POLITECNICO
MILANO 1863

UppaalTD: a Formal Tower Defense Game

Formal Methods for Concurrent and Real-Time Systems

Andrea Manini and Pierluigi San Pietro

Authors: **Ali Bassam - 10879115**

Leonardo Carlo Conti - 10773470

Satvik Sharma - 11054680

Shi Zqiang - 10949772

A.Y: 2024-2025

Version: 1.0

Contents

1. Introduction

1.1 Overview

Tower defense games are a genre of strategy games in which players must defend a designated asset, which is commonly referred to as the Main Tower, against waves of incoming enemies. The player achieves this by strategically placing various types of turrets on a map to intercept and eliminate enemies before they reach the objective.

This report presents a formal analysis of a Tower Defense Game using Uppaal, a model checker for real-time systems based on networks of timed automata. The goal is to create a formal model of the game mechanics and verify key behavioral properties of the system under both deterministic (vanilla) and stochastic conditions. The analysis is conducted from the perspective of a Quality Assurance Engineer, aiming to verify that the game behaves correctly and consistently under various configurations.

Through formal modeling and verification, enemy movement, turret behavior, and different game outcomes are examined to ensure that the system satisfies the intended rules and constraints. The final objective is to assess whether specific turret configurations lead to a win or loss and to quantify system behavior under uncertainty using Uppaal.

1.2 Game Description

The Tower Defense Game modeled in this project is structured around a 16×8 grid-based map, as shown in **Figure 1.1**. The objective is to defend a central asset, referred to as the Main Tower, from waves of incoming enemies by strategically placing turrets on predefined cells. The game involves several key components, each governed by a specific set of rules and parameters.

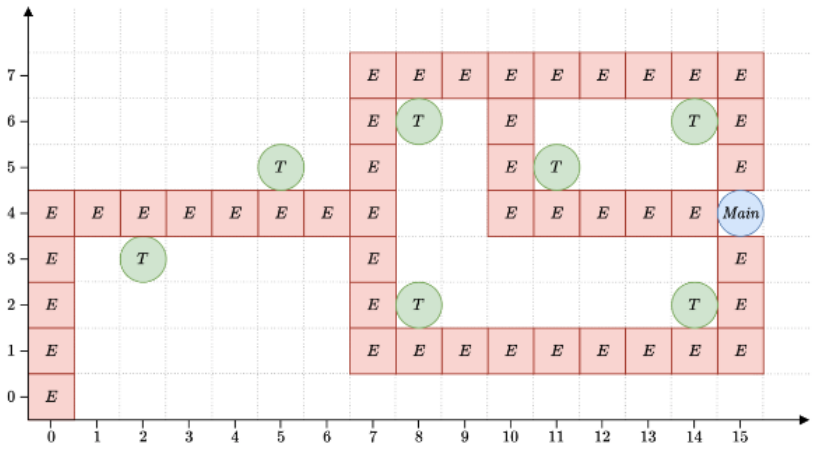

Figure 1.1: Game Map

The Map is divided into cells of three types:

- Blue cell: Contains the Main Tower
- Green cells: Allowed positions where the player can place turrets
- Red cells: Form the path enemies must follow to reach the Main Tower

Turrets can only be placed on green cells, while enemies can move only along the red cells. Multiple enemies can occupy the same cell simultaneously. Enemies cannot deviate from the red path and always move forward towards the Main Tower. At the bifurcation point, enemies non-deterministically select one of the possible paths.

The *Main Tower* starts with a predefined number of life points. Each enemy that successfully reaches the Main Tower attacks it once, reducing its life points by an amount corresponding to its damage value. If the Main Tower's life points drop to zero or below, the game is lost.

The game features two types of enemies, **Circles, and Squares**, each defined by three parameters:

- Speed: Time delay between consecutive moves
- Health: Maximum life before dying
- Damage: Amount of life points subtracted from the Main Tower upon attack

Enemies spawn at the starting cell (0, 0) and move one cell at a time after a delay equal to their speed. Upon reaching the Main Tower, an enemy attacks once and then leaves the map after a delay equal to its speed. Dead enemies or enemies who have left the map cannot be targeted by turrets.

