_{build}. The staff triggers the manufacturing with probability $pr_{manufecture}$. In this case, ground control initiates preparations for launching a new satellite. Once the satellite is built and manufactured, it is prepared for launch (line 19). Finally, when the satellite is ready for launch (line 22), the ground staff returns to standby mode with probability P_n.

The non-player module simulates concurrent on-orbit replacement and ground-based manufacturing processes. The monitor (scheduler) ensures synchronization between these modules, coordinating responses to satellite failures (action failures) with on-orbit and on-ground satellite maneuvers. Other types of synchronization will not be permitted to ensure that unrecorded synchronization occurs during strategy synthesis.

Listing 2: Satellite Environmental Failure Model

```
module Environment
    // s3: System State (0: Normal, 1-10: Various Failure States)
    s3: [0..10] init 0;

    // Scheduled Event: Transition from Normal to Scheduled state
    [Scheduled]    s3=NORMAL    ->  (1-pr_st):(s3'=NORMAL)      +pr_st:(s3'=SCHEDULED) ;

    // Unscheduled Event: Transition from Normal to Unscheduled state
    [Unscheduled]  s3=NORMAL    ->  (1-pr_uns):(s3'=NORMAL)     +pr_uns:(s3'=UNSCHEDULED) ;

    // OMActivities Event: Transition from Normal to OMActivities state
    [OMActivities] s3=NORMAL    ->  (1-pr_om):(s3'=NORMAL)      +pr_om:(s3'=OMACTIVITIES) ;

    // LTFailure Event: Transition from Normal to LTFailure state
    [LTFailure]    s3=NORMAL    ->  (1-pr_lt):(s3'=NORMAL )     +pr_lt:(s3'=LTFAILURE) ;

    // STFailure Event: Transition from Normal to STFailure state
    [STFailure]    s3=NORMAL    ->  (1-pr_st):(s3'=NORMAL )     +pr_st:(s3'=STFAILURE) ;

    // EOLFailure Event: Transition from Normal to EOLFailure state
    [EOLFailure]   s3=NORMAL    ->  (1-pr_eol):(s3'=NORMAL )    +pr_eol:(s3'=EOLFAILURE) ;

    // AMSIAlert Event: Transition from Scheduled to AMSIAlert or UMSIAlert state
    [AMSIAlert]    s3=SCHEDULED ->  0.5:(s3'=PERSISTANTFAILURE )+(0.5*pr_amsi):(s3'=AMSIALERT )
        +(0.5*(1-pr_amsi)):(s3'=UMSIALERT) ;

    [Persist]      s3=PERSISTANTFAILURE -> (pr_amsi):(s3'=AMSIALERT )   +(1-pr_amsi):(s3'=
        UMSIALERT) ;

    // Failure notification
    [Failure]        s3=LTFAILURE | s3=SCHEDULED | s3=UNSCHEDULED | s3=STFAILURE | s3=EOLFAILURE
         | s3=UMSIALERT | s3=AMSIALERT -> true;

    // Reset Event: Transition from any failure state back to Normal state
    [Reset]         true -> (s3'=NORMAL);

endmodule
```

5.2. Properties of the modeled system as game goals

We have identified the need to analyze satellite systems' reliability, availability, and maintainability (RAM) properties. Reliability refers to the satellite's ability to function without failures, considering scheduled and unscheduled interruptions. Maintainability reflects the ease with which the satellite can be repaired, with in-orbit repair being a key aspect. The PRISM-games tool provides support for automated analysis of properties expressed in rPATL. So we express the system properties in natural language and then map them to the rPATL structure, *verifications are conducted over 30 rounds, starting with round 1 and incrementing by one*:

5.2.1. Continuity

Continuity properties that we can analyze using the PRISM-games include:

1. *What is the probability of a continuity loss within 15 years across k attempts*?

$$\langle\langle \mathscr{P}_{\text{env}} \rangle\rangle \texttt{P} =?[\texttt{F}\ (\text{“}\texttt{LossOfContinuty}\text{”}\ \&\ \texttt{rounds} <= \texttt{k})], \texttt{k} = 1:30:1 \qquad \text{(PRO1)}$$

2. *How many satellite continuity losses might eventually be experienced over 15 years, considering k attempts*?

$$\langle\langle \mathscr{P}_{\text{env}} \rangle\rangle \texttt{R}\{\text{“}\texttt{LossOfContinuty}\text{”}\} =?[\texttt{F}\ (\texttt{rounds} <= \texttt{k})], \texttt{k} = 1:30:1 \qquad \text{(PRO2)}$$

Listing 3: OnOrbit Satellite Manoeuvers

```
module OnOrbitManoeuvers

    // State variable declaration with initial value
    s2: [0..10] init 0;  // s2 is an integer variable ranging from 0 to 10, initialized to 0

    // State transition - standby state
    [OnOrbitStandby]  s2=STANDBY -> (s2'=STANDBY);       // The system is in standby state (s2=
        STANDBY) and remains in standby (s2'=STANDBY)

    // State transition - repair on-orbit software update
    [RepairOnOrbitSoftwareUpdate]    s2=STANDBY -> (s2'=UPDATE);       // The system transitions
         from standby (s2=STANDBY) to update state (s2'=UPDATE) for software update

    // Switch to a Redundant Subsystem
    [SwitchToRedundantSubsystem]    s2=STANDBY ->   (1-Pbeta):(s2'=CHECKSATTELITE)+      Pbeta
            :(s2'=STANDBY) ; // The system transitions from the update state to either the
        check satellite state or the standby state based on the probabilities (1-Pbeta) and
        Pbeta

    // State transition - check for redundant satellite (from update state)
    [CheckOnOrbitRedundantSatellite]        s2=STANDBY  ->  (1-Pbeta):(s2'=CHECKSATTELITE)+
                Pbeta  :(s2'=STANDBY) ; // Similar to the previous transition, the system
        transitions to either the check satellite state or the standby state based on
        probabilities

    // State transition - reset to standby
    [ResetOnOrbitManoeuvers]    s2=UPDATE    -> (1-Pbeta):(s2'=CHECKSATTELITE)+         Pbeta
        :(s2'=STANDBY) ;    // Similar to the previous transition, after update the system
        transitions to either the check satellite state or the standby state based on
        probabilities

    // State transition - move/replace redundant satellite (from check state)
    [MoveReplaceRedundantSatellite]         s2=CHECKSATTELITE          -> Pbeta:(s2'=REPLACE)
        +(1-Pbeta):(s2'=ONGROUNDSPARE); // The system transitions from the check satellite state
         to either the replace state or the on-ground spare state based on probabilities

    // State transition - reset to standby
    [ResetOnOrbitManoeuvers]     s2=ONGROUNDSPARE | s2=REPLACE    -> (s2'=STANDBY);  // The
        system can reset to standby from various states (ONGROUNDSPARE, REPLACE, UPDATE) and
        sets s2' (next state) to standby

// End of module definition
endmodule
```

5.2.2. Integrity

Integrity properties that we can analyze using the PRISM-games include:

1. *What is the probability of integrity loss within 15 years that could eventually occur over 15 years, given k attempts?*

$$\langle\langle \mathscr{P}_{\text{env}} \rangle\rangle \texttt{P} =?[\texttt{F}\ (\text{“}\texttt{LossOfIntegrity}\text{”}\ \&\ \texttt{rounds} <= \texttt{k})], \texttt{k} = 1:30:1 \qquad \text{(PRO3)}$$

2. *What is the estimated number of integrity failures that could eventually occur over 15 years, given k attempts?*

$$\langle\langle \mathscr{P}_{\text{env}} \rangle\rangle \texttt{R}\{\text{“}\texttt{LossOfIntegrity}\text{”}\} =?[\texttt{F}\ (\texttt{rounds} <= \texttt{k})], \texttt{k} = 1:30:1 \qquad \text{(PRO4)}$$

5.2.3. Reliability

Reliability properties that we can analyze using the PRISM-games include:

1. *What is the probability that a satellite will eventually need to be replaced within 15 years, considering the variability in slot availability due to potential failures within k attempts?*

Listing 4: On-Ground Satellite Manoeuvers

```
module OnGroundManoeuvers

  // State variable: s1 represents the current state of the satellite on the ground */
  s1 : [0..10] init 0;

  // Transition: Satellite remains in standby state
  [OnGroundStandby] s1=STANDBY -> (s1'=STANDBY);

  // Transition: Satellite moves to spare state
  [OnGroundSpare] s1=STANDBY -> (s1'=ONGROUNDSPARE);

  // Transition: Satellite moves to build a state with probability pr_build
  [CheckOnGroundSatelliteToBuild] s1=ONGROUNDSPARE -> pr_build:(s1'=BUILD) + (1-pr_build):(s1'=
      STANDBY);

  // Transition: Satellite moves to manufacture state with probability pr_manufacture
  [CheckOnGroundSatelliteToManufecture] s1=BUILD -> pr_manufacture:(s1'=MANUFECTURE) + (1-
      pr_manufacture):(s1'=STANDBY);

  // Transition: The satellite is built and ready for launch
  [BuildOnGroundSatellite] s1=MANUFECTURE -> (s1'=LAUNCH);

  // Transition: Satellite resets to standby state with probability Pn
  [ResetOnGroundManoeuvers] s1=LAUNCH -> (1-Pn):(s1'=LAUNCH) + (Pn):(s1'=STANDBY);

endmodule
```

$$\langle\langle \mathscr{P}_{\text{env}}, \mathscr{P}_{\text{o}} \rangle\rangle \text{P} =?[\text{F } (\text{“replace”} \ \& \ \text{rounds} <= \text{k})], \text{k} = 1:30:1 \qquad \text{(PRO5)}$$

2. *What is the probability that a satellite will eventually need to be manufactured within 15 years, whatever the variability in slot availability due to potential failures within k attempts?*

$$\langle\langle \mathscr{P}_{\text{env}}, \mathscr{P}_{\text{g}} \rangle\rangle \text{P} =?[\text{F } (\text{“manufacture”} \ \& \ \text{rounds} <= \text{k})], \text{k} = 1:30:1 \qquad \text{(PRO6)}$$

3. *What is the probability that a satellite will eventually need to be replaced or manufactured within 15 years, considering the variability in slot availability due to potential failures within k attempts?*

$$\langle\langle \mathscr{P}_{\text{env}}, \mathscr{P}_{\text{o}}, \mathscr{P}_{\text{g}} \rangle\rangle \text{P} =?[\text{F } (\text{“ReplaceOrManufacture”} \ \& \ \text{rounds} <= \text{k})], \text{k} = 1:30:1 \qquad \text{(PRO7)}$$

5.2.4. Maintainability

Maintainability properties that we can analyze using the PRISM-games include:

1. *How many satellite replacements might eventually be required over 15 years, given the fluctuations in slot availability across k attempts?*

$$\langle\langle \mathscr{P}_{\text{env}}, \mathscr{P}_{\text{o}} \rangle\rangle \text{R}\{\text{“replace”}\} =?[\text{F } (\text{rounds} <= \text{k})], \text{k} = 1:30:1 \qquad \text{(PRO8)}$$

2. *How many satellite replacements might eventually be required over 15 years, given the fluctuations in slot availability across k attempts involving both staffs ?*

$$\langle\langle \mathscr{P}_{\text{env}}, \mathscr{P}_{\text{o}}, \mathscr{P}_{\text{g}} \rangle\rangle \text{R}\{\text{“replaceOrManufacture”}\} =?[\text{F } (\text{rounds} <= \text{k})], \text{k} = 1:30:1 \qquad \text{(PRO9)}$$

5.2.5. *Availability*

The availability properties that we can analyze using the PRISM-games include the following:

- *What is the expected availability of the satellite system over 15 years, given the variability in slot availability across k attempts*? We use `T` to denote the maximum number of continuity losses at `MAX_ROUNDS` rounds.

$$\langle\langle \mathscr{P}_{\mathtt{env}}, \mathscr{P}_{\mathtt{o}}, \mathscr{P}_{\mathtt{g}} \rangle\rangle \mathtt{R}\{\text{“}\mathtt{ReplaceOrManufacture}\text{”}\} =?[\mathtt{F}\ (\mathtt{rounds} <= \mathtt{k})]/\mathtt{T}, \mathtt{k} = 1:30:1 \quad \text{(PRO10)}$$

First, to address the properties mentioned above, we propose extending the model with a module to track the number of rounds synchronized with module actions as in [34] (see Listing 5). In this case, as the commands are unaffected by the players' choices, they are considered unlabeled with empty action. Consequently, these commands are executed regardless of the actions taken by the players.

Listing 5: Rounds Module [34, 51]

```
1  const k; // number of rounds
2  module rounds // module to count the rounds
3   rounds : [0..k+1];
4   [] rounds<=k -> (rounds'=rounds+1);
5   [] rounds=k+1 -> true;
6  endmodule
```

Second, we define eight PRISM labels to simplify the formulation of the properties presented in Figure 6. Lines 2-4 label states related to on-orbit satellite replacement and ground-based satellite manufacturing. Line 6 labels a state where a decision is made to replace or build a new satellite. Line 8 labels a state where a soft error persists. Line 10 labels a state where all failure types except soft errors have been detected. Line 12 labels a state where an integrity loss occurs due to an unaddressed alerted MSI. Lines 14 and 16 label the winning states for on-orbit and on-ground operations, respectively.

