Discrete-Event Modeling Using DEVS

Assignment 1

SYSC 5104

Player Substitution System (PSS) Model

Youssef Elfaramawy (101366514)

Department of Systems and Computer Engineering

Carleton University

March 2025

PART I - Overview

Real System

For this assignment, I will be modelling the behavior of a player substitution system for a sports team during a game. This system should emulate the decision-making process of a coach when deciding whether to substitute a player in or not; this decision will be based on the player's state and the current game score.

As the game progresses, each player's mental and fatigue level and time metrics (time spent recovering on the bench and total time played) will be monitored. Additionally, the current game score (i.e., team winning, losing, or tied) will be taken into account. The coach will evaluate these factors and then make a decision.

The system's complexity will depend on the substitution logic and the addition of other factors such as hydration level, confidence level, player position, key player, etc. Typically, in sports games, starter players will average more playing time than bench players. However, the scope of this system will focus on two players and not consider the aforementioned factors. If more realism is expected, these factors can still come into play in the future.

Sketch of Model Structure

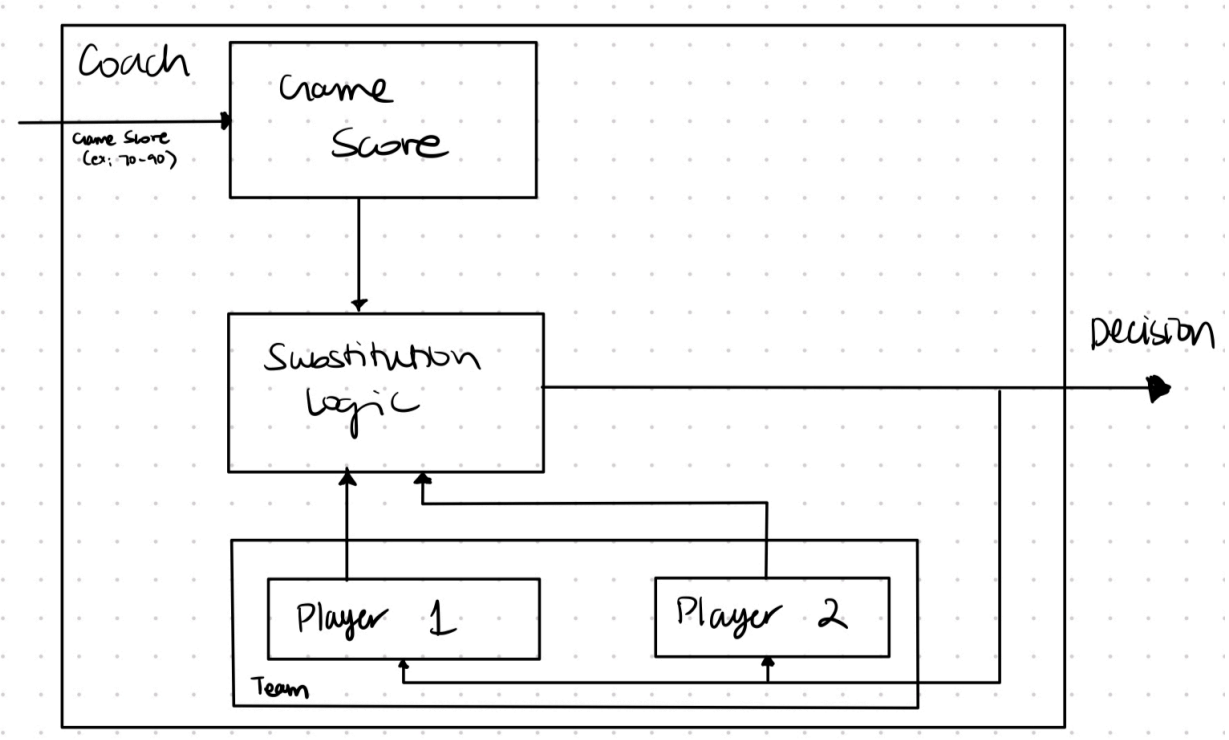

Figure 1: Original Sketch of Coach Model

Behavior of Each Component

- **Coach Model:**
  - Top-level component that contains the submodels below
- **Game Score Model:**
  - Receives inputs of the current game score of team 1 and team 2
  - Output whether team is winning, losing, or tied
- **Player Model:**
  - Monitors player fatigue and mental level
  - Records the following time metrics:
    - time spent recovering on the bench
    - total time played
  - Can either be on the field or bench
  - Receives input of substitution and game score state and updates metrics accordingly
- **Substitution Logic Model:**
  - Contains the decision making logic for making a substitution or not
  - Receives inputs of player and game score status
- **Team Model:**
  - Coupled model that indicates to the player that he has been subbed (routes substitution logic input to the players)

PART II - Behavior and Formal Specification

The atomic models are Game Score, Substitution Logic, Player (duplicated for two players)
The coupled models are Coach (top-level) and Team.

In the following examples, assume 1 time unit is 1 minute (in simulation/code 1 time unit is 1 second)

Player - Behavior

Fatigue

- **Definition:** how tired player is [0, 100]
  - 0: fully rested
  - 100: completely exhausted
- **How it changes: δ_{int}**
  - On the field: increases over time (+1.5 per time unit)
  - On the bench: decreases over time (-1 per time unit)
- **Example:**

    - Player starts on the field
        - Fatigue = 0.0
    - After 10 minutes on the field
        - Fatigue = 15
    - After 20 minutes on the field
        - Fatigue = 30
    - Player is substituted out (goes to bench)
        - Fatigue = 30 (stays constant)
    - After 5 minutes on the bench
        - Fatigue = 25
- **Future improvement:** to replicate real-life performance better, both rates (+/-) should be exponential
    - Get more tired as time goes on
    - Resting becomes worse as time goes (not as effective)