The game also features three types of turrets, **Basic**, **Cannon**, **and Sniper**, each characterized by:

- Shooting range: Enemies within this area can be targeted
- Firing speed: Time between consecutive shots
- Inflicted damage: Amount of health reduced from the enemy per shot

Turrets are limited to engaging a single enemy at a time within their firing range. They focus on the nearest enemy. If several enemies are equally close, the turret selects the one that arrived most recently on the Map. When both Circles and Squares are valid targets, Squares are given priority.

The Game Execution starts with only the Main Tower on the map. Players can place any number of turrets on available green cells before the enemy wave begins. There are no restrictions on the number or types of turrets, except for the available turret positions. Once the wave starts, enemies spawn periodically based on their spawning times. Turrets must intercept and destroy the enemies before they reach and damage the Main Tower. A wave concludes when either all enemies are eliminated, the Main Tower is destroyed, or a timeout expires.

2. Game Modelling

The Tower Defense game was formally modeled using UPPAAL as a network of timed automata. The system comprises several key components, each encoded as separate automata: ***Controller, EnemyProcess, and TurretProcess***.

2.1 Map and Path Structure

The fixed array is used because the enemy path is predetermined and does not change during simulation. By representing each segment as a fixed array of Coordinate structures, the model can precisely define the layout of the path, including any branches. Since the structure of the path is static, a fixed array provides a simple and efficient way to store coordinates, supporting bifurcations without the need for dynamic memory or data manipulation. This setup aligns with UPPAAL's verification approach, where non-deterministic path choices at junctions are explored using binary tree logic to ensure that all possible routes are considered.

2.2 Templates Overview

2.2.1 Controller Template

The *Controller* manages the overall game flow, including turret placement, enemy spawning, and checking win/loss conditions. It initializes turrets based on a selected configuration and sets their parameters such as range, fire speed, and damage according to turret type. It handles the timed spawning of different enemy types (Circle, Square) and tracks key variables like turrets placed and enemies spawned. The Controller also determines when the game ends, either by eliminating all enemies or when the tower's life reaches zero. Communication is managed through synchronization channels for turret placement and enemy spawning.

2.2.2 Enemy Template

The EnemyTemplate defines the dynamic behavior of each enemy in the game. Enemies spawn at the initial position (0,0) and follow a segmented path composed of predefined coordinates. Their movement is controlled by a timer, and their trajectory can branch at specified points using binary decisions, allowing for path variation. Each enemy maintains its own state, including attributes such as type, current position, health, and attack capabilities.

To optimize query execution time and ensure that verification queries terminate, enemy behavior is modeled deterministically where possible, and path choices are constrained to binary logic. While progressing along their path, enemies regularly scan for nearby turrets using the findTurretsToAwaken() function and send signals only to turrets capable of targeting them. This selective signaling reduces unnecessary synchronization and contributes to faster query resolution.

When an enemy reaches the endpoint at coordinates (15,4), it attacks the Main Tower via the attack() method and then deactivates itself. Channels are used to manage enemy spawning and turret awakening, ensuring efficient and concurrent coordination throughout the simulation. This design minimizes nondeterminism and supports UPPAAL in completing queries within a reasonable time frame.

2.2.3 Turret Template

The TurretTemplate governs how individual turrets respond to nearby enemies and manage their firing cycle. After being placed via turretPlacing[i]?, a turret initializes and transitions to the Ready state. In this state, the turret does not continuously check for enemies; instead, it passively waits for a turretAwakening[i]? signal. This signal indicates that a nearby enemy may be within range, allowing the turret to invoke checkEnemy() only when necessary. If a valid target is detected, the turret fires using shoot(int enemyId) and transitions into the Reload state. This state is timed using fireTimer until the defined fireSpeed() is reached, after which the turret returns to Ready.

By relying on external awakening signals rather than active polling, turrets avoid unnecessary checks in the Ready state. This design reduces the number of transitions and guards UPPAAL must evaluate during verification, thereby optimizing query performance and ensuring better scalability. Synchronization is handled through dedicated turret placing and awakening channels.

2.3 Game Parameters

Figure 2.1 lists key constants used to define the attributes and behavior of enemies and turrets in the game.

