2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) | 979-8-3503-8026-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/SEAA64295.2024.00014

Model-Based Reliability, Availability, and Maintainability Analysis for Satellite Systems with Collaborative Maneuvers via Stochastic Games

Abdelhakim Baouya
IRIT, Université de Toulouse, CNRS, UT2
118 Route de Narbonne, 31062 Toulouse Cedex 9, France
abdelhakim.baouya@irit.fr

Brahim Hamid
IRIT, Université de Toulouse, CNRS, UT2
118 Route de Narbonne, 31062 Toulouse Cedex 9, France
brahim.hamid@irit.fr

Otmane Ait Mohamed
Department of Electrical and Computer Engineering (ECE)
Concordia University, Montréal, Canada
otmane.aitmohamed@concordia.ca

Saddek Bensalem
University Grenoble Alpes, VERIMAG, Grenoble, France
Grenoble, France
saddek.bensalem@univ-grenoble-alpes.fr

***Abstract*—Space-based navigation systems rely on satellites to operate in orbit and have lifetimes of 10 years or more. Engineers employ Reliability, Availability, and Maintainability (RAM) analysis during the design phase to maximize a satellite's mean time between failures (MTBF). These design parameters help to optimize maintenance plans, enhance overall reliability, and extend the satellite's lifespan. The paper presents a novel approach using concurrent stochastic games (CSG) to model a single satellite with logical and formal specifications of RAM properties in rPATL. We leverage the PRISM-games model checker for quantitative analysis while considering collaborative behaviors between involved players in orbit and on the ground. This CSG-based approach offers a rich design space where actors considered as players involved in satellite maintenance can collaborate and learn optimal strategies.**

***Index Terms*—Navigation Satellite Systems, Reliability, Availability, Maintainability, Concurrent Stochastic Games**

I. Introduction

Satellite systems have become a crucial part of our daily lives. The demand for reliable communication anywhere on Earth has driven innovation in the space industry, fostering competition among new space entrepreneurs like SpaceX and Amazon. These satellite constellations serve a wide range of purposes, including remote sensing for applications like the Internet of Things (IoT) and weather forecasting, as well as supporting critical military operations. Therefore, Reliability, Availability, and Maintainability (RAM) are paramount considerations during the design phase. A focus on RAM ensures systems are dependable and maintainable, minimizing the costs and complexities associated with potential repairs.

Formal verification [1] is a powerful technique for ensuring the correctness and reliability of complex systems. It utilizes various formalisms, each suited to specific use cases. Some common formalisms include Markov Decision Processes (MDPs), Continuous-Time Markov Chains (CTMCs), and Concurrent Stochastic Games (CSGs). Stochastic games verification [2] 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) [2], [3] 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[4], their adoption for RAM analysis has not been investigated.

This paper demonstrates how to accurately model satellite systems and verify their Reliability, Availability, and Maintainability (RAM) properties using 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 [5]. 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 [6], [7], [8]. This allows us to model multiple players involved in the system's operation, including the environment (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 as they comprehensively represent failure management and collaborative maintenance.

2376-9521/24/$31.00 ©2024 IEEE
DOI 10.1109/SEAA64295.2024.00014

Outline: The remainder of this paper is structured as follows. Section II reviews related work. Section III provides the necessary background on Concurrent Stochastic Games (CSGs) and the PRISM-games language. Section IV 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 V. Finally, Section VI concludes the paper and suggests directions for future research.

II. Related work

Concurrent Stochastic Games (CSGs) [3], [4], [9], [10] have been implemented in various scenarios [11], as evidenced by the literature review in the PRISM library [12]. This research leverages a variation of the stochastic game model presented in [13], [14] to identify optimal adaptation strategies through collaborative human maneuvers. Notably, the work in [14] incorporates the human factor as a state of availability (not as a player) within the model. In contrast, the research presented in [15], [16], focuses on Multi-access Edge Computing (MEC) and proposes a service placement policy that utilizes both static (prioritized placement) and dynamic (runtime adjustments) strategies to optimize latency, resource usage, and energy consumption.

Several relevant research papers have investigated the maintenance of satellite systems [6], [7], [8]. In [8], the authors 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. The authors in [6] build upon the model presented in [8] by incorporating Erlang distributions instead of the exponential distributions supported by CTMCs. This change leads to more accurate results when comparing qualitative findings. Finally, authors in [7] 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 [8], the authors in [17] 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.