Listing 6: Label Structures in PRISM

```
1  // Label for a successful satellite replacement by orbit staff
2  label "replace" = s2=REPLACE_SUCESS;
3  // Assigning the label "manufacture" to the state where a satellite is being manufactured on the
       ground.
4  label "manufacture" = s1=ON_GROUND_MANUFECTURE_SATTELITE;
5  // This label is assigned to the state where a satellite is successfully manufactured on the
       ground or a replacement satellite is successfully deployed.
6  label "repairOnOrbitOnground" =  (s1=ON_GROUND_MANUFECTURE_SATTELITE) | (s2=REPLACE_SUCESS);
7  // A label "Persistence" is assigned to the state PERSISTENTFAILURE, indicating a persistent
       failure scenario.
8  label "Persistence" =  s3=PERSISTENTFAILURE;
9  // This label is assigned to states indicating various failure scenarios
10 label "Loss_Continuty" = (s3=UNSCHEDULED) | (s3=OMACTIVITIES) | (s3=LTFAILURE) | (s3=STFAILURE)
       | (s3=EOLFAILURE);
11 // This label is assigned to the state where a loss of integrity occurs due to an unalerted MSI.
12 label "Loss_Integrity" = s3=UMSIALERT;
13 // This label assigns the value ON_ORBIT_PLAYER to the win variable, indicating a win for the on
       -orbit player.
14 label "On_Orbit_Player" = win=ON_ORBIT_PLAYER;
15 // Assigns the label "On_Ground_Player" to the state where the on-ground player wins.
16 label "On_Ground_Player" = win=ON_GROUND_PLAYER;
```

Third, we define eight reward structures in Listing 7. The first three reward structures (lines 2-13) assign a reward of 1 to successful satellite replacement and on-ground manufacturing. The fourth reward structure (lines 15-17) assigns a reward of 1 to persistent soft failures. The reward structure (lines 19-25) assigns a reward of 1 to each action labeling satellite failures, except for soft failures. Lines 27-29 assign a reward of

one unit to the state where an unalerted MSI is captured causing a loss of integrity. Finally, the last two reward structures (lines 31-37) assign a reward of 1 to the states where either the on-orbit or the on-ground player wins.

Listing 7: Reward Structures in PRISM

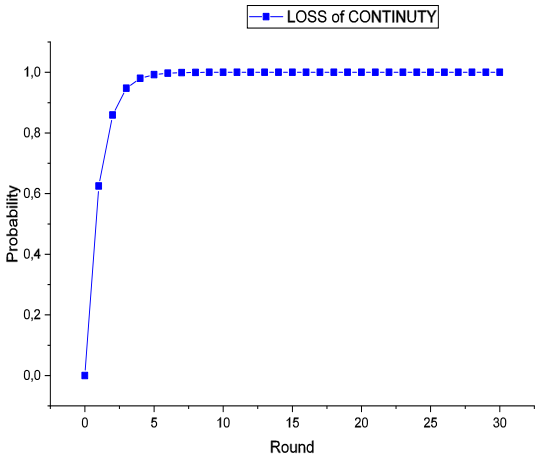
```
1  // A reward of 1 is assigned to the state REPLACE_SUCESS, indicating a successful replacement.
2  rewards "replace"
3      s2=REPLACE_SUCESS : 1;
4  endrewards
5  // A reward indicating that a satellite is being manufactured on the ground.
6  rewards "manufacture"
7      s1=ON_GROUND_MANUFECTURE_SATTELITE : 1;
8  endrewards
9  // A reward of 1 is assigned to both the state REPLACE_SUCESS  and the state
       ON_GROUND_MANUFECTURE_SATTELITE.
10 rewards "repairOnOrbitOnground"
11     s2=REPLACE_SUCESS     : 1;
12     s1=ON_GROUND_MANUFECTURE_SATTELITE : 1;
13 endrewards
14 // A reward of 1 is assigned to the transition labeled "Persist," indicating that a soft failure
       persists in the system.
15 rewards "Persistance"
16     [Persist] true :1 ;
17 endrewards
18 // A reward of 1 is assigned to each of the following events, indicating a loss of continuity:
19 rewards "Loss_Continuty"
20     [Unscheduled]  true :1 ; // - Unscheduled interruptions (soft errors)
21     [OMActivities] true :1 ; // - Operations and Maintenance (O&M) activities
22     [LTFailure]    true :1 ; // - Long-term (LT) failures
23     [STFailure]    true :1 ; // - Short-term (ST) failures
24     [EOLFailure]   true :1 ; // - End-of-life (EOL) failures
25 endrewards
26 // A reward of 1 is assigned to the state UMSIALERT, indicating an unalerted MSI (loss of
       integrity).
27 rewards "Loss_Integrity"
28     s3=UMSIALERT : 1 ;
29 endrewards
30 // A reward of 1 is assigned to the state ON_ORBIT_PLAYER, indicating an intervention by on-
       orbit staff. This can be used to count the number of such interventions.
31 rewards "On_Orbit_Player"
32     win=ON_ORBIT_PLAYER : 1 ;
33 endrewards
34 // A reward of 1 is assigned to the state ON_GROUND_PLAYER, indicating an intervention by ground
       staff.
35 rewards "On_Ground_Player"
36     win=ON_GROUND_PLAYER : 1 ;
37 endrewards
```

5.3. Experiments results analyses

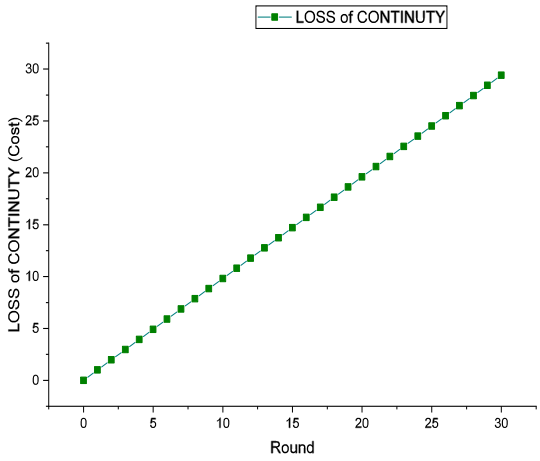
We investigate the performance and dependability of satellite systems, focusing on failures and errors resulting from radiation and software degradation. The results are derived from probabilistic model checking of CSG models, specifically targeting rPATL properties as outlined above.

5.3.1. Continuity

The verification results for PRO1 in Figure 5a demonstrate that satellite operational continuity decreases due to increasing continuity losses as the number of rounds increases. This suggests that independent failures can exacerbate damage to the satellite, especially in scenarios where multiple radiation impacts occur within a 15-year operational period. Additionally, Figure 5b illustrates a linear relationship between the number of rounds and the loss of service continuity, indicating that as the number of rounds increases, so does the probability of service interruptions. The result of CSG verification is a projection of an MDP, as we consider only a single player in properties PRO1 and PRO2.



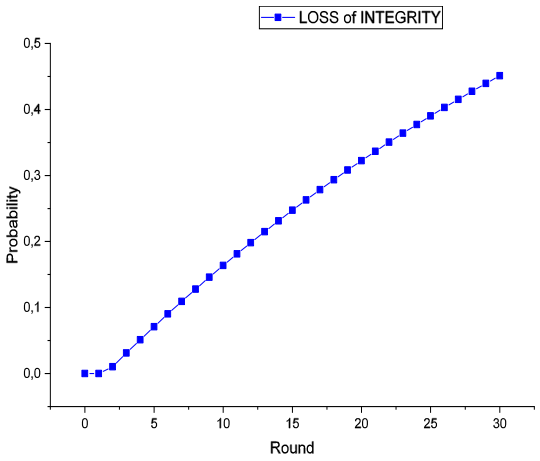
(a) Probability of Service Disruption via PRO1.



