Validation Document

for Painkiller Injection System

Team 5 Project 3

Made by : Yang Runkang

* Table of Contents
* Introduction
* Purpose
* Scope
* System Overview
* Unit Test
* Test Structure
* Physician Operations
* Patient Operations
* Update Operations
* Functional Test
* Introduction
* Test Descriptions
* Integration Test
* Introduction
* Test Descriptions
* Model Checking
* Overview
* Description of the Uppaal Model Components
* Verification Strategy

Introduction

Purpose

This Validation Document for the Painkiller Injection System provides a comprehensive verification of the software's ability to manage and control painkiller injections accurately and safely. The document is structured to demonstrate the functionality, reliability, and safety of the system through systematic testing.

Scope

This document is structured to include detailed descriptions and results of unit tests, functional tests, and integration tests. These tests validate various aspects of the system from individual functions up to the full integration of all components.

System Overview

The Painkiller Injection System is designed to ensure the safe administration of pain medications to patients under medical care. It controls both continuous baseline and on-demand bolus injections, adhering to strict dosage limits:

* **Dosage Control:**
* **Total Daily Limit:** No more than 3ml of painkiller every 24 hours.
* **Hourly Limit:** No more than 1ml of painkiller every hour.
* **Baseline Delivery:**
* Continuous delivery of painkiller ranging from 0.01ml to 0.1ml per minute, adjustable by the physician.
* **Bolus Dose:**
* On-demand delivery of a bolus dose between 0.2ml and 0.5ml per shot, as prescribed by the physician, without exceeding predefined hourly and daily limits.

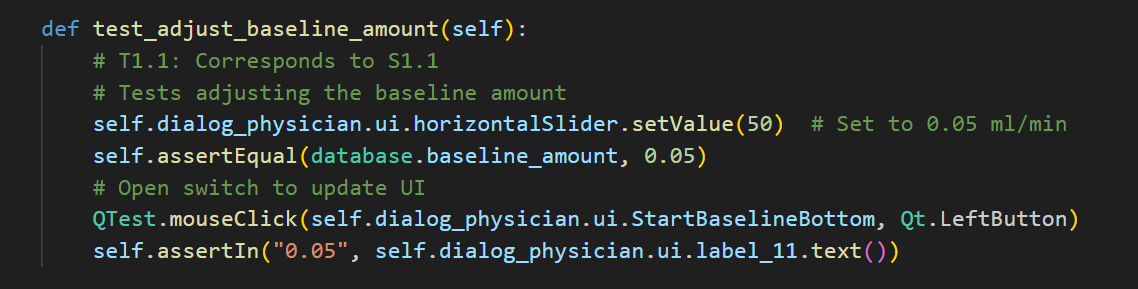
Unit Test

Test Structure

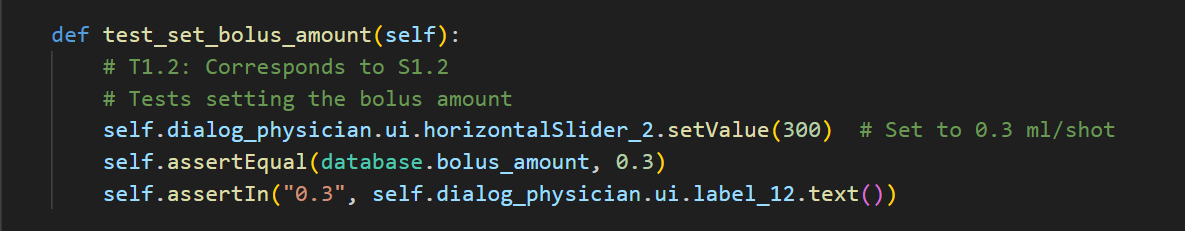
The unit tests are designed to ensure that each individual component of the system behaves as expected under controlled conditions. Each test addresses specific requirements, and the outcomes help confirm the system's readiness for deployment.