III. Background on Concurrent Stochastic Games

Concurrent stochastic multi-player games (CSGs) [2], [3] is an extension of Probabilistic Automata (PA) PA [18] where the formalism is based on the idea that players make choices concurrently in each state and then transition simultaneously. In CSGs, each player controls one or more modules, and the actions that label commands within a player's modules must only be used by that specific player.

To express the coalition game, we rely on PRISM [1]. The PRISM model is composed of a set of modules that can synchronize. Each module is characterized by a set of variables and commands (or transitions). The valuations of these variables represent the state of the module. A set of commands is used to describe the behavior of each module. 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 [2], [3]. 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 in the temporal logic rPATL [19] (short for reward Probabilistic Alternating Temporal Logic). The property grammar is based on CTL [20], extended with the coalition operator $\langle\langle \mathtt{C} \rangle\rangle$ of ATL [21] and the probabilistic operator $\mathtt{P}$ of PCTL [22]. For instance, the following property expressed in natural language: *Players 1 and 2 have a strategy to ensure that the probability of system failure occurring within 100 rounds is less than 0.001, regardless of the strategies of attackers* is expressed in rPATL as:

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

Here, `fail` is the label that refers to the system failure states. Regarding the reward structure, the property expressed in natural language: *What is the reward r within 100 rounds to reach* `fail` *for both Players 1 and 2 for a selected strategy?* is expressed in rPATL as:

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

Example 1: Consider the CSG shown in Figure 1 in [11], corresponding to two players repeatedly performing scheduled read and write operations with probability $p = 0.5$. The orchestrator uses actions to label its transitions, where $A = \{(r_1 r_2), (w_1 w_2), (w_2 r_1), (r_2 w_2), (reset_1 reset_2)\}$. The CSG starts in state s_0, and states s_1, s_2, and s_3 are labeled with atomic propositions corresponding to a player winning. Each player is involved in writing and reading operations. Considering the modeled system in Figure 1, when Player 1 initiates the game and emerges victorious by writing, the property is expressed as:

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

The model and properties associated with the example are available in [11]. A dedicated non-player module orchestrates

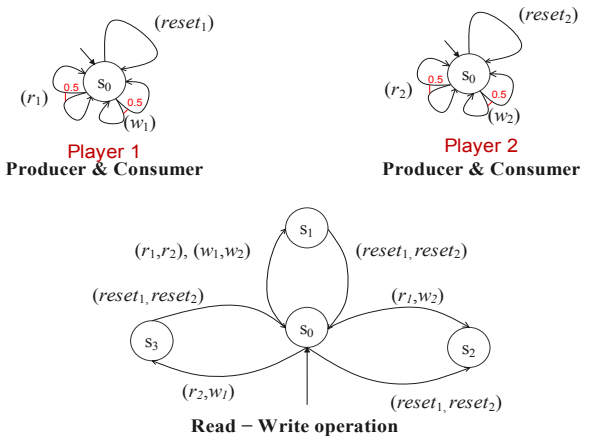

Figure 1: Read and Write Game Model in CSG [11].

read and write operations in the PRISM code shown in Listing III.1. All commands are labeled with at least two ports, 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 first commands in lines 5-6 represent unscheduled writing (i.e., reading) operations. As these operations are executed, a reset command is introduced in line 8 to indicate an idle state. Subsequently, the commands depicted in lines 9-10 enforce an order between writing and reading operations.

Listing III.1: PRISM Code for Read/Write of Figure 1

```
1  module Orchestrator
2   win : [0..2] init 0;
3   s : [0..2] init 0;
4
5   [w1, w2] s=0 -> (s'=1) & (win'=0);
6   [r1, r2] s=0 -> (s'=1) & (win'=0);
7
8   [reset1, reset2] s=1 | s=2 | s=3 -> (s'=0) & (win'=0);
9   [r1, w2] s=0 -> (s'=2) & (win'=2);
10  [w1, r2] s=0 -> (s'=3) & (win'=1);
11 endmodule
```

IV. 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 creation of state-based models, such as the single satellite system shown in Figure 2.

A. The system model

This paper leverages a previously established satellite model described in [6], [7], [8]. 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. 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, however, necessitate maneuvers in orbit or on the ground. Upon satellite failure, ground and orbit control evaluate the best course of action. In some cases, the issue might be resolved remotely by sending software commands to the satellite. If the problem persists, 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 $t_\alpha = 4320h$. If a failure occurs, there is an $p_\beta = 80\%$ chance of resolving it on orbit by replacing the faulty satellite with a redundant one. 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.