Mental

- **Definition:** player's stress/confidence level [100, 0]
    - 100: not stressed or high confidence
    - 0: very stressed or low confidence
- **How it changes: δ_{int}**
    - On the field: fluctuates randomly to mimic real-life performance
        - Note that this will be removed for testing purposes to keep things deterministic
    - On the bench: slightly increase over time (+1 per unit of time)
- **How it changes: δ_{ext}**
    - Team is losing: decrease over time (-2 per unit of time)
    - Team is winning: increase over time (+2 per unit of time)
    - Team is tied: mental stays constant (no change)
- **Rules for δ_{int} (uniform random number to discrete variate):**
    - Small mistake:
        - Decrease mental by 5 units
        - Probability: 25% per time unit
    - Big mistake:
        - Decrease mental by 10 units
        - Probability: 10% per time unit
    - Score a goal:
        - Increase mental by 15 units
        - Probability: 5% per time unit
    - Assist a goal:
        - Increase mental by 7.5 units

        - Probability: 10% per time unit
    - No event:
        - Mental remains constant
        - Probability: 50% per time unit
- **Example 1 (with random events)**
    - Minute 1 (Team Tied, On Field)
        - Random event: No event
        - Event input: Team Tied → no change
        - Mental State 100 → 100
    - Minute 2 (Team Losing, On Field)
        - Random event: big mistake mistake (25%, -10)
        - Event input: Team Losing → -2 per time unit
        - Total change: -10 - 2 = -12
        - Mental State: 100 → 88
    - Minute 3 (Team Losing, On Field)
        - Random event: Assist a goal (10%, +7.5)
        - Event input: Team Losing → -2 per time unit
        - Total change: +7.5 - 2 = 5.5
        - Mental State: 88 → 93.5
    - Minute 4 (Team Winning, On Field)
        - Random event: No event
        - Event input: Team Winning → +2 per time unit
        - Mental State: 93.5 → 95.5
    - Minute 5 (Team Winning, On Bench)
        - Recovery: +1 per time unit
        - Mental State: 95.5 → 96.5
- **Example 2 (no random events):**
    - Minute 1 (Team Tied, On Field)
        - Event input: Team Tied → no change
        - Mental State 100 → 100
    - Minute 2 (Team Losing, On Field)
        - Event input: Team Losing → -2 per time unit
        - Mental State: 100 → 98
    - Minute 3 (Team Losing, On Field)
        - Event input: Team Losing → -2 per time unit
        - Mental State: 98 → 96
    - Minute 4 (Team Winning, On Field)
        - Event input: Team Winning → +2 per time unit
        - Mental State: 96 → 98
    - Minute 5 (Team Winning, On Bench)

        - Recovery: +1 per time unit
        - Mental State: 98 → 99
- **Future improvement:** to replicate real-life performance better, implement random mental fluctuations according to the rules above, and mimic Example 1 shown above
    - For implementation and testing purposes, the code will reflect closer to the behavior of Example 2

Time

- **Definition:** two different time metrics
    - Time played: total time the player has been on the field
    - Time on bench: the time a player has spent recovering on the bench
- **How it changes: δ_{int}**
    - On the field:
        - Time played: increases over time (+1 per time unit)
    - On the bench
        - Time on bench: increases over time (+1 per time unit)
- **How it changes: δ_{ext}**
    - Player gets subbed in (goes to field):
        - Time bench: resets to 0
    - Player gets subbed out (goes to bench):
        - Time played: remains constant
- **Example:**
    - Player starts on the field
        - Time played: 0
        - Time on bench: 0
    - After 10 minutes on the field
        - Time played: 10
        - Time on bench: 0
    - Player gets substituted out
        - Time played: 10 (remains constant)
        - Time on bench: 0
    - After 5 minutes on the bench
        - Time played: 10
        - Time on bench: 5
    - Player gets substituted in:
        - Time played: 10
        - Time on bench: 0 (resets to 0)
- **Future improvement:** monitor more time variables, such as the last time played, for better decision making

Full Example with all metrics

- 0 Minute: Player starts on field
    - Game state: TIED
    - Fatigue: 0
    - Mental: 100
    - Time played: 0
    - Time on bench: 0
- 10 minutes: ON_FIELD (10 minutes elapsed)
    - Game state: TIED
    - Fatigue: 15
    - Mental: 100
    - Time played: 10
    - Time on bench: 0
- 20 minutes: ON_FIELD (10 minutes elapsed)
    - Game state: TIED
    - Fatigue: 30
    - Mental: 100
    - Time played: 20
    - Time on bench: 0
- At 20 minutes: Player gets substituted out: ON_BENCH
    - Game state: TIED
    - Fatigue: 30
    - Mental: 100
    - Time played: 20
    - Time on bench: 0
- 25 minutes: ON_BENCH (5 minutes elapsed)
    - Game state: TIED
    - Fatigue: 25
    - Mental: 100
    - Time played: 20
    - Time on bench: 5
- At 26 minutes Player gets substituted in: ON_FIELD
    - Game state: TIED
    - Fatigue: 24
    - Mental: 100
    - Time played: 20
    - Time on bench: 0
- 30 minutes: ON_FIELD (4 minutes elapsed)

    - Game state: TIED
    - Fatigue: 30
    - Mental: 100
    - Time played: 24
    - Time on bench: 0
- At 30 minutes, Game State changes to LOSING
    - Game state: LOSING
    - Fatigue: 30
    - Mental: 100
    - Time played: 24
    - Time on bench: 0
- 40 minutes: ON_FIELD (10 minutes elapsed)
    - Game state: LOSING
    - Fatigue: 45
    - Mental: 80
    - Time played: 34
    - Time on bench: 0
- 50 minutes: ON_FIELD (10 minutes elapsed)
    - Game state: LOSING
    - Fatigue: 60
    - Mental: 60
    - Time played: 44
    - Time on bench: 0
- At 50 minutes, Game State changes to WINNING
    - Game state: WINNING
    - Fatigue: 60
    - Mental: 60
    - Time played: 44
    - Time on bench: 0
- At 51 minutes, Player gets substituted out: ON_BENCH
    - Game state: WINNING
    - Fatigue: 61.5
    - Mental: 62
    - Time played: 45
    - Time on bench: 0
- 60 minutes: ON_BENCH (9 minutes elapsed)
    - Game state: WINNING
    - Fatigue: 51.5
    - Mental: 71
    - Time played: 45