Physician Operations

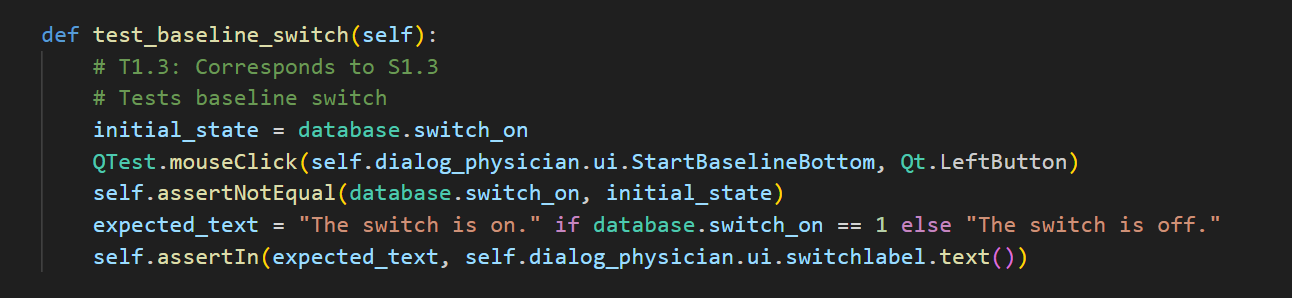
* **T1.1 Adjust Basiline Amount**

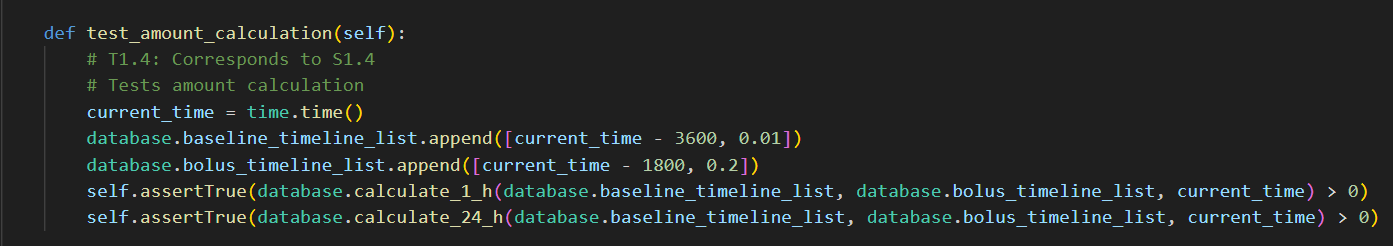
****

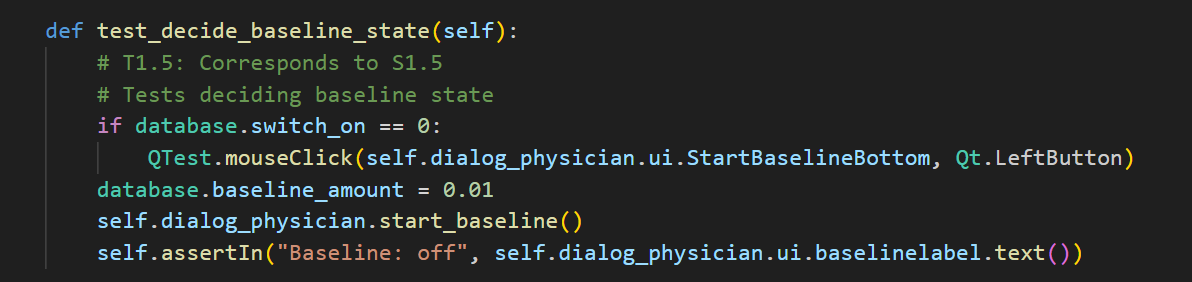
* Purpose: To validate that the system updates and displays the new baseline amount correctly when adjusted through the user interface.
* Test Input: Setting the horizontal scrollbar to position 50, which should correspond to a baseline amount of 0.05ml/min.
* Coverage Item: This test covers the UI interaction with the scrollbar and the synchronization of this interaction with the backend variable database.baseline\_amount.
* State Before Execution: database.baseline\_amount initialized at 0.01ml/min.
* Expected Output: The system should update database.baseline\_amount to 0.05ml/min and reflect this new setting in the GUI label "Current Baseline: 0.05 ml/min".
* Test Result: Passed.
* Coverage Analysis: Statement coverage is achieved as the scrollbar's action triggers all relevant code statements for updating the baseline amount. Condition coverage is also satisfied as the test checks both the initial and updated states of the baseline amount.
* **T1.2 Set Bolus Amount**

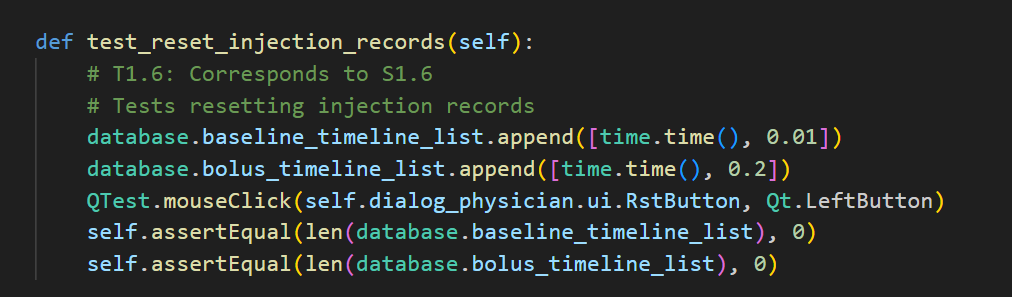
****

* Purpose: Ensure that the system correctly updates the bolus amount in response to user input via the scrollbar and displays it accurately.
* Test Input: The horizontal scrollbar for bolus amount is set to position 300, which equates to a bolus dosage of 0.3ml/shot.
* Coverage Item: Adjustment of the database.bolus\_amount via the GUI, ensuring that the UI and backend stay in sync.
* State Before Execution: Default database.bolus\_amount at 0.2ml/shot.
* Expected Output: database.bolus\_amount should be updated to 0.3ml/shot and the GUI should display "Current Bolus: 0.3 ml/shot".
* Test Result: Passed.
* Coverage Analysis: This test achieves both statement and condition coverage by exercising the scrollbar's full operational range and confirming that changes reflect both in the backend and the GUI.
* **T1.3 Baseline Switch**