B. On orbit and ground support capabilities

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

- *Environment*: Triggers scheduled, unscheduled, and failure events (represented by the environment player: $\mathscr{P}_{env}$) shown in red in Figure 2. 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 2. 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 2). Their actions are labeled as ω.

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 1: 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,

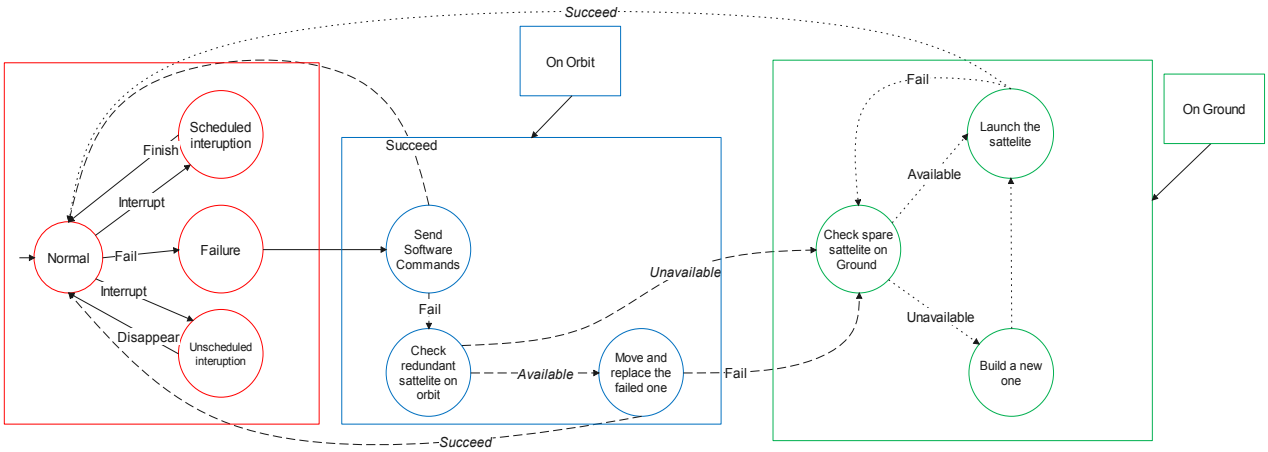

Figure 2: Satellite Maintenance Process Model [6], [7]

- $S = S_{env} \times S_o \times S_g$ is a set of states, 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 $p_i \in \mathbb{R}$.

C. Measure the efficacy of collaborative maneuvers

Measuring the efficacy of collaborative maneuvers consists of synthesizing a strategy for 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 $\langle\langle \mathscr{P}_{\mathtt{env}}, \mathscr{P}_{\mathtt{o}}, \mathscr{P}_{\mathtt{g}} \rangle\rangle \mathtt{P} =?[\mathtt{F}\ goal]$ where $goal = (\text{“}win\text{”} \& \ rounds <= k)$ to quantitatively evaluate the efficacy of collaborative maneuvers up to the round k. However, to calculate the reward or cost related to collaborative maneuvers it will take the following pattern: $\langle\langle \mathscr{P}_{\mathtt{env}}, \mathscr{P}_{\mathtt{o}}, \mathscr{P}_{\mathtt{g}} \rangle\rangle \mathtt{R}\{\text{“win”}\} =?[F\ goal]$ where $goal = (rounds <= k)$. In this case, the reward reflects the number of times the collaborative maneuvers wins the game within a specific round k.

V. Quantitative Analysis using PRISM-games

A. Players and system model

The formal model of the satellite system will be divided into three Markov Decision Process (MDP) player models scheduled using a non-player module. The environment player, denoted by $\mathscr{P}_{env}$, encapsulates the failure states, including both scheduled maintenance and unscheduled interruptions. The model is presented in Listing V.1. The environment player encapsulates five commands: Lines 4 and 5 trigger an interruption in the normal state with probability $1 - e^{mtbf/t_\alpha}$ (where MTBF is the mean time between failures and t_α is a temporary unavailability). Line 6 triggers a failure according to the satellite's reliability ($1 - r$, where r represents reliability). Both scheduled and unscheduled interruptions transition the satellite system back to its normal state. However, in case of failure, the reset is synchronized with the completion (success or termination) of maneuvers by ground or on-orbit staff.