- Time on bench: 9

Player (Atomic) - Formal Specification

Player = <X, Y, S, δ_{int}, δ_{ext}, λ, t_a>
X = {(player_out_id $\in \mathbb{N}$, player_in_id $\in \mathbb{N}$), game_state $\in$ {WINNING,LOSING,TIED}}
Y = {player_id $\in \mathbb{N}$, mental $\in$ [0,100], fatigue $\in$ [0,100], phase $\in$ {ON_FIELD, ON_BENCH}, time_played $\in \mathbb{R}^+$, time_on_bench $\in \mathbb{R}^+$}
S = {player_id $\in \mathbb{N}$, mental $\in$ [0,100], fatigue $\in$ [0,100], phase $\in$ {ON_FIELD, ON_BENCH}, time_played $\in \mathbb{R}^+$, time_on_bench $\in \mathbb{R}^+$, game_state $\in$ {WINNING,LOSING,TIED}, sigma $\in \mathbb{R}^+$}

δ_{int}(s) = {
    case phase:
        ON_FIELD:
            fatigue = state.fatigue + 1.5 * sigma

            if game_state == WINNING:
                mental += 2.0 * sigma
            elif game_state == LOSING:
                mental -= 2.0 * sigma

            time_played += state.sigma
        ON_BENCH:
            fatigue = state.fatigue - 1.0 * sigma

            mental = mental + 1.0 * sigma

            time_on_bench += sigma

            state.sigma = 1.0
    return *this
}
δ_{ext}(s, e, x) = {
    if not empty(x.ids):
        player_out_id, player_in_id = get_substitution_ids(x.ids)
        if player_out_id == player_id:
            phase = ON_BENCH
        elif player_in_id == player_id:
            phase = ON_FIELD

```
                time_on_bench = 0

        if not empty(x.game):
                game_state = get_game_state(x.game)

        state.sigma -= e
        return *this
}
λ(s) = {
        return {
                player_id,
                mental,
                fatigue,
                phase,
                time_played,
                time_on_bench
        }
}

t_a(s): {
        return sigma
}
```

<u>Game Score - Behavior</u>

Game State

- **Definition:** represents the current state of the game based on the scores of two teams (favors team 1)
    - WINNING: Team 1 is leading
    - LOSING: Team 1 is trailing
    - TIED: both teams have the same score
- **How it changes: δ_{ext}**
    - Team 1 Score > Team 2 Score: WINNING
    - Team 1 Score < Team 2 Score: LOSING
    - Team 1 Score = Team 2 Score: TIED
- **How it changes: δ_{int}**
    - No internal event occurs
- **Example:**
    - Minute 1(Team 1: 0 | Team 2: 0)
        - TIED

- Minute 2 (Team 1: 1 | Team 2: 0)
  - WINNING
- Minute 3 (Team 1: 1 | Team 2: 1)
  - TIED
- Minute 4 (Team 1: 1 | Team 2: 2)
  - LOSING

Game Score (Atomic) - Formal Specification

Game Score = $<X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a>$
$X = \{team\_1\_score \in \mathbb{N}, team\_2\_score \in \mathbb{N}\}$
$Y = \{game\_state \in \{WINNING,LOSING,TIED\}\}$
$S = \{team\_1\_score \in \mathbb{N}, team\_2\_score \in \mathbb{N}, game\_state \in \{WINNING,LOSING,TIED\}, sigma \in \mathbb{R}^+\}$
$\delta_{int}(s)$ = {
    sigma = 1.0
    return *this
}

$\delta_{ext}(s, e, x)$ = {
    if not empty(x.team_1_score):
        team_1_score = get_team_1_score(x.team_1_score)
    if not empty(x.team_2_score):
        team_2_score = get_team_2_score(x.team_2_score)

    if team_1_score > team_2_score:
        game_state = WINNING
    elif team_1_score < team_2_score:
        game_state = LOSING
    else:
        game_state = TIED

    sigma -= e
    return *this
}

$\lambda(s)$ = {
    return game_state
}

$t_a(s)$: {
    return sigma
}

Substitution Logic - Behavior

Substitution Command

- **Definition:** determines when to substitute players based on their metrics (fatigue, mental, time played, time on bench) and the current game state
    - player_out_id: id of the player being subbed out (goes to bench)
    - player_in_id: id of the player being subbed in (goes to field)
- **How it works:** uses weighted decision making, each metric is normalized and then assigned a weight. Based on the game state (WINNING, LOSING, TIED), the weights and substitution threshold dynamically change
    - **Field Players:**
        - **Metrics and Normalization:**
            - Fatigue:
                - Range: 0 to 100
                - Normalization: fatigue/100
                    - 1 = maximum fatigue
            - Mental:
                - Range: 0 to 100
                - Normalization: (mental - 100)/100
                    - 1 = maximum stress
            - Time played:
                - Range: 0 to ∞
                - Normalization: tanh(time_played/scaling factor)
                    - Asymptotic to 1 as it increases based on scaling factor
                    - Scaling factor changes based on the type of game
                        - Ex: soccer (90/45 minutes) or basketball (48/12 minutes)
            - Time on bench:
                - Range: 0 to ∞
                - Normalization: must pass minimum rest period
                    - Time on bench > minimum rest → eligible for substitution
        - **Dynamic Weights and Thresholds for Field Players:**
            - Baseline threshold: 1.5
                - Losing: 1.5 → 1.2