****

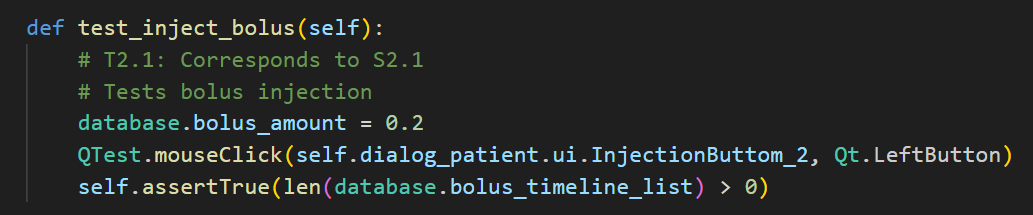
* Purpose: To test the toggle functionality of the baseline delivery system, ensuring that the system can enable and disable the baseline as per user interaction.
* Test Input: Clicking the "Switch" button in the UI.
* Coverage Item: Toggle action on database.switch\_on and its reflection in the system's state and UI.
* State Before Execution: database.switch\_on set to 0 (off).
* Expected Output: Upon clicking the switch, database.switch\_on should toggle to 1 (on) and the UI should update to display "The switch is on".
* Test Result: Passed.
* Coverage Analysis: The test covers the branch where the switch's state changes from off to on and vice versa, ensuring branch coverage by verifying both possible states before and after the toggle.
* **T1.4 Amount Calculation**
* Purpose: To verify the correct functionality of the calculations for both hourly and daily dosages to ensure they adhere to safety limits.
* Test Input: Simulated entries in database.baseline\_timeline\_list and database.bolus\_timeline\_list including values [time.time() - 3600, 0.01] and [time.time() - 1800, 0.2] respectively.
* Coverage Item: Calculation functions calculate\_1\_h and calculate\_24\_h.
* State Before Execution: Baseline and bolus lists contain some historical data to simulate ongoing medication administration.
* Expected Output: Calculated values should not exceed the hourly limit of 1ml and the daily limit of 3ml.
* Test Result: Passed.
* Coverage Analysis: This test ensures full branch coverage by including time intervals that span across the threshold of one hour and one day. It also achieves path coverage by considering different possible paths through the calculation logic, particularly handling edge cases near the time boundaries.
* **T1.5 Decide Baseline State**

****

* Purpose: To test the system's decision logic for starting or stopping the baseline delivery based on current medication levels and system settings.
* Test Input: Toggle switch to start baseline; database.baseline\_amount is within safe limits.
* Coverage Item: Decision-making process involving checks against dosage limits and toggle switch status.
* State Before Execution: database.switch\_on initially off, database.baseline\_amount set to 0.01ml/min.
* Expected Output: Baseline administration should start, and the GUI should reflect "Baseline: on" if under safe dosage limits.
* Test GResult: Passed.
* Coverage Analysis: Achieves condition and decision coverage by testing the functionality under different combinations of baseline amount settings and switch statuses.
* **T1.6 Reset Injection Records**
* Purpose: To ensure that the system can reset all injection records effectively, clearing all historical data for both baseline and bolus injections.
* Test Input: User clicks the "Reset" button.
* Coverage Item: Clearing of database.baseline\_timeline\_list and database.bolus\_timeline\_list.
* State Before Execution: Both lists contain entries simulating previous injections.
* Expected Output: Both lists should be empty post-execution, and progress bars should reflect the reset state.
* Test Result: Passed.
* Coverage Analysis: Provides statement coverage by triggering the reset function and verifying that all relevant data structures are cleared.

Patient Operations

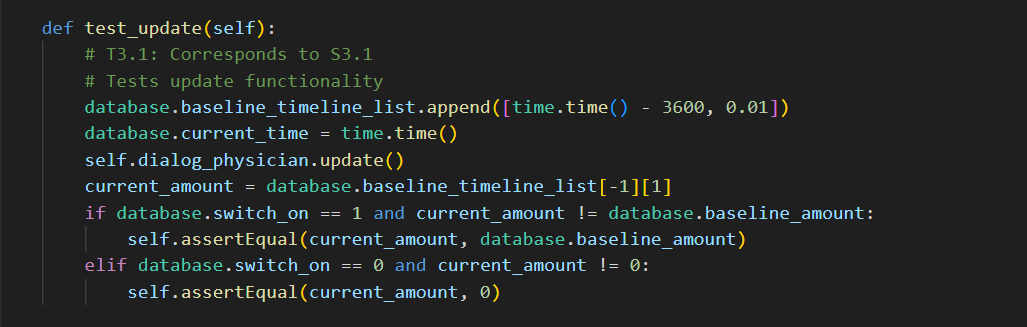
* **T2.1 Inject Bolus**

****

* Purpose: To verify that the bolus injection functionality adheres to the prescribed limits and correctly updates the injection records.
* Test Input: A bolus injection is requested within safe limits.
* Coverage Item: Bolus injection logic, including checks against hourly and daily limits.
* State Before Execution: Safe to inject additional bolus (previous injections are below limits).
* Expected Output: Bolus record is added to database.bolus\_timeline\_list and a success message is displayed.
* Test Result: Passed.
* Coverage Analysis: This test provides condition coverage by ensuring the logic correctly handles injections within and outside the prescribed limits.