The On-orbit player model, denoted by $\mathscr{P}_o$, is detailed in Listing V.2. The model encapsulates four commands that interpret global behavior from Figure 2. The first command (line 5) models a software update that transitions the satellite from standby status to update by modifying the variable $s2$. A successful update is modeled by the command in line 8 with probability $e^{mtbf/mttr}$ (where MTTR represents the mean time to repair). In the worst case, ground control checks the satellite's availability and replaces it with probability P_β (line 14). Upon successful update or failure, the satellite resets to standby (line 13).

$$\frac{[\![\mathscr{P}_{env}]\!] = s_i \xrightarrow{\alpha}_{p_1} s_i' \bigwedge_{j=0}^{|A_o|} [\![\mathscr{P}_o]\!] = s_j \xrightarrow{\beta_j}_{p_2} s_j' \wedge [\![\mathscr{P}_g]\!] = s_k \xrightarrow{\omega}_{p_3} s_k' \wedge [\![\mathscr{P}_{\mathscr{R}}]\!] = s_m \xrightarrow{\alpha,\beta,\omega} s_m'}{\langle s_i, \ldots, s_j, \ldots s_k, \ldots s_m, \theta\rangle \xrightarrow{\alpha,\beta,\omega}_{p_1 \cdot p_2 \cdot p_3} \langle s_i', \ldots, s_j', \ldots, s_k', \ldots, s_m', \theta'\rangle} \quad (\textit{On-Orbit})$$

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

$$\frac{[\![\mathscr{P}_{env}]\!] = s_i \xrightarrow{\alpha}_{p_1} s_i' \wedge [\![\mathscr{P}_o]\!] = s_j \xrightarrow{\beta}_{p_2} s_j' \bigwedge_{k=0}^{|A_g|} [\![\mathscr{P}_g]\!] = s_k \xrightarrow{\omega_k}_{p_3} s_k' \wedge [\![\mathscr{P}_{\mathscr{R}}]\!] = s_m \xrightarrow{\alpha,\beta,\omega} s_m'}{\langle s_i, \ldots, s_j, \ldots s_k, \ldots s_m, \theta\rangle \xrightarrow{\alpha,\beta,\omega}_{p_1 \cdot p_2 \cdot p_3} \langle s_i', \ldots, s_j', \ldots, s_k', \ldots, s_m', \theta'\rangle} \quad (\textit{On-Ground})$$

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

Figure 3: Operational Semantics Rules of the CSG Game Model.

Listing V.1: Satellite Environmental Failure Model

```
1  module FailureMode//z is the satellite's lifetime
2  s3: [0..3] init 0;//0 NORMAL, 1 FAILURE, 2 SCHEDULED, 3
       UNSCHEDULED
3  //interruptions and failures commands
4  [Scheduled]   s3=NORMAL ->  pow(exp,-mod(z,mtbf)/talpha
       ):(s3'=NORMAL)   +(1-pow(exp,-mod(z,mtbf)/talpha))
       :(s3'=SCHEDULED) ;
5  [Unscheduled] s3=NORMAL ->  pow(exp,-mod(z,mtbf)/talpha
       ):(s3'=NORMAL)+(1-pow(exp,-mod(z,mtbf)/talpha)):(
       s3'=UNSCHEDULED) ;
6  [Failure]     s3=NORMAL ->  r:(s3'=NORMAL )     +(1-r)
       :(s3'=FAILURE) ;
7  [Normal]      s3=NORMAL | s3=SCHEDULED | s3=UNSCHEDULED
       -> (s3'=NORMAL);
8  [Reset]       s3=Failure-> (s3'=NORMAL);
9  endmodule
```

Listing V.2: OnOrbit Satellite Manoeuvers

```
1  module OnOrbitManoeuvers
2  // State variable declaration with initial value
3  s2: [0..10] init 0;  // s2 is an integer variable
       ranging from 0 to 10, initialized to 0
4  // State transition - repair on-orbit software update
5  [RepairOnOrbitSoftwareUpdate]           // Label for
       the transition
6  s2=STANDBY -> (s2'=UPDATE);       // The system
       transitions from standby (s2=STANDBY) to update
       state (s2'=UPDATE) for software update
7  // State transition - check for redundant satellite (
       from update state)
8  [CheckOnOrbitRedundantSatellite]        s2=UPDATE  ->
       (1-pow(exp,-mod(z,mtbf)/tmttr)):(s2'=
       CHECKSATTELITE)+(pow(exp,-mod(z,mtbf)/tmttr)):(s2
       '=STANDBY) ;
9  // State transition - move/replace redundant satellite
       (from check state)
10 [MoveReplaceRedundantSatellite]         s2=
       CHECKSATTELITE           -> Pbeta:(s2'=REPLACE)
       +(1-Pbeta):(s2'=ONGROUNDSPARE);
11 // State transition - reset to standby
12 [ResetOnOrbitManoeuvers]                // Label for
       the transition
13 s2=ONGROUNDSPARE | s2=REPLACE | s2=UPDATE    -> (s2'=
       STANDBY);  // The system can reset to standby from
        various states (ONGROUNDSPARE, REPLACE, UPDATE)
       and sets s2' (next state) to standby
14 endmodule
```