    - Tied: 1.5 → 1.5
    - Winning: 1.5 → 1.8

Game State	Fatigue Weight	Mental Weight	Time Played Weight	Threshold
Losing	1.2x	1.5x	1.1x	-20% (easier to sub)
Tied	1x	1x	1x	Baseline
Winning	1.2x	0.7x	1x	+20% (harder to sub)

    - **Bench Players:**
        - **Thresholds:**
            - Fatigue must be below 40
            - Mental must be at least 60
            - Time on bench must be at least 10
- **How it changes: δ_{ext}**
    - Updates player metrics/game state when receiving inputs at the state level
- **How it changes: δ_{int}**
    - No internal event occurs
- **Example 1:**
    - Game State: LOSING
        - Weights: 1.2x fatigue, 1.5x mental, 1.1x time played
    - Player 1 Metrics (ON FIELD):
        - Mental: 60
            - Normalized: (100-60)/100 = 0.4
        - Fatigue: 40
            - Normalized: 40/100 = 0.4
        - Time Played: 20
            - Normalized: tanh(20/45) = 0.417
        - Time on Bench: 0
            - Normalized: N/A → field player
        - **Score:** 1.2(0.4) + 1.5(0.4) + 1.1(0.417) = 1.5387 > 1.2 (threshold) → can be substituted out
    - Player 2 Metrics (ON BENCH):
        - Mental: 80
            - 80 >= 60 (mental threshold for bench player)
        - Fatigue: 10
            - 10 <= 40 (fatigue threshold for bench player)
        - Time Played: 0
            - N/A → bench player
        - Time on Bench: 20

            - 20 >= 10 (bench time threshold for bench player)
        - **Passes all checks** → can be substituted in
    - **Both players pass** → can make the substitution
        - Player 1 OUT
        - Player 2 IN
- **Example 2:**
    - Game State: WINNING
        - Weights: 1.2x fatigue, 0.7x mental, 1x time played
    - Player 1 Metrics (ON BENCH):
        - Mental: 80
            - 80 >= 60 (mental threshold for bench player)
        - Fatigue: 20
            - 20 <= 40 (fatigue threshold for bench player)
        - Time Played: 20
            - N/A → bench player
        - Time on Bench: 30
            - 30 >= 10 (bench time threshold for bench player)
        - **Passes all checks** → can be substituted in
    - Player 2 Metrics (ON FIELD):
        - Mental: 40
            - Normalized: (100-40)/100 = 0.6
        - Fatigue: 70
            - Normalized: 70/100 = 0.7
        - Time Played: 30
            - Normalized: tanh(30/45) = 0.5827
        - Time on Bench: 0
            - Normalized: N/A → field player
        - **Score:** 1.2(0.7) + 0.7(0.6) + 1(0.5827) = 1.8417 > 1.8 (threshold) → can be substituted out
    - **Both players pass** → can make the substitution
        - Player 1 IN
        - Player 2 OUT

Substitution Logic (Atomic) - Formal Specification

Substitution Logic = $\langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$
X = {(player_id $\in \mathbb{N}$, mental $\in [0,100]$, fatigue $\in [0,100]$, phase $\in$ {ON_FIELD, ON_BENCH}, time_played $\in \mathbb{R}^+$, time_on_bench $\in \mathbb{R}^+$), (player_id $\in \mathbb{N}$, mental $\in [0,100]$, fatigue $\in [0,100]$, phase $\in$ {ON_FIELD, ON_BENCH}, time_played $\in \mathbb{R}^+$, time_on_bench $\in \mathbb{R}^+$), game_state $\in$ {WINNING,LOSING,TIED}}

$Y = \{(player\_out\_id \in \mathbb{N}, player\_in\_id \in \mathbb{N})\}$
$S = \{(player\_id, mental, fatigue, phase, time\_played, time\_on\_bench), (player\_id, mental, fatigue, phase, time\_played, time\_on\_bench), game\_state \in \{WINNING,LOSING,TIED\}, sigma \in \mathbb{R}^{+}\}$
$\delta_{int}(s) = \{$

```
        sigma = 1.0
        return *this;
}
```

$\delta_{ext}(s, e, x) = \{$

```
        if not empty(x.players):
                player_1_metrics, state.player_2_metrics = get_player_metrics(x.players)
        if not empty(x.game_state):
                game_state = get_game_state(x.game_state)

        state.sigma -= e
        return *this
}

λ(s) = {
        weights = getWeights(game_state)
        threshold = getThreshold(game_state)

        if player_1_metrics.phase == BENCH:
                bench_ready = checkIfBenchPlayerReady(player_1_metrics)
                norm_metrics = normalizePlayerMetrics(player_2_metrics)
                score = calculateScore(norm_metrics, weights)

                if (score > threshold and bench_ready):
                        return {
                                player_out_id = player 2.id,
                                player_in_id = player 1.id
                        }
        elif player_2.metrics.phase == BENCH:
                bench_ready = checkIfBenchPlayerReady(player_2_metrics)
                norm_metrics = normalizePlayerMetrics(player_1_metrics)
                score = calculateScore(norm_metrics, weights)

                if (score > threshold and bench_ready):
                        return {
                                player_out_id = player 1.id,
```

```
                                player_in_id = player 2.id
                        }
        return None
}

t_a(s): {
        return sigma
}
```

Team (Coupled) - Formal Specification

Team = <X, Y, M, EIC, EOC, IC, select>

X = {(player_out_id ∈ ℕ, player_in_id ∈ ℕ), game_state ∈ {WINNING,LOSING,TIED}}

Y = {(player_id ∈ ℕ, mental ∈ [0,100], fatigue ∈ [0,100], phase ∈ {ON_FIELD, ON_BENCH}, time_played ∈ $\mathbb{R}^+$, time_on_bench ∈ $\mathbb{R}^+$), (player_id ∈ ℕ, mental ∈ [0,100], fatigue ∈ [0,100], phase ∈ {ON_FIELD, ON_BENCH}, time_played ∈ $\mathbb{R}^+$, time_on_bench ∈ $\mathbb{R}^+$)}