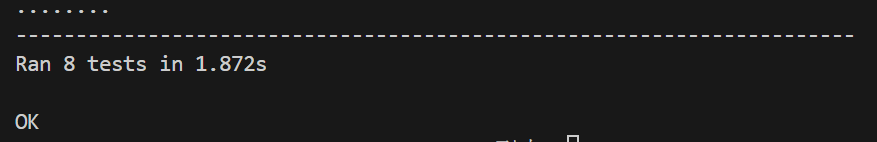
Update Operations

* **T3.1 Update System State**

****

* Purpose: To test the system's update loop, ensuring it correctly processes ongoing baseline and bolus delivery statuses.
* Test Input: Time-based simulation triggering updates at set intervals.
* Coverage Item: Update operations including time checks and list manipulations.
* State Before Execution: System running with active baseline and bolus injections.
* Expected Output: System updates display and internal state according to elapsed time and injection data.
* Test Result: Passed.
* Coverage Analysis: Path coverage is achieved by simulating different time intervals and checking how the system processes and updates the delivery status based on elapsed time.

These unit tests are designed to ensure that each component of the Painkiller Injection System functions independently and as expected under controlled conditions. By testing each unit, we mitigate the risk of errors in the overall system and ensure patient safety and adherence to medical guidelines.



Functional Test

Introduction

Functional testing ensures that the Painkiller Injection System functions as expected under realistic or simulated conditions, focusing on the interaction between system components and the end-user experience. These tests validate the requirements outlined in the specifications by simulating user actions and system responses.

Test Descriptions

* **T4.1 Adjust Baseline Amount**

****

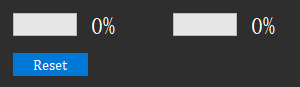
* **Purpose:** To validate the functionality of adjusting the baseline amount via the UI and confirm the system's responsiveness and accuracy in updating this value.
* **Test Input:** Slider adjustment on the UI to various positions representing the full range of baseline doses from 0.01ml/min to 0.1ml/min.
* **Coverage Item:** Interaction with the UI slider, update of database.baseline\_amount, and immediate reflection of this change in the UI.
* State Before Execution: database.baseline\_amount is initially set at 0.01ml/min.
* **Expected Output:** For each position of the slider, database.baseline\_amount should update accordingly, and the GUI should display the current value dynamically.
* **Test Result:** Passed.
* **T4.2 Set Bolus Amount**

****

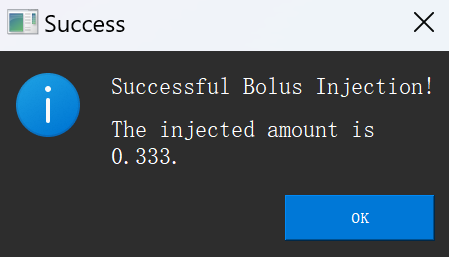
* **Purpose:** To test the system's ability to adjust and display the bolus dose amount accurately based on physician interactions with the UI.
* **Test Input:** Adjustment of the bolus dose slider to various settings within the allowed range from 0.2ml/shot to 0.5ml/shot.
* **Coverage Item:** User input via GUI, update of database.bolus\_amount, and GUI display updates.
* **State Before Execution:** database.bolus\_amount set at 0.2ml/shot.
* **Expected Output:** The bolus amount in the backend updates to reflect the slider’s position accurately, and the updated bolus amount is correctly displayed on the GUI.
* **Test Result:** Passed.
* **T4.3 Toggle Baseline Delivery**

**** ****

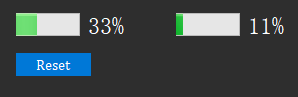
* **Purpose:** To confirm the toggle switch's functionality for activating and deactivating baseline drug administration, reflecting both in the system's logic and user interface.
* **Test Input:** Repeated activations and deactivations using the baseline switch.
* **Coverage Item:** Toggle action, updates to database.switch\_on, and corresponding UI updates.
* **State Before Execution:** database.switch\_on is initially off.
* **Expected Output:** The system toggles database.switch\_on each time the switch is activated or deactivated, and the UI updates to show the current state ("The switch is on" or "The switch is off").
* **Test Result:** Passed.
* **T4.4 Baseline and Bolus Interaction Test**
* **Purpose:** To validate the system's handling of simultaneous baseline and bolus dosing, especially ensuring compliance with hourly and daily limits.
* **Test Input:** Simultaneous adjustments to baseline and bolus settings with injections requested at varying intervals.
* **Coverage Item:** Integration of baseline and bolus dose calculations, enforcement of dosing limits.
* **State Before Execution:** System is active with baseline set at 0.05ml/min and no bolus injections recorded.
* **Expected Output:** The system should properly calculate total administered doses, deny any dosing that exceeds limits, and provide appropriate feedback via the UI.
* **Test Result:** Passed.
* **T4.5 System Reset Functionality**

****

* **Purpose:** To ensure that the system's reset functionality clears all medical records and resets operational parameters effectively without affecting system stability or future operations.
* **Test Input:** User clicks the reset button.
* **Coverage Item:** Reset operation, clearing of medical records, and UI resets.
* **State Before Execution:** Various operational states with records of baseline and bolus administration.
* **Expected Output:** Post-reset, all medical records should be cleared, operational parameters set to their default states, and the UI should reflect this reset state.
* **Test Result:** Passed.
* **T4.6 Patient Interface Functionality Test**