The On-ground player model, denoted by $\mathscr{P}_g$, is detailed in Listing V.3. The model encapsulates five commands that interpret global on-ground behavior from Figure 2. The first command (line 5) 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 6 with probability $1 - e^{mtbf/t_\gamma}$(where $t_\gamma$represents the mean-time to build a satellite). In this case, ground control initiates preparations for launching the new satellite. When the satellite is built (line 8, taking t_δ time), they prepare it for launch (line 10). Finally, when the satellite is ready for launch (line 12), the ground staff returns to standby mode with probability P_n.

Listing V.3: On-Ground Satellite Manoeuvers

```
1  module OnGroundManoeuvers
2  s1: [0..10] init 0; //refers to the on-ground modes
3  //The satellite on the ground is on standby mode
4  [OnGroundStandby] s1=STANDBY  -> (s1'=ONGROUNDSPARE);
5  //Building mode of a new satellite or initiate the
       launch
6  [CheckOnGroundSatelliteToBuild] s1=ONGROUNDSPARE  ->
       (1-pow(exp,-mod(z,mtbf)/tgamma)):(s1'=BUILD)+(pow(
       exp,-mod(z,mtbf)/tgamma)):(s1'=LAUNCH);
7  //build a new satellite or fail and  standby mode
8  [CheckOnGroundSatelliteToManufecture]   s1=BUILD  ->
       (1-pow(exp,-mod(z,mtbf)/tdelta)):(s1'=MANUFECTURE)
       +(pow(exp,-mod(z,mtbf)/tdelta)):(s1'=STANDBY);
9  // When the satellite is manufactured launch it
10 [BuildOnGroundSatellite] s1=MANUFECTURE    -> (s1'=
       LAUNCH);
11 //reset the on-ground operations
12 [ResetOnGroundManoeuvers] s1=LAUNCH -> (1-Pn):(s1'=
       LAUNCH)+(Pn):(s1'=STANDBY);
13 endmodule
```

a) Experimental setup: We have encoded properties in rPATL formalism. PRISM-games model checker v3.0 [3] (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 [23]) offering performance benefits for specific model structures.

b) Artifacts: The source code for the experiments described in this section is publicly available on a GitHub

repository[24].

B. 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*:

1) Reliability: What is the probability that a satellite will need to be replaced by a new one in 15 years, given a reliability of 0.80 regarding failures within 30 attempts?

$$\langle\langle \mathscr{P}_{env}, \mathscr{P}_{o}, \mathscr{P}_{g} \rangle\rangle \; \mathtt{P} = ?[\mathtt{F}\ (\text{“}\mathtt{replace}\text{”}\ \&\ \mathtt{rounds} <= \mathtt{k})], \mathtt{k} = 1:30:1 \quad \text{(PRO1)}$$

2) Maintainability: How many times would a satellite need to be replaced by a new one in 15 years, assuming a reliability of 0.80 regarding failures within 30 attempts?

$$\langle\langle \mathscr{P}_{env}, \mathscr{P}_{o}, \mathscr{P}_{g} \rangle\rangle \; \mathtt{R}\{\text{“}\mathtt{replace}\text{”}\} = ?[\mathtt{F}\ (\mathtt{rounds} <= \mathtt{k})], \mathtt{k} = 1:30:1 \quad \text{(PRO2)}$$

3) Availability: What is the availability of the satellite in 15 years, assuming a reliability of 0.80 regarding failures within 30 attempts?. We use T as a reference to the maximum number of rounds (attempts) to reach a successful replacement.