M = {Player1, Player2}

EIC = {(self.(player_out_id, player_in_id), Player1.(player_out_id, player_in_id)), (self.(player_out_id, player_in_id), Player2.(player_out_id, player_in_id)), (self.game_state, Player1.game_state), (self.game_state, Player2.game_state)}

EOC = {(Player1.(player_id, mental, fatigue, phase, time_played, time_on_bench), self.(player_id, mental, fatigue, phase, time_played, time_on_bench)), (Player2.(player_id, mental, fatigue, phase, time_played, time_on_bench), self.(player_id, mental, fatigue, phase, time_played, time_on_bench))}

IC = ∅

Coach (Coupled) - Top Level - Formal Specification

Coach = <X, Y, M, EIC, EOC, IC, select>

X = ∅

Y = ∅

M = {GameScore, Team, SubstitutionLogic}

EIC = ∅

EOC = ∅

IC = {(GameScore.game_state, SubstitutionLogic.game_state), (GameScore.game_state, Team.game_state), (Team.Player1.(player_id, mental, fatigue, phase, time_played, time_on_bench), SubstitutionLogic.(player_id, mental, fatigue, phase, time_played, time_on_bench)), (Team.Player2.(player_id, mental, fatigue, phase, time_played, time_on_bench), SubstitutionLogic.(player_id, mental, fatigue, phase, time_played, time_on_bench)), (SubstitutionLogic.(player_out_id, player_in_id), Team.(player_out_id, player_in_id))}

PART III - Testing Strategies and Documentation

Updated Sketch of Model Structure

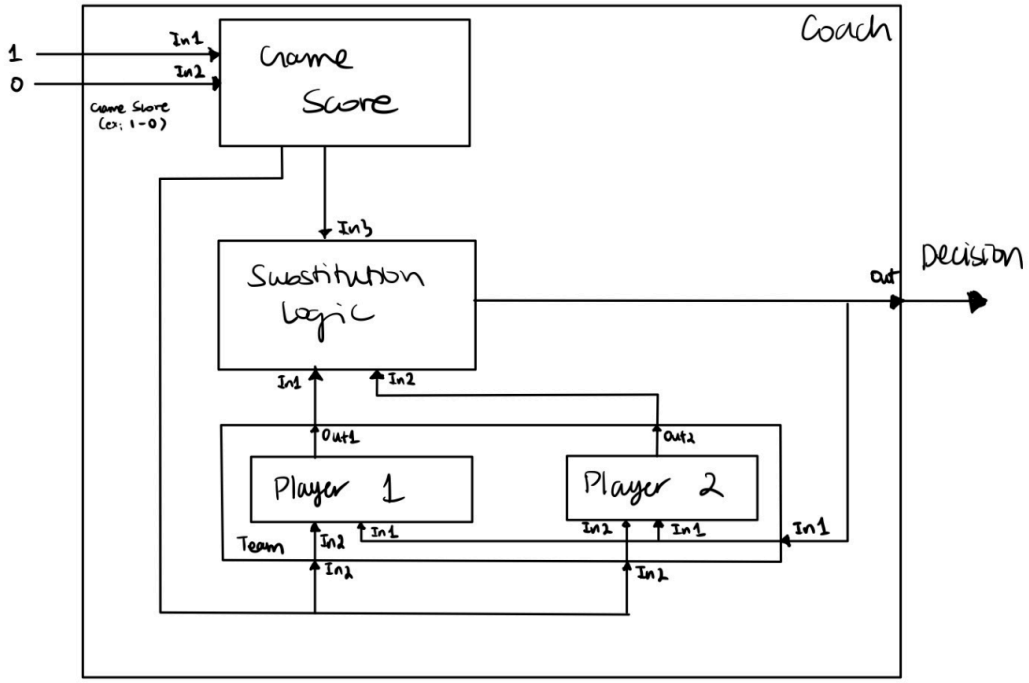

Figure 2: Updated Sketch of Coach Model

Compared to the original sketch of the coach model structure, the team coupled model now has an external output coupling with the **Player** model and receives an additional external input from the **GameScore** model. This was done to ensure that Players receive *game_state* updates, which could play a factor in influencing their mental levels.

Atomic Model Validation

For each of the following atomic models, testing was done by feeding each model's input a *.txt* file that followed closely the examples proposed in Part II of the report. Each atomic component was treated as a black box, and test cases were created with different inputs to check their correctness. If the observed output matches the expected results, then the atomic model is validated.

GameScore

Given team 1 and team 2's score, this component should output a state accordingly (WINNING, LOSING, or TIED). Note that the model favors team 1.

team_1_score_test.txt

0.0 0
1.0 0
2.0 1
3.0 1
4.0 1

team_2_score_test.txt
0.0 0
1.0 0
2.0 0
3.0 1
4.0 2

Expected:

- Time 1: TIED
- Time 2: WINNING
- Time 3: TIED
- Time 4: LOSING

Observed:
0;3;game_score_model;;{game state: TIED, team 1 score: 0, team 2 score: 0}
1;3;game_score_model;;{game state: TIED, team 1 score: 0, team 2 score: 0}
2;3;game_score_model;;{game state: WINNING, team 1 score: 1, team 2 score: 0}
3;3;game_score_model;;{game state: TIED, team 1 score: 1, team 2 score: 1}
4;3;game_score_model;;{game state: LOSING, team 1 score: 1, team 2 score: 2}

The observed matches the expected ⇒ ✅

Player
Given game state and substitution command inputs, player metrics should fluctuate accordingly to the proposed behavior in Part II.