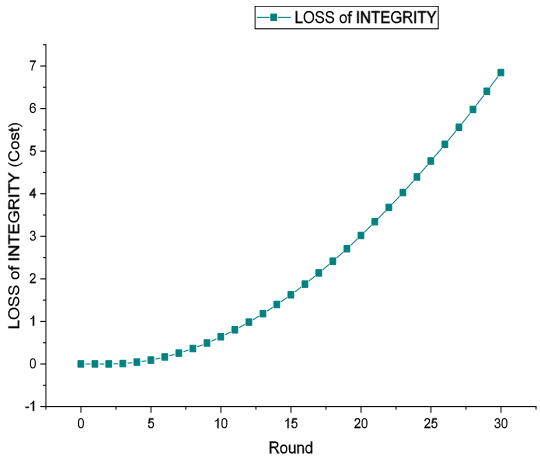
(b) Cost of Service Disruption via PRO2.

Figure 5: Assessment of Continuity Loss Due to Satellite Failure.

5.3.2. Integrity



(a) Probability of Integrity Loss via PRO3.



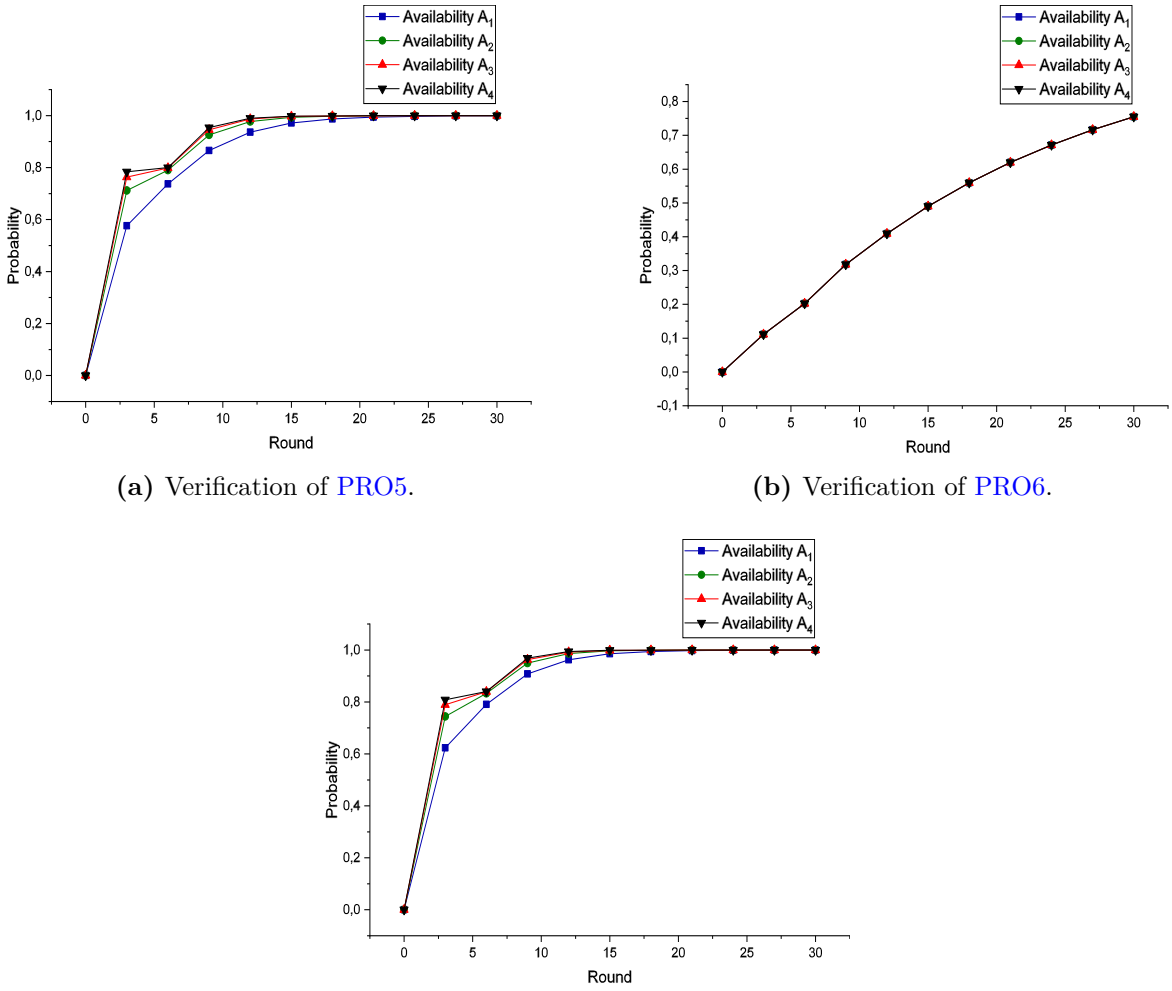
(b) Cost of Integrity Loss via PRO4.

Figure 6: Assessment of Integrity Loss Due to Satellite Failure.

The verification results for PRO3, depicted in Figure 6a, show that the probability of losing integrity increases with the number of rounds, but at a slower rate compared to the results in Figure 5a. This is because the loss of integrity in this case is solely due to soft errors, excluding other types of failures. However, the verification results for PRO4, presented in Figure 6b, indicate that the cost of integrity loss increases exponentially with the number of rounds. As the number of rounds increases, the rate at which integrity is lost accelerates, primarily due to successful UMSI alerts and external factors like radiation and extreme temperatures. While exponential growth can be beneficial in certain contexts (e.g., economic growth), it can also lead to rapid and uncontrolled escalation, such as population growth or climate change.

In satellite systems, this can manifest as a rapid degradation of system performance, potentially exceeding the capabilities of on-ground and on-orbit staff to maintain the system.

5.3.3. Reliability



(a) Verification of PRO5.

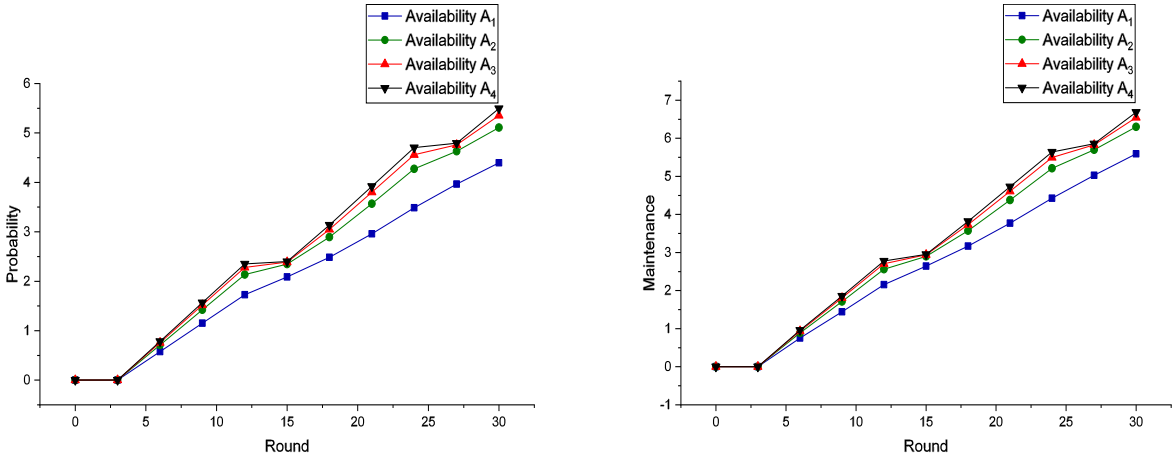
(b) Verification of PRO6.