$$\langle\langle \mathscr{P}_{env}, \mathscr{P}_{o}, \mathscr{P}_{g} \rangle\rangle \; \mathtt{R}\{\text{“}\mathtt{replace}\text{”}\} = ?[\mathtt{F}\ (\mathtt{rounds} <= \mathtt{k})]/\mathtt{T}, \mathtt{k} = 1:30:1 \quad \text{(PRO3)}$$

4) Maintenance cost: What is the cost of the satellite replacement in 15 years, assuming a reliability of 0.80 regarding failures within 30 attempts?

$$\langle\langle \mathscr{P}_{env}, \mathscr{P}_{o}, \mathscr{P}_{g} \rangle\rangle \; \mathtt{R}\{\text{“}\mathtt{cost}\text{”}\} = ?[\mathtt{F}\ (\mathtt{rounds} <= \mathtt{k})], \mathtt{k} = 1:30:1 \quad \text{(PRO4)}$$

To address the properties mentioned above, we propose extending the model with a module to track the number of rounds synchronized with module actions. Additionally, we will incorporate a reward structure to model replacement times and costs.

First, we enhance the model by incorporating an integer constant as in [9] and a module (see Listing V.4) to keep track of the number of rounds. 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 V.4: Rounds Module [9], [25]

```
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 five reward structures in Listing V.5. These structures correspond to the replacement times for in-orbit and on-ground operations (lines 3-6). They are divided into separate reward structures for in-orbit and on-ground replacements (lines 7-12). The cost of these replacements is modeled in lines 13-18. Here, c1 represents the cost of replacing a satellite in orbit, while c2 represents the cost of manufacturing a new satellite on the ground. *Properties in PRO2 and PRO4 with tag name "replace" and "cost" can be replaced with on-orbit and on-ground replace and cost as rewards names in Listing V.5.*

Listing V.5: Reward Structures in PRISM