game_state_test.txt
0.0 2
30.0 1
50.0 0
substitution_test.txt
20 1 2
26 2 1
51 1 2

Expected:

- At minute 0 and onwards:
    - **Fatigue** → increases by +1.5 per time unit
    - **Mental** → unaffected (initial game state, TIED, has no effect)
    - **Time played** → increases by +1 per time unit
    - **Time bench** → stays 0
- At minute 20 (player gets substituted out):
    - **Fatigue** → decreases by -1. per time unit
    - **Mental** → remains constant (max 100)
    - **Time played** → remains constant
    - **Time bench** → increases by +1 per time unit
- At minute 26 (player gets substituted back in):
    - **Fatigue** → increases again by +1.5 per time unit
    - **Mental** → remains constant
    - **Time played** → increases again by +1 per time unit
    - **Time bench** → resets to 0
- At minute 30 (game state changes from TIED to LOSING):
    - **Fatigue** → keeps increasing by +1.5 per time unit
    - **Mental** → starts decreasing by -2 per time unit
    - **Time played** → keeps increasing by +1 per time unit
    - **Time bench** → remains 0
- At minute 50 (game state change from LOSING to WINNING):
    - **Fatigue** → keeps increasing by +1.5 per time unit
    - **Mental** → starts increasing by +2 per time unit
    - **Time played** → keeps increasing by +1 per time unit
    - **Time bench** → remains 0
- At minute 51, player gets substituted out. Onwards, we should expect:
    - **Fatigue** → starts decreasing by -1 per time unit
    - **Mental** → starts increasing by +1 per time unit
    - **Time played** → remains constant
    - **Time bench** → starts increasing by +1 per time unit

Observed:

0;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 0, phase: ON_FIELD, time played: 0, time on bench: 0}
1;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 1.5, phase: ON_FIELD, time played: 1, time on bench: 0}

2;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 3, phase: ON_FIELD, time played: 2, time on bench: 0}
3;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 4.5, phase: ON_FIELD, time played: 3, time on bench: 0}
4;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 6, phase: ON_FIELD, time played: 4, time on bench: 0}
5;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 7.5, phase: ON_FIELD, time played: 5, time on bench: 0}
6;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 9, phase: ON_FIELD, time played: 6, time on bench: 0}
7;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 10.5, phase: ON_FIELD, time played: 7, time on bench: 0}
8;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 12, phase: ON_FIELD, time played: 8, time on bench: 0}
9;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 13.5, phase: ON_FIELD, time played: 9, time on bench: 0}
10;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 15, phase: ON_FIELD, time played: 10, time on bench: 0}
11;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 16.5, phase: ON_FIELD, time played: 11, time on bench: 0}
12;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 18, phase: ON_FIELD, time played: 12, time on bench: 0}
13;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 19.5, phase: ON_FIELD, time played: 13, time on bench: 0}
14;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 21, phase: ON_FIELD, time played: 14, time on bench: 0}
15;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 22.5, phase: ON_FIELD, time played: 15, time on bench: 0}
16;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 24, phase: ON_FIELD, time played: 16, time on bench: 0}
17;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 25.5, phase: ON_FIELD, time played: 17, time on bench: 0}
18;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 27, phase: ON_FIELD, time played: 18, time on bench: 0}
19;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 28.5, phase: ON_FIELD, time played: 19, time on bench: 0}
20;2;substitution_file;out;{player_out_id: 1, player_in_id: 2}
20;2;substitution_file;;6
20;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 30, phase: ON_BENCH, time played: 20, time on bench: 0}

21;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 29, phase: ON_BENCH, time played: 20, time on bench: 1}
22;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 28, phase: ON_BENCH, time played: 20, time on bench: 2}
23;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 27, phase: ON_BENCH, time played: 20, time on bench: 3}
24;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 26, phase: ON_BENCH, time played: 20, time on bench: 4}
25;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 25, phase: ON_BENCH, time played: 20, time on bench: 5}
26;2;substitution_file;out;{player_out_id: 2, player_in_id: 1}
26;2;substitution_file;;25
26;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 24, phase: ON_FIELD, time played: 20, time on bench: 0}
27;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 25.5, phase: ON_FIELD, time played: 21, time on bench: 0}
28;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 27, phase: ON_FIELD, time played: 22, time on bench: 0}
29;3;player_model;;{player_id:1, game state: TIED, mental: 100, fatigue: 28.5, phase: ON_FIELD, time played: 23, time on bench: 0}
30;1;game_state_file;out;LOSING
30;1;game_state_file;;20
30;3;player_model;;{player_id:1, game state: LOSING, mental: 100, fatigue: 30, phase: ON_FIELD, time played: 24, time on bench: 0}
31;3;player_model;;{player_id:1, game state: LOSING, mental: 98, fatigue: 31.5, phase: ON_FIELD, time played: 25, time on bench: 0}
32;3;player_model;;{player_id:1, game state: LOSING, mental: 96, fatigue: 33, phase: ON_FIELD, time played: 26, time on bench: 0}
33;3;player_model;;{player_id:1, game state: LOSING, mental: 94, fatigue: 34.5, phase: ON_FIELD, time played: 27, time on bench: 0}
34;3;player_model;;{player_id:1, game state: LOSING, mental: 92, fatigue: 36, phase: ON_FIELD, time played: 28, time on bench: 0}
35;3;player_model;;{player_id:1, game state: LOSING, mental: 90, fatigue: 37.5, phase: ON_FIELD, time played: 29, time on bench: 0}
36;3;player_model;;{player_id:1, game state: LOSING, mental: 88, fatigue: 39, phase: ON_FIELD, time played: 30, time on bench: 0}
37;3;player_model;;{player_id:1, game state: LOSING, mental: 86, fatigue: 40.5, phase: ON_FIELD, time played: 31, time on bench: 0}
38;3;player_model;;{player_id:1, game state: LOSING, mental: 84, fatigue: 42, phase: ON_FIELD, time played: 32, time on bench: 0}