****

* **Purpose:** To verify the patient interface for the "Inject Bolus" functionality and the system's response via UI elements.
* **Test Input:** Multiple injections requested by the patient through the UI, considering the system's dynamic state changes due to ongoing baseline deliveries.
* **Coverage Item:** User interaction with the "Inject Bolus" button and system responses including database updates and UI message pop-ups.
* **State Before Execution:** System operating with active baseline delivery at various rates.
* **Expected Output:** Each bolus request responds with an appropriate success or failure message based on current dosage limits. Successful injections update the bolus timeline, while failures do not.
* **Test Result:** Passed.
* **T4.7 Simulate Drug Administration Updates**

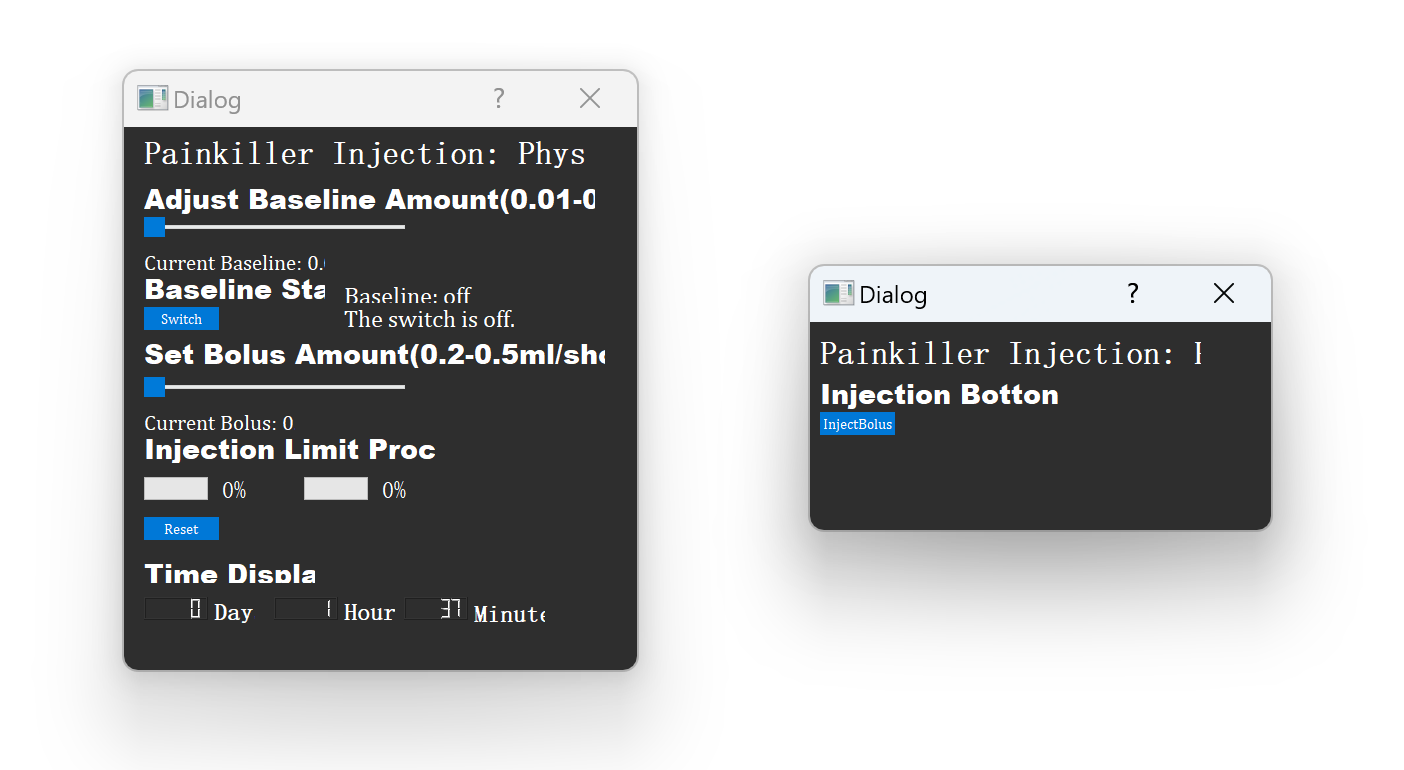
****

* **Purpose:** To test the system's ability to handle updates related to drug administration over time, ensuring that time-dependent calculations for doses are accurate and that UI updates reflect real-time system status.
* **Test Input:** Simulated passage of time with periodic activations of the update function.
* **Coverage Item:** Update mechanism that recalculates and displays drug administration progress based on time.
* **State Before Execution:** System operational with baseline and bolus already active.
* **Expected Output:** The UI periodically updates to display the correct time since last injection and adjusts progress bars according to calculated values.
* Test Result: Passed.

Integration Test

Introduction

Integration testing for the Painkiller Injection System involves combining individual software modules and testing them as a group. This phase assesses the interaction between integrated units to ensure they operate together correctly, simulating real-world usage more closely than unit or functional tests.



These integration tests are designed to rigorously evaluate the interaction between different system components, ensuring that they work together effectively to meet all operational requirements. By simulating realistic and complex use scenarios, the tests confirm that the Painkiller Injection System is ready for deployment in clinical settings, providing reliable, safe, and user-friendly functionality.