(c) Verification of PRO7.

Figure 7: Reliability Properties Verification with Satellite Availabilities: A_1, A_2, A_3, A_4.

The verification results for PRO5, depicted in Figure 7a, demonstrate that the reliability of the satellite system, considering various failures and errors, increases with the number of rounds, especially when on-orbit staff are available for satellite replacement. However, even with on-orbit staff, the reliability reaches a plateau around 95% after 10 rounds. In contrast, the verification results for PRO6, shown in Figure 7b, indicate a slower increase in reliability, as the model relies solely on ground-based staff. The reliability does not reach the desired 95% threshold. Finally, the verification results for PRO7, presented in Figure 7c, demonstrate a similar rate of increase in reliability as Figure 1, but the 95% threshold is reached after 9 rounds. This scenario involves both on-orbit and ground-based staff working together to replace or build new satellites. To achieve high initial reliability, it is crucial to ensure the effectiveness of command updates.

5.3.4. *Maintainability*



(a) Verification of PRO8. **(b)** Verification of PRO9.

Figure 8: Maintenance Properties Verification with Satellite Availabilities: A_1, A_2, A_3, A_4.

The verification results for PRO8, depicted in Figure 8a, demonstrate that as the number of rounds increases, the cost of maintenance also increases, particularly when on-orbit staff are involved in satellite updates and replacements. The maintainability of the system increases at a similar rate as the availability of on-orbit satellites. However, in Figure 8b , a slight increase in ground-based satellite manufacturing can significantly impact the overall system availability, especially when on-orbit maintenance is limited. This highlights the importance of a balanced approach to both on-orbit and ground-based maintenance strategies.

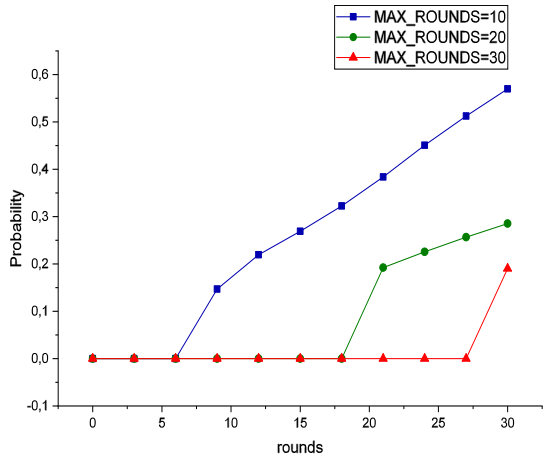
5.3.5. *Availability*

The verification results for PRO10 are illustrated in Figure 9, which vary in the number of rounds and availability scenarios in Figure 9a, Figure 9b,Figure 9c, and Figure 9d. Additionally, we considered independent satellite failures and their impact on continuity loss over 10, 20, and 30 rounds. We observed that higher numbers of rounds with continuity loss lead to lower overall availability. For instance, with 10 rounds of continuity loss, the availability is 57% for A_1, 62% for A_2, 66% for A_3, and 70% for A_4. However, with 30 rounds of continuity loss, availability drops to between 10% and 22%. This decline is attributed to the inability of on-ground and on-orbit staff to effectively address the high level of degradation caused by radiation and other factors, necessitating rapid and multiple interventions to maintain the satellite system.

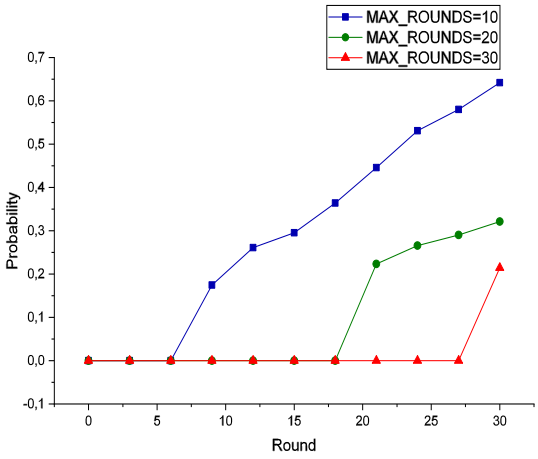
5.4. *Human-in-the-loop operations influences*

Most operational maintenance tasks, such as replacements and manufacturing, are performed sequentially by on-orbit and on-ground operators to ensure the correct functioning of the satellite system. We aim to investigate the prioritization of tasks assigned to both on-ground and on-orbit operators. As the monitor (hardware implementation) can dynamically assign tasks to either staff, the system's overall strategy may evolve over its lifecycle and due to the skills of each staff embers.

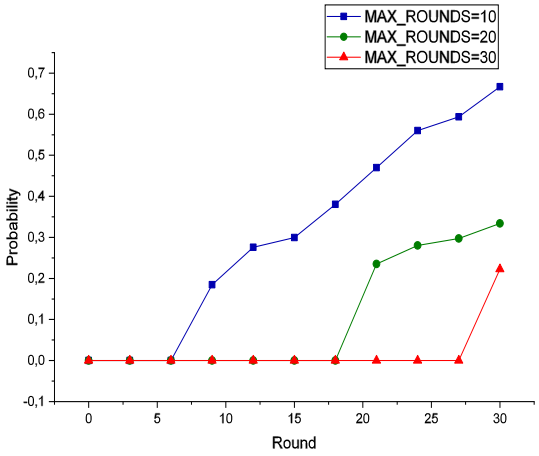
The updated model, detailed in Listing 8, assigns probabilities to on-ground and on-orbit staff for task allocation. In line 4, the probability `OnGroundPriority` is assigned to on-ground staff, while $(1 - \texttt{OnGroundPriority})$ is assigned to on-orbit staff. This probability is determined during model verification. All player actions should be labeled based on the command actions executed when the satellite is in failure mode and both ground and orbit staff are in standby mode. The `win` variable is assigned to the



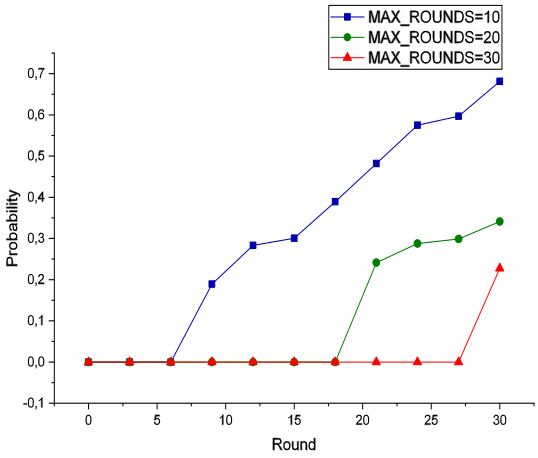
(a) Verification with Availability A_1.



(b) Verification with Availability A_2 .



(c) Verification with Availability A_3.



(d) Verification with Availability A_4.