39;3;player_model;;{player_id:1, game state: LOSING, mental: 82, fatigue: 43.5, phase: ON_FIELD, time played: 33, time on bench: 0}
40;3;player_model;;{player_id:1, game state: LOSING, mental: 80, fatigue: 45, phase: ON_FIELD, time played: 34, time on bench: 0}
41;3;player_model;;{player_id:1, game state: LOSING, mental: 78, fatigue: 46.5, phase: ON_FIELD, time played: 35, time on bench: 0}
42;3;player_model;;{player_id:1, game state: LOSING, mental: 76, fatigue: 48, phase: ON_FIELD, time played: 36, time on bench: 0}
43;3;player_model;;{player_id:1, game state: LOSING, mental: 74, fatigue: 49.5, phase: ON_FIELD, time played: 37, time on bench: 0}
44;3;player_model;;{player_id:1, game state: LOSING, mental: 72, fatigue: 51, phase: ON_FIELD, time played: 38, time on bench: 0}
45;3;player_model;;{player_id:1, game state: LOSING, mental: 70, fatigue: 52.5, phase: ON_FIELD, time played: 39, time on bench: 0}
46;3;player_model;;{player_id:1, game state: LOSING, mental: 68, fatigue: 54, phase: ON_FIELD, time played: 40, time on bench: 0}
47;3;player_model;;{player_id:1, game state: LOSING, mental: 66, fatigue: 55.5, phase: ON_FIELD, time played: 41, time on bench: 0}
48;3;player_model;;{player_id:1, game state: LOSING, mental: 64, fatigue: 57, phase: ON_FIELD, time played: 42, time on bench: 0}
49;3;player_model;;{player_id:1, game state: LOSING, mental: 62, fatigue: 58.5, phase: ON_FIELD, time played: 43, time on bench: 0}
50;1;game_state_file;out;WINNING
50;1;game_state_file;;inf
50;3;player_model;;{player_id:1, game state: WINNING, mental: 60, fatigue: 60, phase: ON_FIELD, time played: 44, time on bench: 0}
51;2;substitution_file;out;{player_out_id: 1, player_in_id: 2}
51;2;substitution_file;;inf
51;3;player_model;;{player_id:1, game state: WINNING, mental: 62, fatigue: 61.5, phase: ON_BENCH, time played: 45, time on bench: 0}
52;3;player_model;;{player_id:1, game state: WINNING, mental: 63, fatigue: 60.5, phase: ON_BENCH, time played: 45, time on bench: 1}
53;3;player_model;;{player_id:1, game state: WINNING, mental: 64, fatigue: 59.5, phase: ON_BENCH, time played: 45, time on bench: 2}
54;3;player_model;;{player_id:1, game state: WINNING, mental: 65, fatigue: 58.5, phase: ON_BENCH, time played: 45, time on bench: 3}
55;3;player_model;;{player_id:1, game state: WINNING, mental: 66, fatigue: 57.5, phase: ON_BENCH, time played: 45, time on bench: 4}
56;3;player_model;;{player_id:1, game state: WINNING, mental: 67, fatigue: 56.5, phase: ON_BENCH, time played: 45, time on bench: 5}

57;3;player_model;;{player_id:1, game state: WINNING, mental: 68, fatigue: 55.5, phase: ON_BENCH, time played: 45, time on bench: 6}
58;3;player_model;;{player_id:1, game state: WINNING, mental: 69, fatigue: 54.5, phase: ON_BENCH, time played: 45, time on bench: 7}
59;3;player_model;;{player_id:1, game state: WINNING, mental: 70, fatigue: 53.5, phase: ON_BENCH, time played: 45, time on bench: 8}
60;3;player_model;;{player_id:1, game state: WINNING, mental: 71, fatigue: 52.5, phase: ON_BENCH, time played: 45, time on bench: 9}

The observed matches the expected ⇒ ✅

SubstitutionLogic
Given two player metrics .txt files, the substitution will use threshold/weighted decision making to evaluate whether a substitution will be made. Weights will vary based on the game state input .txt file. The expected behavior should mimic the logic proposed in Part II.

game_state_test.txt
20.0 1
30.0 2
50.0 0

player_1_test.txt
20.0 1 60.0 40.0 0 20.0 0.0
30.0 1 70.0 30.0 1 20.0 10.0
50.0 1 80.0 20.0 1 20.0 30.0

player_2_test.txt
20.0 2 80.0 10.0 1 0.0 20.0
30.0 2 40.0 75.0 0 10.0 0.0
50.0 2 40.0 70.0 0 30.0 0.0

Expected:

- **Minute 20:**
  - Game State: LOSING
    - Weights: 1.2x fatigue, 1.5x mental, 1.1x time played
    - Threshold: 1.2
  - Player 1 Metrics (ON FIELD):
    - Mental: 60 → (100-60)/100 = 0.4
    - Fatigue: 40 → 40/100 = 0.4
    - Time Played: 20 → tanh(20/45) = 0.417