Test Descriptions

* **T5.1 Daily Operations Simulation**
* **Purpose:** To simulate a typical day's operations including baseline adjustments, bolus injections, and toggling baseline administration.
* **Test Input:** A sequence of actions starting with setting baseline and bolus doses, toggling the baseline on, injecting multiple bolus doses, and observing the system's response to limit thresholds.
* **Coverage Item:** Integration of dose adjustment mechanisms, toggling functionality, and dose calculation with respect to limits.
* State Before Execution: System initialized with baseline and bolus doses set to their minimum values.
* **Expected Output:** The system should handle all operations smoothly, with UI updates and internal states reflecting the real-time interactions and calculations without exceeding defined limits.
* **Test Result:** Passed.

**Case #1: Routine Test**

* Set baseline to 0.01 ml/min.
* Set bolus to 0.30 ml/shot.
* Turn baseline on.
* (Simulated time: about 30 minutes later) Request bolus (should be successfully injected).
* (Simulated time: about 75 minutes later) Request bolus (should not be injected).
* Turn baseline off.
* Set baseline to 0.10 ml/min.
* Turn baseline on.
* (Simulated time: about 1-3 minutes later) Baseline injection should trigger the hour limit and stop.
* (Simulated time: 10 minutes later) Baseline injections occur every 10 minutes.
* (After the day limit is reached) Reach day limit, stop injections.
* **T5.2 System Limits**
* **Purpose:** To test the system's error handling capabilities and adherence to dosage limits under extreme conditions.
* **Test Input:** Rapid succession of bolus dose requests that approach or exceed the hourly and daily limits, interspersed with attempts to adjust baseline doses.
* **Coverage Item:** System's limit checking logic, error messaging, and recovery from state changes that attempt to violate dosage policies.
* **State Before Execution:** System operating normally with baseline dose active.
* **Expected Output:** The system should deny any bolus requests that exceed limits, provide clear error messages, and maintain accurate record-keeping without crashing or incorrect behavior.
* **Test Result:** Passed.

**Case #2: Maximum Dose Test**

* Set baseline to 0.10 ml/min.
* Set bolus to 0.50 ml/shot.
* Turn baseline on.
* (Simulated time: about 10 minutes later) Trigger hour limit and stop injections.
* Turn baseline off.
* (Simulated time: about 70 minutes later) Request bolus (should be successfully injected).
* Turn baseline on.
* (Simulated time: about 100 minutes later) Request bolus (should not be injected).
* **T5.3 Limit and Reset**
* **Purpose:** To validate the system's overall functionality through a scenario that includes all major features.
* **Test Input:** Starting with adjusting baseline and bolus settings, activating baseline, administering several bolus doses under varying conditions, and observing the cumulative effects on dosage calculations follow by a reset operation..
* **Coverage Item:** System's functional elements, including UI interactions, backend calculations, and reset.
* **State Before Execution:** Initial state post-system reset with no active doses.
* **Expected Output:** The system should adjust doses according to user inputs and safety limits, with all changes reflected in the UI and internal logs without any discrepancies and finally reset the amount to zero.
* **Test Result:** Passed.

**Case #3: Limit and Reset Test**

* Set baseline to 0.10 ml/min.
* Set bolus to 0.50 ml/shot.
* (Upon start) Request bolus (should be successfully injected).
* Request bolus (should be successfully injected).
* Request bolus (should trigger the hour limit and stop injections).
* (Simulated time: about 2 hours later) Turn baseline on.
* (Simulated time: 10 minutes later) Trigger hour limit and stop injections.
* (Simulated time: 70 minutes later) Trigger hour limit and stop injections.
* Reset by doctor.
* Injection limits should be reset.

Model Checking

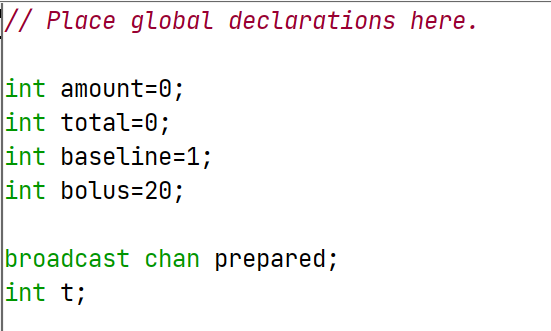
Overview

The Uppaal model for the Painkiller Injection System is constructed to provide a robust simulation and verification framework that ensures the system adheres to critical safety and operational guidelines. This model checks the integrated functionality of the software components designed to manage and control the dosing of painkillers to patients in a clinical setting. Utilizing Uppaal's real-time system model checker, we can simulate the behavior under various conditions and verify important safety properties.

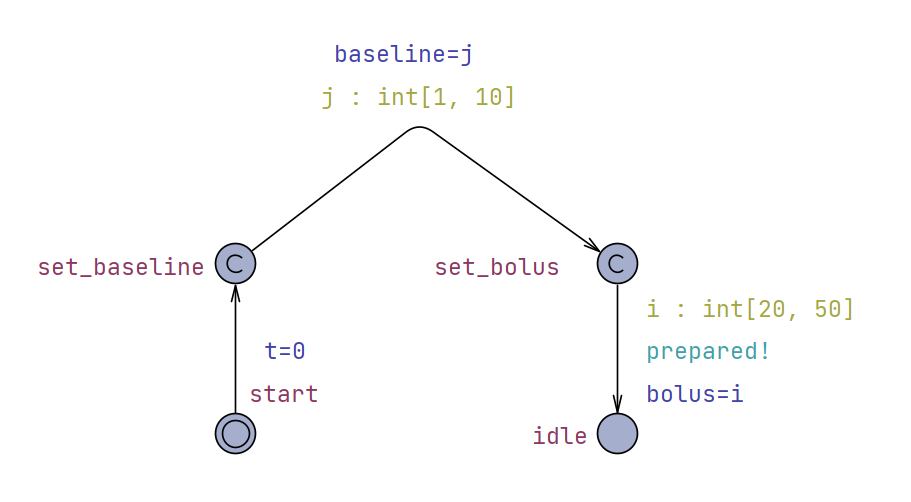
Description of the Uppaal Model Components