```
1  const double c1; //Cost of replacing a satellite in
       orbit
2  const double c2; //Cost of manufacturing a new
       satellite on the ground
3  rewards "replace" // Reward for replacing a satellite
       on ground and on-orbit
4    s2=REPLACE : 1;  // Replace satellite with a new one
5    s1=MANUFECTURE : 1; // Manufacture a new satellite to
         replace satellite
6  endrewards
7  rewards "replaceOnOrbit" // Reward for replacing a
       satellite on orbit
8    s2 = REPLACE : 1; // Replace satellite with a new one
9  endrewards
10 rewards "replaceOnGround" // Reward for replacing a
       satellite on ground
11   s1 = MANUFACTURE : 1; // Manufacture a new satellite
       to replace satellite
12 endrewards
13 rewards "costOnOrbit" // Cost of replacing a satellite
       on orbit
14   s2 = REPLACE : c1 * rounds; // Replace satellite with
       a new one, incurring a cost proportional to
       rounds
15 endrewards
16 rewards "costOnGround" // Cost of replacing a satellite
       on ground
17   s1 = MANUFACTURE : c2 * rounds; // Manufacture a new
       satellite to replace satellite, incurring a cost
       proportional to rounds
18 endrewards
```

C. Experiments results analyses

The verification results for PRO1 are shown in Figure 4. We observe that as the number of rounds increases, the probability of replacement for satellites in orbit or on the ground also increases. This is likely because the staff are more willing to replace satellites in multiple rounds to ensure maximum functionality and avoid failures throughout their 15-year lifespan. The verification results for PRO2 are shown in

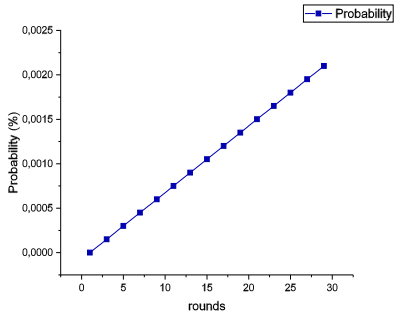

Figure 4: Verif. PRO1 .

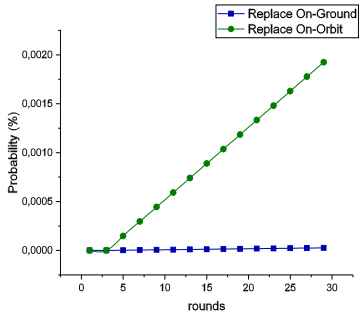

Figure 5: Verif. PRO2.

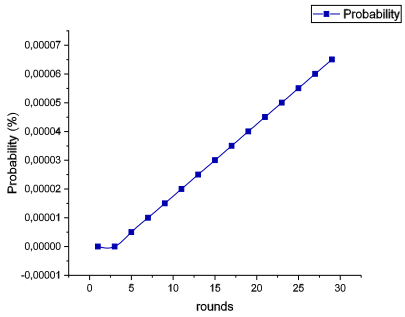

Figure 6: Verif. PRO3 .

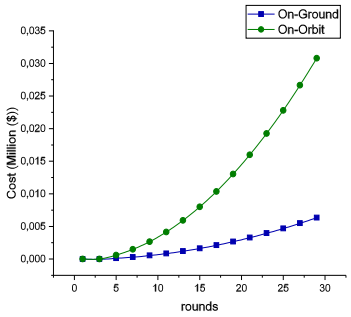

Figure 7: Verif. PRO4.

Figure 5. As the number of rounds increases, the probability of replacing both on-orbit and ground satellites also increases. However, on-orbit maintenance is more frequent than ground maintenance. This is due to the higher cost of manufacturing and launching a new satellite. Consequently, on-orbit staff tend to prioritize repairing existing satellites or finding spare ones in orbit whenever possible.

The verification results for PRO3 are shown in Figure 6. We observe that the availability of the satellite increases. This can be interpreted as an increase in the satellite replacement rate as the number of rounds increases. This strategy aims to maximize the satellite's availability after 30 rounds. However, it's important to note that the maximum achievable replacement rate is equivalent to 0.008%, which represents the limit of the staff's effort. The verification results for Equation PRO4 are shown in Figure 7. These results indicate that on-orbit staff maintenance incurs a cost of $0.035 million more than on-ground maintenance. This is because staff prioritize on-orbit over ground replacements. Interestingly, our initial assumption was that on-ground maintenance would be more expensive than on-orbit maintenance. However, the actual cost depends on the total number of replacements performed (as shown in Figure 7). This suggests that on-ground staff can develop a maintenance strategy that achieves higher satellite availability in fewer rounds, even though this strategy might require a larger initial maintenance budget.

D. *Threats to validity*

Our model focuses on a single satellite system. (i) For scenarios involving multiple satellites, additional staff collaboration would be necessary for repairs and maintenance, potentially introducing a first-in-first-out (FIFO) queueing system. Furthermore, the maintenance parameters may change. (ii) The model does not consider communication between the ground station and the satellite system, which factors like solar radiation can impact. (iii) The interaction between on-orbit and ground staff has not been explicitly modeled in our current approach. However, their collaboration can significantly impact the maintenance strategy like the team size and involvement cost.

E. *Discussion*

Strategic maintenance can enhance the quality of a satellite system while optimizing failure repair. The model can be extended with additional commands to simulate communication. Our current focus is on existing commands implemented by our collaborators and researchers.

The model is parametric, allowing customizable PRISM-games models via parameters. The data employed aligns closely with the models described in [6], [7]. However, a key difference lies in the model formalism. Previous implementations relied on CTMC and MDPs, whereas this work leverages the CSG formalism to include the human response.

To our knowledge, this is the first implementation using CSGs for RAM analysis in satellite systems with collaborative maintenance (based on the PRISM library [26]).

Failures are primarily modeled using an exponential distribution. Additional distributions can be incorporated through appropriate model updates, reflecting the failure distribution of specific components.

VI. Conclusion

This paper demonstrates how a probabilistic model checker, specifically the PRISM-games, can be effectively used to model collaborative maintenance between on-orbit and ground staff. We evaluated the collaborative operation's effectiveness by performing a RAM (Reliability, Availability, and Maintainability) analysis on the satellite system. Previous research has primarily focused on improving model performance, neglecting the role of human intervention and response to failures.