Figure 9: Impact of Constellation Slot Availability $(A_1, A_2, A_3, A_4.)$ on Satellite System Availability with PRO10.

reference number of the winning staff member (either on-orbit or on-ground). The commands in lines 7-16 schedule the actions of the on-orbit staff (by labeling player actions) and the standby mode of the on-ground staff when a failure is detected. The command in line 19 labels the actions associated with on-orbit failure and ground-based manufacturing and lunch scenarios. The command in line 20 models the situation where only ground-based operations are initiated.

Listing 8: Prioritizing Tasks in Collaborative Space Operations

```
1  module SatteliteMaintenanceScheduling
2      win: [0..MAX_PLAYERS] init 0; // Win refers to the player
3
4      [Failure,OnOrbitStandby,OnGroundStandby] s2=ON_ORBIT_STANDBY & s1=ON_GROUND_STANDBY ->
           OnGroundPriority:(win'=ON_GROUND_PLAYER)+(1-OnGroundPriority):(win'=ON_ORBIT_PLAYER);
5
6      // Manage ST failure: If a short-term failure occurs, the system attempts to switch to a
           redundant subsystem and goes to the standby state.
7      [STFailure,SwitchToRedundantSubsystem,OnGroundStandby] s3=NORMAL & s2=ON_ORBIT_STANDBY & s1=
           ON_GROUND_STANDBY & win=ON_ORBIT_PLAYER-> (win'=ON_ORBIT_PLAYER);
8
9      // Manage EOL failure: If an end-of-life failure occurs, the system checks for a redundant
           satellite and goes to the standby state.
10     [EOLFailure,CheckRedundantSatellite,OnGroundStandby] s3=NORMAL & s2=ON_ORBIT_STANDBY & s1=
           ON_GROUND_STANDBY & win=ON_ORBIT_PLAYER-> (win'=ON_ORBIT_PLAYER);
11
12     // Manage LT failure: If a long-term failure occurs, the system sends software commands and
           goes to the standby state.
13     [LTFailure, SendSoftwareCommands,OnGroundStandby] s3=NORMAL & s2=ON_ORBIT_STANDBY & s1=
           ON_GROUND_STANDBY & win=ON_ORBIT_PLAYER -> (win'=ON_ORBIT_PLAYER);
14
15     // Successful replacement of a failed satellite: If a replacement is successful, the system
           goes to the standby state.
16     [Failure, ReplaceSucess,OnGroundStandby]  s2=REPLACE_SUCESS & s1=ON_GROUND_STANDBY & win=
           ON_ORBIT_PLAYER -> (win'=ON_ORBIT_PLAYER);
17
18     // Successful launch of a new satellite and system reset: If a new satellite is launched
           successfully, the system goes to the standby state.
19     [Failure, FailureOnOrbit,LaunchSucess]  s1=LAUNCH_SUCESS & win=ON_GROUND_PLAYER-> (win'=
           ON_GROUND_PLAYER);
20     [Reset, OnOrbitStandby,LaunchSucess] s2=ON_ORBIT_STANDBY & s1=LAUNCH_SUCESS & win=
           ON_GROUND_PLAYER-> (win'=ON_GROUND_PLAYER);
21
22 endmodule
```

We verify the updated model against the following properties while varying the probability value of `OnGroundPriority`:

$$\langle\langle \mathscr{P}_{\text{env}}, \mathscr{P}_{\text{o}} \rangle\rangle \text{R}\{\text{“replace”}\} =?[\text{F (rounds <= MAX\_ROUNDS)}], \text{OnGroundPriority} = 0:1:0.1 \quad (\textit{Replace On Orbit})$$

$$\langle\langle \mathscr{P}_{\text{env}}, \mathscr{P}_{\text{g}} \rangle\rangle \text{R}\{\text{“manufecture”}\} =?[\text{F (rounds <= MAX\_ROUNDS)}], \text{OnGroundPriority} = 0:1:0.1 \quad (\textit{Manufacture new Satellite})$$

The verification results for *Replace On Orbit* and *Manufacture new Satellite*, illustrated in Figure 10, demonstrate that the frequency of satellite replacements varies based on the prioritization of on-ground and on-orbit staff. Assigning higher priority to on-ground operations can lead to decreased replacement rates, but it may also increase costs associated with manufacturing new satellites. Conversely, prioritizing on-orbit operations can reduce manufacturing costs but may increase the risk of satellite failures. A balanced approach, such as assigning equal priority to both on-ground and on-orbit operations, can help maintain a cost-effective and reliable satellite system. However, the optimal strategy will depend on the specific requirements such as the cost of investment and constraints of the satellite system.

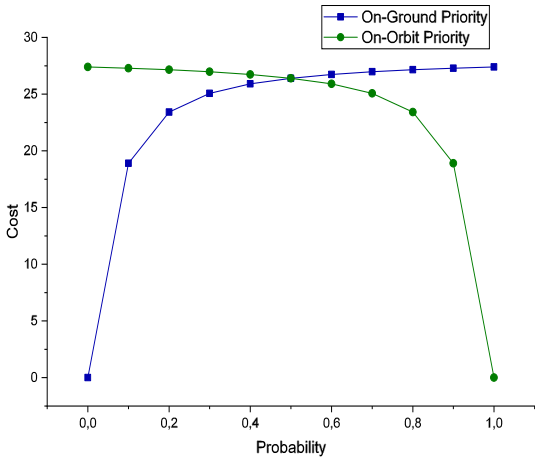


Figure 10: Maintenance Assessment (*Replace On Orbit* and *Manufacture new Satellite*) of Satellite Systems with Varying the On-Ground Priority at maximum rounds (MAX_-ROUNDS=30)

5.5. Threats to validity

Our model focuses on the reliability aspects of the satellite constellation, including RAM (Reliability, Availability, Maintainability) parameters, as well as potential losses of integrity and continuity. However, other considerations were not addressed:

- While the standard GPS SPS specification involves complex calculations for satellite positioning, these computations are not explicitly modeled in PRISM due to language limitations.
- The model does not account for communication disruptions between ground stations and satellites, which can be influenced by factors like solar radiation.
- Our current approach does not explicitly model the complex interactions between on-orbit and ground staff, including the impact of task prioritization on maintenance strategies. The monitor, however, handles task assignments to either on-orbit or ground staff based on predefined priorities.

5.6. Discussion

Strategic maintenance can enhance the quality and reliability of a satellite system by optimizing failure repair strategies. Considering multiple failure modes, including EOL, ST, and LT failures, derived from simulations, can help approximate the availability of satellite slots in orbit.

The model is parametric, allowing for customization of PRISM-games models through the use of parameters. The results closely align with the specifications of the GPS SPS [4], and the model achieves accurate learning results after approximately 9 rounds, as demonstrated by experiments with 95% probability level. However, a key difference lies in the model formalism. Previous implementations relied on CTMC [5, 7] and MDPs [6], whereas this work leverages the CSG formalism to include the human-in-the-loop. To our knowledge, this is the first implementation using CSGs for Performance and RAM analysis in satellite systems with collaborative maintenance (based on the PRISM library [33] as an extended version of SEAA paper).