Parameter	Value
Enemies Spawning Time	
Circle Spawning Time	2
Square Spawning Time	3
Parameters of Different Enemy Types	
Circle Speed	1
Circle Health	10
Circle Damage	2
Square Speed	3
Square Health	20
Square Damage	4
Parameters of Different Turret Types	
Basic Shooting Range	2
Basic Fire Speed	2
Basic Inflicted Damage	2
Cannon Shooting Range	1
Cannon Fire Speed	7
Cannon Inflicted Damage	5
Sniper Shooting Range	4
Sniper Fire Speed	20
Sniper Inflicted Damage	8
Parameters of the Main Tower	
Main Tower Life Points	10

Figure 2.1: Game Parameters

2.4 Game Configurations

2.4.1 Configuration 1 - Default (Winning)

- **Turrets:** 7
- **Layout:**
    - Basic: (5,5)
    - Cannons: (8,2), (8,6), (14,2), (14,6)
    - Snipers: (2,3), (11,5)

2.4.2 Configuration 2 - Custom 1 (Winning)

- **Turrets:** 5
- **Layout:**
    - Basics: (5,5), (8,6)
    - Cannon: (11,5)
    - Snipers: (2,3), (14,2)

2.4.3 Configuration 3 - Custom 2 (Losing)

- **Turrets:** 4
- **Layout:**
    - Cannons: (14,2), (14,6), (11,5), (8,2)

3. Vanilla Version

The vanilla version represents the deterministic baseline of the Tower Defense game, where no stochastic elements (e.g., randomness in movement or turret behavior) are present. This version serves to validate the functional correctness of the game mechanics and ensures that all interactions conform to the specified design.

3.1 Configuration and Parameters

This version includes **6 enemies (3 Circles and 3 Squares)** and a Main Tower starting with **10 life points**. Turrets can be placed in various configurations.

3.2 Purpose and Use

The vanilla version offers a stable and deterministic foundation to:

- Experiment with turret configurations
- Analyze enemy flow and Main Tower survivability
- Set a baseline before introducing randomness in the stochastic version

It is also critical from a *Quality Assurance perspective* to ensure the model is correct before adding complex, probabilistic features.

3.3 Verification of Properties

To ensure the model's correctness, a set of properties were verified using UPPAAL's query engine. These properties include both enemy-only scenarios and turret-based gameplay. **All required properties listed in the project specification were successfully verified**, validating the deterministic model's correctness.

S.No.	Property	Status
1	The game always terminates without a deadlock.	Passed
2	All enemies reach the Main Tower.	Passed
3	Circles reach the tower within expected time bounds.	Passed
4	Squares reach the tower within expected time bounds.	Passed
5	Enemies never leave the valid red path.	Passed
6	Correctly identifies if the given configuration is winning/losing.	Passed

S.No.	Property	Status
7	The game reaches termination without deadlock for the given configuration.	Passed
8	Two additional configurations were tested and classified accordingly (one winning, one losing)	Passed

These validations confirm that the model:

- Behaves according to the designed rules.
- Respect timing and spatial constraints.
- Appropriately reflects targeting priorities and turret firing behavior.

4. Stochastic Version

The stochastic version extends the deterministic baseline by introducing probabilistic behavior into the model. This version leverages **UPPAAL SMC** to simulate a more realistic gameplay scenario, where uncertainties in turret firing and enemy movement are modeled using **exponential probability distributions**.

4.1 Configuration and Parameters

This version introduces a large-scale wave composed of 700 enemies: 400 Circles and 300 Squares. The Main Tower begins with at least 100 life points. All turret and enemy parameters—such as damage, range, and base speed—remain consistent with the vanilla version.

Two major elements now exhibit stochastic behavior to reflect more realistic and unpredictable gameplay dynamics:

- Enemy speed follows an exponential distribution with rate = speed / 10.
- Turret firing speed follows an exponential distribution with rate = 1 / fireSpeed.

To support this stochastic modeling, all communication channels (e.g., for turret placement, awakening, and enemy spawning) are configured as broadcast channels. This ensures proper synchronization under stochastic semantics and enables non-deterministic yet analyzable interactions during simulation and verification.

4.2 Purpose and Use

The stochastic version enables:

- Statistical analysis of game outcomes under probabilistic conditions.
- Estimation of **Main Tower survivability** across simulations.
- Evaluation of turret configuration robustness under random delays.
- Comparative analysis against the deterministic (vanilla) baseline.

This version is crucial for modeling **real-world unpredictability** and validating the system under varying conditions, helping uncover rare but possible behaviors.