Our future work will concentrate on analyzing maintenance strategies for constellations of satellite systems while ensuring a certain level of communication reliability between staff and equipment. This will ultimately lead to improved maintenance planning and investment optimization.

Acknowledgements

The research leading to the presented results was conducted within the research profile of Human-Centric Collaborative Architectural Decision-Making for Secure System Design (HERMES-Design[1]) supported by Institut de Cybersécurité d'Occitanie (ICO).

References

[1] Marta Kwiatkowska, Gethin Norman, and David 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*, pages 585–591. Springer, 2011.

[2] Marta Kwiatkowska, Gethin Norman, David Parker, and Gabriel Santos. Equilibria-based probabilistic model checking for concurrent stochastic games. In *Formal Methods – The Next 30 Years*, pages 298–315, Cham, 2019. Springer International Publishing.

[3] Marta Kwiatkowska, Gethin Norman, David Parker, and Gabriel Santos. Prism-games 3.0: Stochastic game verification with concurrency, equilibria and time. In *Computer Aided Verification*, pages 475–487, Cham, 2020. Springer International Publishing.

[4] M. Kwiatkowska, G. Norman, D. Parker, and G. Santos. Automatic verification of concurrent stochastic systems. *Formal Methods in System Design*, 58(1):188–250, 10 2021.

[5] Prism - bibliography - prism model checker. https://www.prismmodelchecker.org/bib.php. [Accessed: July 10, 2024].

[6] Khaza Anuarul Hoque, Otmane Ait Mohamed, and Yvon 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)*, pages 1635–1640, 2015.

[7] Yu Lu, Zhaoguang Peng, Alice Miller, Tingdi Zhao, and Chris W. Johnson. How reliable is satellite navigation for aviation? checking availability properties with probabilistic verification. *Reliab. Eng. Syst. Saf.*, 144:95–116, 2015.

[8] Zhaoguang Peng, Yu Lu, Alice Miller, Chris W. Johnson, and Tingdi Zhao. A probabilistic model checking approach to analysing reliability, availability, and maintainability of a single satellite system. In David Al-Dabass, Alessandra Orsoni, and Zheng Xie, editors, *Seventh UKSim/AMSS European Modelling Symposium, EMS 2013, 20-22 November, 2013, Manchester UK*, pages 611–616. IEEE, 2013.

[9] M. Kwiatkowska, G. Norman, D. Parker, and 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*, pages 298–315. Springer, 2019.

[10] Marta Kwiatkowska, Gethin Norman, David Parker, and Gabriel 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*, page 60–78. Springer, 2022.

[11] Abdelhakim Baouya, Brahim Hamid, Levent Gürgen, and Saddek Bensalem. Rigorous security analysis of rabbitmq broker with concurrent stochastic games. *Internet of Things*, 26:101161, 2024.

[12] PRISM-games. PRISM-games - Case Studies - PRISM model checker. https://www.prismmodelchecker.org/games/casestudies.php, 2024. [Accessed: July 10, 2024].

[13] Javier Cámara, David Garlan, Bradley Schmerl, and Ashutosh 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, page 428–435, New York, NY, USA, 2015. Association for Computing Machinery.

[14] Javier Cámara, Gabriel Moreno, and David 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*, pages 146–156, 2015.

[15] Kaustabha Ray. Adaptive service placement for multi-access edge computing: A formal methods approach. In *2023 IEEE International Conference on Web Services (ICWS)*, pages 14–20, 2023.

[16] Kaustabha Ray and Ansuman Banerjee. Prioritized fault recovery strategies for multi-access edge computing using probabilistic model checking. *IEEE Transactions on Dependable and Secure Computing*, 20(1):797–812, 2023.

[17] Zhaoguang Peng, Yu Lu, Alice Miller, Chris W. Johnson, and Tingdi Zhao. Risk assessment of railway transportation systems using timed fault trees. *Qual. Reliab. Eng. Int.*, 32(1):181–194, 2016.

[18] Vojtech Forejt, Marta Kwiatkowska, Gethin Norman, and David Parker. Automated verification techniques for probabilistic systems. In *Formal Methods for Eternal Networked Software Systems (SFM'11)*, volume 6659 of *LNCS*, pages 53–113. Springer, 2011.

[19] Taolue Chen, Vojtěch Forejt, Marta Kwiatkowska, David Parker, and Aistis Simaitis. Automatic verification of competitive stochastic systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 7214, pages 315–330. Springer Berlin Heidelberg.

[20] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. The MIT Press. OCLC: ocn171152628.

[21] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, sep 2002.

[22] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. 6(5):512–535.

[23] PRISM Development Team (eds.). Prism manual. https://www.prismmodelchecker.org/manual/ConfiguringPRISM/ComputationEngines. Accessed: July 10, 2024.

[24] Abdelhakim Baouya. Paper Artefacts Sources. https://hermes-design.github.io/seaa2024.html.

[25] PRISM Model Checker. Public good game case study. https://www.prismmodelchecker.org/casestudies/public_good_game.php, 2020. [Accessed: July 10, 2024].

[26] PRISM Model Checker. Prism-games - publications. https://www.prismmodelchecker.org/games/publ.php. [Accessed: July 10, 2024].

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