In MDPs, there is a single decision-maker (i.e., agent) who makes sequential decisions, with the current state and action determining the next state. In contrast, CSGs involve multiple decision-makers (i.e., multiple agents) who make decisions simultaneously, leading to strategic interactions. The transition between states in a CSG is probabilistic, but it is influenced by the joint actions of all players. Our model provides a more realistic representation of satellite operations by considering the concurrent decision-making of human operators, who may choose to maintain an existing satellite or build a new one, depending on the specific circumstances (priority) and constellation availability. This level of detail is not captured in previous models.

The approach involving human maneuvers within CSGs could be integrated into robust model-based design frameworks like Eclipse Papyrus[52] and Business Process Model and Notation (BPMN) [53]. By leveraging model-to-model transformation techniques as in [54, 55, 26], the artifacts can be seamlessly integrated into such frameworks, enhancing the overall efficiency of the modeling process.

6. Conclusion

This paper showcases the application of PRISM-games, a probabilistic model checker for stochastic games, to model collaborative maintenance scenarios involving on-orbit and ground staff. We evaluated the effectiveness of collaborative operations by conducting a RAM (Reliability, Availability, Maintainability) analysis and assessing performance metrics such as loss of continuity and integrity. This approach incorporated additional failure modes that were not previously considered. In addition, previous research has primarily focused on improving model availability, overlooking the crucial role of human intervention and response to failures.

Our future work will focus on analyzing different satellite systems, such as the European Sattelite Systems Galileo, despite the potential challenges posed by limited documentation. This will ultimately allow us to investigate the performance of both European and American space industries. We will also conduct an in-depth investigation into the performance of satellite systems, focusing on real-time positioning, which necessitates complex computations. Utilizing OMNeT++, we will simulate and monitor system behavior to extract patterns and gain valuable insights.

Acknowledgements

The research leading to the presented results was conducted within the research profile of **H**uman-**C**entric **C**ollaborative **A**rchitectural **D**ecision-**M**aking for **S**ecure **S**ystem **D**esign (HERMES-Design[1]) supported by Institut de Cybersécurité d'Occitanie (ICO).

References

[1] Satellite map - track satellites around the world, https://satellitemap.space/?constellation=GPS, Accessed on 2024-10-17.

[2] Amazon, What is amazon project kuiper?, https://www.aboutamazon.com/news/innovation-at-amazon/what-is-amazon-project-kuiper, 2023.

[3] D. Farias, B. Nogueira, I. F. Júnior, E. Andrade, A modeling-based approach for dependability analysis of a constellation of satellites, Software and Systems Modeling (2024).

[4] U. Government, GPS performance, https://www.gps.gov/systems/gps/performance/, [Accessed: January 2025].

[5] K. A. Hoque, O. A. Mohamed, Y. Savaria, Towards an accurate reliability, availability and maintainability analysis approach for satellite systems based on probabilistic model checking, in: 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015, pp. 1635–1640. doi:10.7873/DATE.2015.0817.

[6] Y. Lu, Z. Peng, A. Miller, T. Zhao, C. W. Johnson, How reliable is satellite navigation for aviation? checking availability properties with probabilistic verification, Reliab. Eng. Syst. Saf. 144 (2015) 95–116. URL: https://doi.org/10.1016/j.ress.2015.07.020. doi:10.1016/J.RESS.2015.07.020.

[7] Z. Peng, Y. Lu, A. Miller, C. W. Johnson, T. Zhao, A probabilistic model checking approach to analysing reliability, availability, and maintainability of a single satellite system, in: D. Al-Dabass, A. Orsoni, Z. Xie (Eds.), Seventh UKSim/AMSS European Modelling Symposium, EMS 2013, 20-22 November, 2013, Manchester UK, IEEE, 2013, pp. 611–616. URL: https://doi.org/10.1109/EMS.2013.102. doi:10.1109/EMS.2013.102.

[8] V. Konin, O. Pogurelskiy, A. Turovska, Simulation and estimation parameters of low orbit satellite navigation system, in: 2021 IEEE International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo), 2021, pp. 160–163. doi:10.1109/UkrMiCo52950.2021.9716604.

[9] L. Patino-Studencka, G. Rohmer, J. Thielecke, Approach for detection and identification of multiple faults in satellite navigation, in: IEEE/ION Position, Location and Navigation Symposium, 2010, pp. 221–226. doi:10.1109/PLANS.2010.5507204.

[10] M. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: Verification of probabilistic real-time systems, in: Proc. 23rd International Conference on Computer Aided Verification (CAV'11), volume 6806 of *LNCS*, Springer, 2011, pp. 585–591.