4.3 Verification of Properties

The correctness and performance of the stochastic model were validated using **UPPAAL's Statistical Model Checking (SMC)** engine. Queries were focused on estimating probabilities and expected values based on multiple simulation runs. The main configuration tested was the same as in the Vanilla section, supplemented by two additional turret layouts.

S.No.	Property	Status
1	Estimate the average time until the Main Tower is first damaged (within 200 time units).	Passed
2	Estimate the probability that the Main Tower survives the full 200-time units (default configuration).	Passed
3	Simulated and analyzed 2 other significant configurations for behavioral insights.	Passed

---

4.4 Results

- **Property 1: Expected Time Before Main Tower is Damaged**
  Estimated Time Before First Damage: *Approximately 4 seconds*
- **Property 2: Probability of Main Tower Survival**
  Estimated Probability of Survival: *~0.049941*
- **Property 3: Analysis of Custom Configurations**
  To gain further insights into gameplay dynamics under stochastic behavior, two additional turret configurations were simulated and analyzed using UPPAAL SMC. The first configuration, which included two Basic turrets (at positions (5, 5) and (8, 6)), two Sniper turrets (at (2, 3) and (14, 2)), and one Cannon turret (at (11, 5)), demonstrated a strong defense. The Main Tower survived the entire 200-time unit simulation, indicating that this configuration effectively balanced range, damage, and firing rate to manage the large wave of enemies. In contrast, the second configuration relied solely on Cannons, placing them at (14, 2), (14, 6), (11, 5), and (8, 2). Despite their high damage, the Main Tower was eventually defeated, primarily due to the Cannons' slower firing speed and shorter range, which left critical gaps in coverage. This comparative analysis emphasizes that a mix of turret types and strategic placement is crucial for maximizing robustness in stochastic environments, where random delays in firing and movement significantly influence outcomes.

4.5 Observations and Insights

The stochastic model confirms:

- Variability in **enemy impact** times due to randomness in movement and turret delays.
- Some turret configurations that performed well deterministically may exhibit weaker performance probabilistically.
- **Main Tower survivability** is sensitive to turret placement strategy, especially under stochastic delays.

Statistical results provide both **quantitative metrics** (e.g., probability of survival) and **qualitative insights** (e.g., identifying vulnerable phases of gameplay), making this version essential for final game design validation.

5. Conclusion

The formal modeling and verification of the Tower Defense game were successfully completed using UPPAAL. In the vanilla version, all required properties were verified, confirming the model's correctness. Enemy movement followed valid paths, timing constraints were respected, and turret targeting behaved as specified—prioritizing Squares and resolving ties by spawn time. Multiple turret configurations were tested, accurately identifying winning and losing scenarios. No deadlocks occurred, and all enemies reached the Main Tower within expected bounds.

In the stochastic version, randomness was introduced in enemy speed and turret firing using exponential distributions. Statistical model checking estimated the Main Tower's survivability under uncertain conditions. The default turret configuration performed well, while others highlighted weaknesses caused by delayed responses or clustered enemies. The stochastic model revealed timing-sensitive behaviors that are not visible in the deterministic setup.

Together, both versions validate the system's logic and resilience. The vanilla model ensures baseline correctness, while the stochastic version provides insights into system behavior under real-world variability.

5.1 GitHub Repository

GitHub Repository: https://github.com/fmcsCaseStudy/caseStudy

---

5.2 System Configuration

- **Laptop:** Lenovo ThinkBook
- **Processor:** AMD Ryzen AI 9 365
- **RAM:** 32 GB (minimum 8 GB allocated required, if not heap error)
- **OS:** Windows 11 24H2
- **UPPAAL Version:** 5.0.0

6. Bibliography

- Manini, A., & San Pietro, P. (2025). UppaalTD: *A Formal Tower Defense Game – Project Statement*. Politecnico di Milano, Formal Methods for Concurrent and Real-Time Systems.
- Uppaal Official Website. Accessed April 2025.
- Uppaal Documentation. Accessed April 2025.
- Behrmann, G., David, A., & Larsen, K. G. (2004). A Tutorial on Uppaal. In M. Bernardo & F. Corradini (Eds.), Formal Methods for the Design of Real-Time Systems (pp. 200–236). Springer Berlin Heidelberg.
- David, A., Larsen, K. G., Legay, A., Mikucionis, M., & Poulsen, D. B. (2015). Uppaal SMC Tutorial. International Journal on Software Tools for Technology Transfer, 17(4), 397–415.