The model consists of several primary components, structured to reflect the interactions between the physician and the patient, as well as the system's internal logic handling dosages over time:

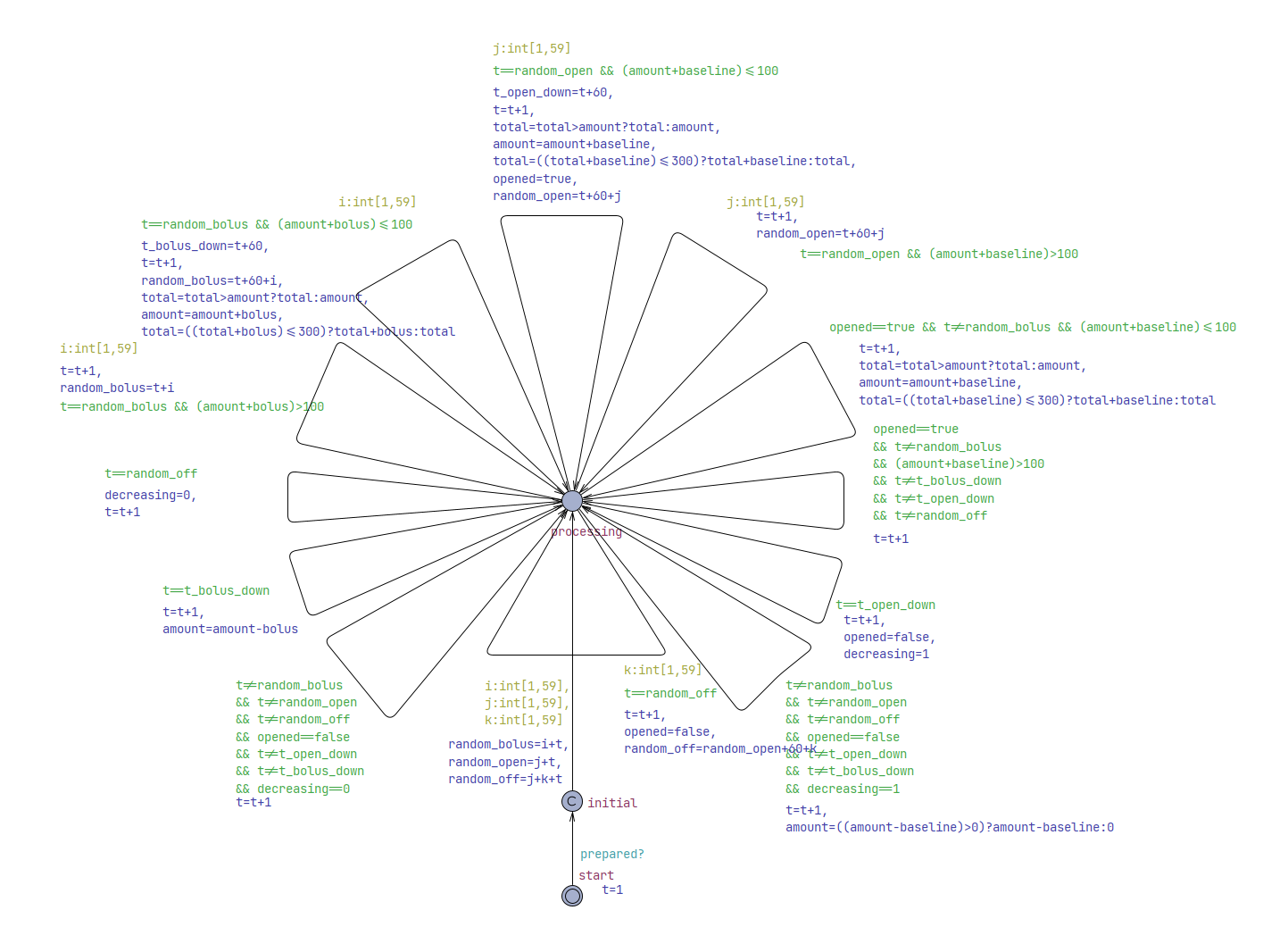
* **Variables and Declarations:**

****

* **int amount = 0;:** Represents the amount of painkiller administered within a current hour, crucial for monitoring hourly dosage limits.
* **int total = 0;:** Accumulates the total dosage given over a 24-hour period, ensuring the daily maximum is not exceeded.
* **int baseline;:** Set by the physician, this variable determines the continuous dosing rate per minute, critical for maintaining a consistent medication flow.
* **int bolus;:** Also set by the physician, it specifies the dosage amount delivered per bolus injection, triggered by patient requests.
* **broadcast chan prepared;:** A synchronization channel that signals when the physician has set the baseline and bolus, allowing the patient's model to proceed with medication administration.
* **int t;:** A time counter, incrementing by one each minute, used to manage and reset dosage calculations on hourly and daily cycles.
* **Physician Template (Control):**

****

* The physician's template features states such as set\_baseline and set\_bolus which are crucial for initializing the dosing parameters. Transitions between these states allow the physician to input the desired medication rates, which are then committed to the system's operational parameters.
* **Patient Template:**

****

* The patient's interactions are modeled to simulate the request and administration of bolus doses. This template includes logic to check against the current dosages (amount and total) before allowing a bolus to be administered, directly influencing patient safety by preventing overdose.