[1] HERMES-Design: https://hermes-design.github.io/

[11] M. Kwiatkowska, G. Norman, D. Parker, G. Santos, Equilibria-based probabilistic model checking for concurrent stochastic games, in: Formal Methods – The Next 30 Years, Springer International Publishing, Cham, 2019, pp. 298–315.
[12] M. Kwiatkowska, G. Norman, D. Parker, G. Santos, Prism-games 3.0: Stochastic game verification with concurrency, equilibria and time, in: Computer Aided Verification, Springer International Publishing, Cham, 2020, pp. 475–487.
[13] M. Kwiatkowska, G. Norman, D. Parker, G. Santos, Automatic verification of concurrent stochastic systems, Formal Methods in System Design 58 (2021) 188–250. doi:10.1007/s10703-020-00356-y.
[14] A. Baouya, B. Hamid, O. A. Mohamed, S. Bensalem, Model-based reliability, availability, and maintainability analysis for satellite systems with collaborative maneuvers via stochastic games, in: The 50th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), 2024.
[15] PRISM - Bibliography - PRISM model checker, https://www.prismmodelchecker.org/bib.php, [Accessed: July 10, 2024].
[16] R. Palin, D. Ward, I. Habli, R. Rivett, Iso 26262 safety cases: Compliance and assurance, in: 6th IET International Conference on System Safety 2011, 2011, pp. 1–6. doi:10.1049/cp.2011.0251.
[17] J. Khan, Iso 26262 system level functional safety validation for battery management systems in automobiles, in: 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), 2017, pp. 1–5. doi:10.1109/IPACT.2017.8245081.
[18] A. Nouri, J. Warmuth, IEC 61508 and ISO 26262 – a comparison study, in: 2021 5th International Conference on System Reliability and Safety (ICSRS), 2021, pp. 138–142. doi:10.1109/ICSRS53853.2021.9660661.
[19] European Union Agency for Railways (ERA), Etcs hazard log, 2022. URL: https://www.era.europa.eu/system/files/2022-10/ETCS%20Hazard%20Log.pdf?t=1729606726, accessed on October 22, 2024.
[20] D. Knutsen, N. O. Olsson, J. Fu, Ertms/etcs level 3: Development, assumptions, and what it means for the future, Journal of Intelligent and Connected Vehicles 6 (2023) 34–45. doi:10.26599/JICV.2023.9210003.
[21] A. Piccolo, V. Galdi, F. Senesi, R. Malangone, Use of formal languages to represent the ertms/etcs system requirements specifications, in: 2015 International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles (ESARS), 2015, pp. 1–5. doi:10.1109/ESARS.2015.7101503.
[22] A. E. Amraoui, K. Mesghouni, Colored petri net model for discrete system communication management on the european rail traffic management system (ertms) level 2, in: 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, 2014, pp. 248–253. doi:10.1109/UKSim.2014.110.
[23] S. Qiu, M. Sallak, W. Schön, Z. Cherfi-Boulanger, Modeling of ertms level 2 as an sos and evaluation of its dependability parameters using statecharts, IEEE Systems Journal 8 (2014) 1169–1181. doi:10.1109/JSYST.2013.2297751.
[24] RTCA, Software Considerations in Airborne Systems and Equipment, 2012. DO-178C.
[25] F. Couadau, N. Dervaux, C. Fayollas, J.-F. Thuong, J. Lyons, Automated testing of arinc 661 cockpit display systems: Factors to accelerate do-178c certification, in: 2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC), 2023, pp. 1–7. doi:10.1109/DASC58513.2023.10311110.
[26] A. Baouya, S. Chehida, S. Bensalem, L. Gürgen, R. Nicholson, M. Cantero, M. Diaz-Nava, E. Ferrera, Deploying warehouse robots with confidence: the brain-iot framework's functional assurance, J. Supercomput. 80 (2024) 1206–1237. URL: https://doi.org/10.1007/s11227-023-05483-x. doi:10.1007/S11227-023-05483-X.
[27] B. L. Mediouni, I. Dragomir, A. Nouri, S. Bensalem, Model-based design of resilient systems using quantitative risk assessment, Innov. Syst. Softw. Eng. 20 (2024) 3–16. URL: https://doi.org/10.1007/s11334-023-00527-0. doi:10.1007/S11334-023-00527-0.
[28] J. Baxter, G. Carvalho, A. Cavalcanti, F. R. Júnior, Roboworld: Verification of robotic systems with environment in the loop, Form. Asp. Comput. 35 (2023). URL: https://doi.org/10.1145/3625563. doi:10.1145/3625563.
[29] A. Cavalcanti, R. M. Hierons, Challenges in testing of cyclic systems, in: 2023 27th International Conference on Engineering of Complex Computer Systems (ICECCS), 2023, pp. 1–6. doi:10.1109/ICECCS59891.2023.00010.
[30] Q. Rouland, B. Hamid, J. Jaskolka, Formal specification and verification of reusable communication models for distributed systems architecture, Future Generation Computer Systems 108 (2020) 178–197.
[31] S. Kanav, V. Aravantinos, Modular transformation from af3 to nuxmv, in: ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, 2017.
[32] PRISM, PRISM model checker publications, 2024. URL: https://www.prismmodelchecker.org/publ-lists.php, PRISM Model Checker Website.
[33] P. M. Checker, PRISM-games - publications, https://www.prismmodelchecker.org/games/publ.php, [Accessed: January 10, 2025].
[34] M. Kwiatkowska, G. Norman, D. Parker, G. Santos, Equilibria-based probabilistic model checking for concurrent stochastic games, in: Proc. 23rd International Symposium on Formal Methods (FM'19), volume 11800 of *LNCS*, Springer, 2019, pp. 298–315.
[35] M. Kwiatkowska, G. Norman, D. Parker, G. Santos, Correlated equilibria and fairness in concurrent stochastic games, in: Proc. 28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'22), volume 13244 of *LNCS*, Springer, 2022, p. 60–78.
[36] M. Kwiatkowska, G. Norman, D. Parker, G. Santos, Prism-games 3.0: Stochastic game verification with concurrency, equilibria and time, in: S. K. Lahiri, C. Wang (Eds.), Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II, volume 12225 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 475–487. URL: https://doi.org/10.1007/978-3-030-53291-8_25. doi:10.1007/978-3-030-53291-8_25.
[37] A. Baouya, B. Hamid, L. Gürgen, S. Bensalem, Rigorous security analysis of rabbitmq broker with concurrent stochastic games, Internet of Things 26 (2024) 101161. URL: https://www.sciencedirect.com/science/article/pii/S2542660524001021. doi:https://doi.org/10.1016/j.iot.2024.101161.
[38] K. Ray, Adaptive service placement for multi-access edge computing: A formal methods approach, in: 2023 IEEE

International Conference on Web Services (ICWS), 2023, pp. 14–20. doi:10.1109/ICWS60048.2023.00014.
[39] K. Ray, A. Banerjee, Prioritized fault recovery strategies for multi-access edge computing using probabilistic model checking, IEEE Transactions on Dependable and Secure Computing 20 (2023) 797–812. doi:10.1109/TDSC.2022.3143877.
[40] A. Wells, Synthesis for Stochastic Robotic Systems, Ph.d. thesis, Rice University, 2021.
[41] J. Cámara, D. Garlan, B. Schmerl, A. Pandey, Optimal planning for architecture-based self-adaptation via model checking of stochastic games, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 428–435. URL: https://doi.org/10.1145/2695664.2695680. doi:10.1145/2695664.2695680.
[42] J. Cámara, G. Moreno, D. Garlan, Reasoning about human participation in self-adaptive systems, in: 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2015, pp. 146–156. doi:10.1109/SEAMS.2015.14.
[43] Z. Peng, Y. Lu, A. Miller, C. W. Johnson, T. Zhao, Risk assessment of railway transportation systems using timed fault trees, Qual. Reliab. Eng. Int. 32 (2016) 181–194. URL: https://doi.org/10.1002/qre.1738.
[44] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, Automated verification techniques for probabilistic systems, in: Formal Methods for Eternal Networked Software Systems (SFM'11), volume 6659 of *LNCS*, Springer, 2011, pp. 53–113.
[45] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, A. Simaitis, Automatic verification of competitive stochastic systems, in: Tools and Algorithms for the Construction and Analysis of Systems, volume 7214, Springer Berlin Heidelberg, ????, pp. 315–330.
[46] C. Baier, J.-P. Katoen, Principles of model checking, The MIT Press, 2008. OCLC: ocn171152628.
[47] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (2002) 672–713.
[48] H. Hansson, B. Jonsson, A logic for reasoning about time and reliability 6 (1994) 512–535. URL: https://dl.acm.org/doi/10.1007/BF01211866. doi:10.1007/BF01211866.
[49] Abdelhakim Baouya, Paper Artefacts Sources, https://hermes-design.github.io/jss2025.html, 2024.
[50] PRISM Development Team (eds.), PRISM Manual, https://www.prismmodelchecker.org/manual/ConfiguringPRISM/ComputationEngines, Accessed: July 10, 2024.
[51] PRISM Model Checker, Public good game case study, https://www.prismmodelchecker.org/casestudies/public_good_game.php, [Accessed: January, 2025]. [Accessed: January, 2025].
[52] E. Foundation, Eclipse papyrus, Accessed July 20, 2024. URL: https://www.eclipse.org/papyrus/.
[53] O. M. G. (OMG), Business process model and notation (bpmn) standard, 2024. URL: https://www.bpmn.org/, accessed November 12, 2024.
[54] A. Baouya, B. Hamid, L. Gürgen, S. Bensalem, Formal modeling and analysis of tampering attacks and energy consumption effects on edge servers using concurrent stochastic games, Soft Computing (2025). In Press.
[55] A. Baouya, D. Bennouar, O. A. Mohamed, S. Ouchani, A quantitative verification framework of SysML activity diagrams under time constraints 42 (2015) 7493–7510. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417415003851. doi:10.1016/j.eswa.2015.05.049.