        - Time on Bench: 0 → N/A (field player)
        - **Score:** 1.2(0.4) + 1.5(0.4) + 1.1(0.417) = 1.5387 > 1.2 ✓
        - **Passes threshold** → can be substituted out
    - Player 2 Metrics (ON BENCH):
        - Mental: 80 → 80 >= 60 ✓
        - Fatigue: 10 → 10 <= 40 ✓
        - Time Played: 0 → N/A (bench player)
        - Time on Bench: 20 → 20 >= 10
        - **Passes all checks** → can be substituted in
    - **Both players pass** → can make the substitution
        - Player 1 OUT
        - Player 2 IN
- **Minute 30:**
    - Game State: TIED
        - Weights: 1x fatigue, 1x mental, 1x time played
        - Threshold: 1.5
    - Player 1 Metrics (ON BENCH):
        - Mental: 70 → 70 >= 60 ✓
        - Fatigue: 30 → 30 <= 40 ✓
        - Time Played: 20 → N/A (bench player)
        - Time on Bench: 10 → 10 >= 10 ✓
        - **Passes all checks** → can be substituted in
    - Player 2 Metrics (ON FIELD):
        - Mental: 40 → (100-40)/100 = 0.6
        - Fatigue: 75 → 75/100 = 0.75
        - Time Played: 10 → tanh(10/45) = 0.2186
        - Time on Bench: 0 → N/A (field player)
        - **Score:** 1(0.75) + 1(0.6) + 1(0.2186) = 1.5686 > 1.5 ✓
        - **Passes threshold** → can be substituted out
    - **Both players pass** → can make the substitution
        - Player 1 IN
        - Player 2 OUT
- **Minute 50:**
    - Game State: WINNING
        - Weights: 1.2x fatigue, 0.7x mental, 1x time played
        - Threshold: 1.8
    - Player 1 Metrics (ON BENCH):
        - Mental: 80 → 80 >= 60 ✓
        - Fatigue: 20 → 20 <= 40 ✓
        - Time Played: 20 → N/A (bench player)

        - Time on Bench: 30 → 30 >= 10 ✓
        - **Passes all checks** → can be substituted in
    - Player 2 Metrics (ON FIELD):
        - Mental: 40 → (100-40)/100 = 0.6
        - Fatigue: 70 → 70/100 = 0.7
        - Time Played: 30 → tanh(30/45) = 0.5827
        - Time on Bench: 0 → N/A (field player)
        - **Score:** 1.2(0.7) + 0.7(0.6) + 1(0.5827) = 1.8417 > 1.8 ✓
        - **Passes threshold** → can be substituted out
    - **Both players pass** → can make the substitution
        - Player 1 IN
        - Player 2 OUT

Observed:

0;4;substitution_logic_model;;{game state: TIED, player 1: {player_id:0, mental: 100, fatigue: 0, phase: ON_BENCH, time played: 0, time on bench: 0}, player 2: {player_id:0, mental: 100, fatigue: 0, phase: ON_BENCH, time played: 0, time on bench: 0}}
1;4;substitution_logic_model;;{game state: TIED, player 1: {player_id:0, mental: 100, fatigue: 0, phase: ON_BENCH, time played: 0, time on bench: 0}, player 2: {player_id:0, mental: 100, fatigue: 0, phase: ON_BENCH, time played: 0, time on bench: 0}}
.
.
.
20;1;game_state_file;out;LOSING
20;2;player_2_file;out;{player_id:2, mental: 80, fatigue: 10, phase: ON_BENCH, time played: 0, time on bench: 20}
20;3;player_1_file;out;{player_id:1, mental: 60, fatigue: 40, phase: ON_FIELD, time played: 20, time on bench: 0}
20;4;substitution_logic_model;;{game state: LOSING, player 1: {player_id:1, mental: 60, fatigue: 40, phase: ON_FIELD, time played: 20, time on bench: 0}, player 2: {player_id:2, mental: 80, fatigue: 10, phase: ON_BENCH, time played: 0, time on bench: 20}}
21;4;substitution_logic_model;substitution_out;{player_out_id: 1, player_in_id: 2}
.
.
.
30;1;game_state_file;out;TIED
30;2;player_2_file;out;{player_id:2, mental: 40, fatigue: 75, phase: ON_FIELD, time played: 10, time on bench: 0}
30;3;player_1_file;out;{player_id:1, mental: 70, fatigue: 30, phase: ON_BENCH, time played: 20, time on bench: 10}

30;4;substitution_logic_model;;{game state: TIED, player 1: {player_id:1, mental: 70, fatigue: 30, phase: ON_BENCH, time played: 20, time on bench: 10}, player 2: {player_id:2, mental: 40, fatigue: 75, phase: ON_FIELD, time played: 10, time on bench: 0}}
31;4;substitution_logic_model;substitution_out;{player_out_id: 2, player_in_id: 1}
.
.
.
50;1;game_state_file;out;WINNING
50;2;player_2_file;out;{player_id:2, mental: 40, fatigue: 70, phase: ON_FIELD, time played: 30, time on bench: 0}
50;3;player_1_file;out;{player_id:1, mental: 80, fatigue: 20, phase: ON_BENCH, time played: 20, time on bench: 30}
50;4;substitution_logic_model;;{game state: WINNING, player 1: {player_id:1, mental: 80, fatigue: 20, phase: ON_BENCH, time played: 20, time on bench: 30}, player 2: {player_id:2, mental: 40, fatigue: 70, phase: ON_FIELD, time played: 30, time on bench: 0}}
51;4;substitution_logic_model;substitution_out;{player_out_id: 2, player_in_id: 1}

The observed matches the expected ⇒ ✅

Coupled Model Integration Testing

For **Team** and **Coach** coupled models, integration testing was done at the top level to ensure that all atomic components worked together correctly. The focus was on verifying:

- Each atomic component's output matched expected behavior
- State updates between models (internal coupling) as inputs were correctly propagated
- All atomic components received inputs without any unintended overrides or unexpected behavior

One key issue identified was the lack of an atomic component tp generate scores for both teams. To address this, external input *.txt* files were used for simulation purposes.

team_1_score_test.txt
0.0 0
20.0 0
30.0 0
50.0 2

team_2_score_test.txt
0.0 0
20.0 1
30.0 1
50.0 1

Verification

To verify the correctness of the **Player Substitution System**:

- Clone the project from github
- Compile and run the simulation
- Compare the output logs/CSV file against expected behavior outlined above

Link to the repository

https://github.com/yelfaram/Cadmium-Player-Substitution-System