**Working Flow**

The system's operation begins with the physician setting the baseline and bolus parameters. Once these are configured, the patient can interact with the system to receive medication. The model tracks time using the variable t, which increments to simulate minutes passing. This timing mechanism is crucial for resetting the hourly (amount) and daily (total) dosage counters, ensuring adherence to prescribed limits.

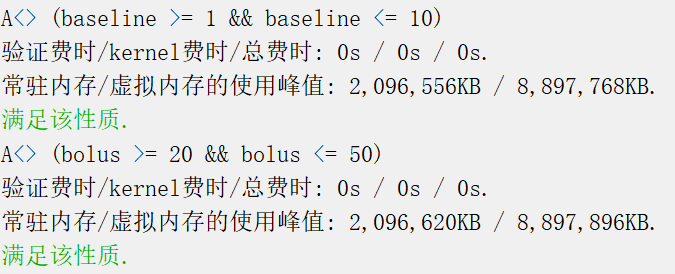
Verification Strategy

The verification of the Painkiller Injection System using Uppaal primarily employs always eventually (A<>) to identify any potential scenarios that could lead to violations of the system's operational constraints. This approach is chosen due to the complexity of the model and the practical limitations of performing exhaustive checks within a reasonable time frame. By focusing on these queries, the verification process effectively verify whether the amount per hour and per day exceeded the requirement to provide strong evidence that the system operates correctly under its intended conditions.



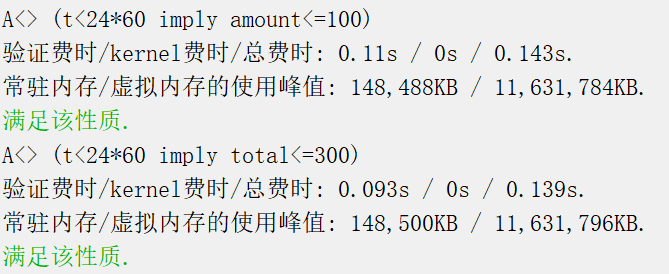
These verification properties provide an assessment of the system's functionality and safety measures. By testing these crucial aspects, the Uppaal model helps to confirm that the Painkiller Injection System is designed and functioning in a manner that upholds the standards of medical safety and operational reliability.

* **Verification of System Setup Functions**
* **Properties:**
* A<> (baseline >= 1 && baseline <= 10)
* A<> (bolus >= 20 && bolus <= 50)
* **Purpose:** These properties assert that the system will eventually set the baseline and bolus within the specified ranges. The ranges are chosen based on the scaled values appropriate for the system, where baseline is the minute-wise medication rate and bolus is the dosage per patient request.
* **Operational Rationale:** The use of A<> (eventually) operator ensures that regardless of the initial conditions or the sequence of operations, the system configurations for baseline and bolus will always end up within the defined safe limits. This is crucial for maintaining consistent and safe medication dosing.
* **Result:** Passed.

****

* **Explanation:** The settings for baseline and bolus are critical control parameters that dictate the regular and on-demand medication rates. By confirming that these parameters are always set within safe bounds, the model verifies that the physician's inputs are correctly captured and enforced by the system. This setup ensures that the system adheres to clinical standards and prevents dosing errors right from the initial configuration phase.
* **Verification of Hourly/Daily Dosage Limit**
* **Properties:**
* A<> (t<24\*60 imply total<=300)
* A<> (t<24\*60 imply amount<=100)
* **Purpose:** These properties are essential to ensure that the painkiller injection system adheres to critical dosage limits set for patient safety. The first property guarantees that the total amount of painkiller delivered within a 24-hour period (equivalent to 1440 minutes) does not exceed 300 units, correlating to a daily limit of 3ml. The second property ensures that the amount of painkiller administered within any given hour does not exceed 100 units, aligning with an hourly limit of 1ml. These limits are designed to prevent overdose and ensure effective pain management without exceeding safe medication thresholds.
* **Operational Rationale:**

The use of the "A<>" (eventually) operator in the Uppaal model checker is crucial for these verifications. It asserts that, irrespective of the system’s state transitions or the timing of injections (baseline or bolus), the system will not exceed the specified dosage limits within the designated time frames. This temporal logic ensures that at any point during the operation, under any sequence of actions taken by the system, the total and hourly dosages remain within safe boundaries. This approach is vital for verifying continuous compliance with medical safety standards throughout the system's operation.



* **Result:** Passed.
* **Explanation:** The defined properties for dosage control in the Uppaal model are directly linked to the safety and efficacy of the painkiller injection system. The total daily limit of 3ml and the hourly limit of 1ml are based on clinical guidelines that prevent the risks associated with overdose.