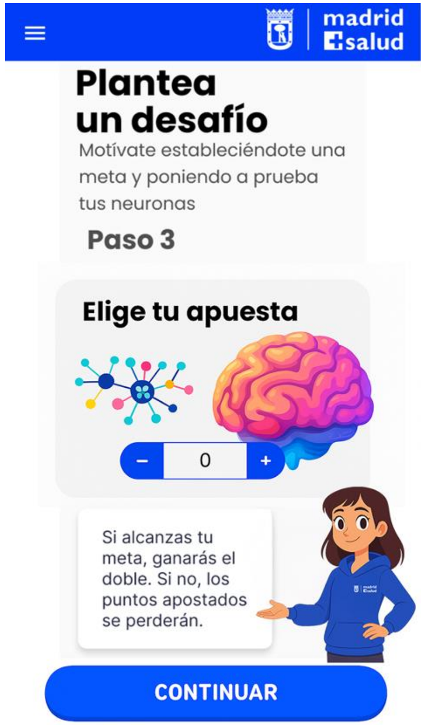
Motiva Ontology: A domain model for motivational assistants in Personal Development Applications

Walkthrough and review

Introduction

- What is a **Personal Development Application (PA)**
  - Application that supports users in improving habits, skills and well being
  - Provides interactive lessons or content around a certain personal development domain
- **Users often lose motivation and discipline over time**
- What does the **Motivational Assistant (MA)** do
  - Helps the user stay engaged and motivated by providing different gamification mechanics
  - Uses personalized messages and features like progress tracking, challenges and rewards
  - Keeps a user model containing:
    - State (dynamic)
    - Traits (static)



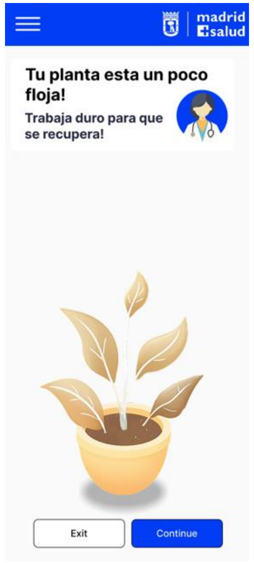
Plantea un desafío

Motívate estableciéndote una meta y poniendo a prueba tus neuronas

Paso 3

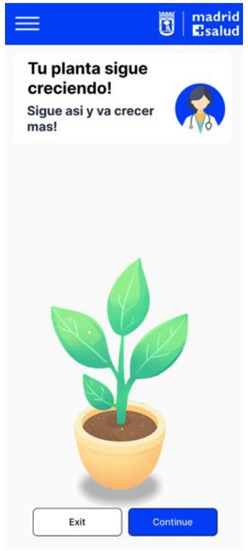
Elige tu apuesta

Si alcanzas tu meta, ganarás el doble. Si no, los puntos apostados se perderán.



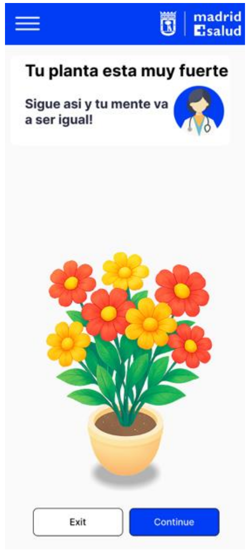
Tu planta esta un poco floja!

Trabaja duro para que se recupera!



Tu planta sigue creciendo!

Sigue asi y va crecer mas!



Tu planta esta muy fuerte

Sigue asi y tu mente va a ser igual!

Motivational Assistant Concept

Core Pipeline: *measure → decide → deliver*

Purpose

Maintain engagement and habit formation through adaptive feedback without overloading user.

Guardrails:

- Temporal smoothing for consistent adaptation
- Provenance ensures continuity with baseline profiles
- Deactivate mechanics with low observed usage despite trait fit
- Continuous recalibration from recent metrics

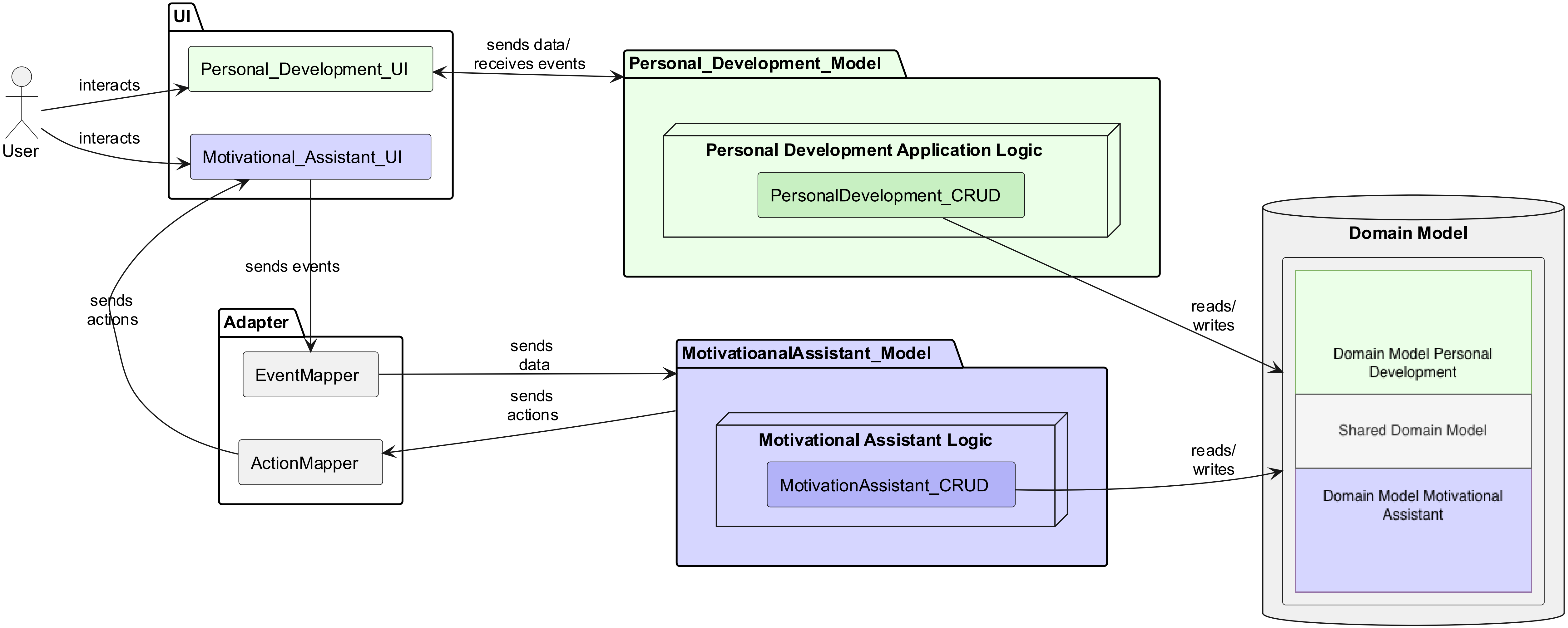
Context and Scope

The Motivational Assistant works as a Modular Plugin

- The **Motivational Assistant (MA)** is a standalone codebase/app/module that is separated from the base system
- The **Personal Development Application (PA)** delivers content, lessons, or activities to the user

- The MA can be *attached* to different applications via a plugin-like architecture.
- Implementation of the plugin-like architecture differs for different use cases

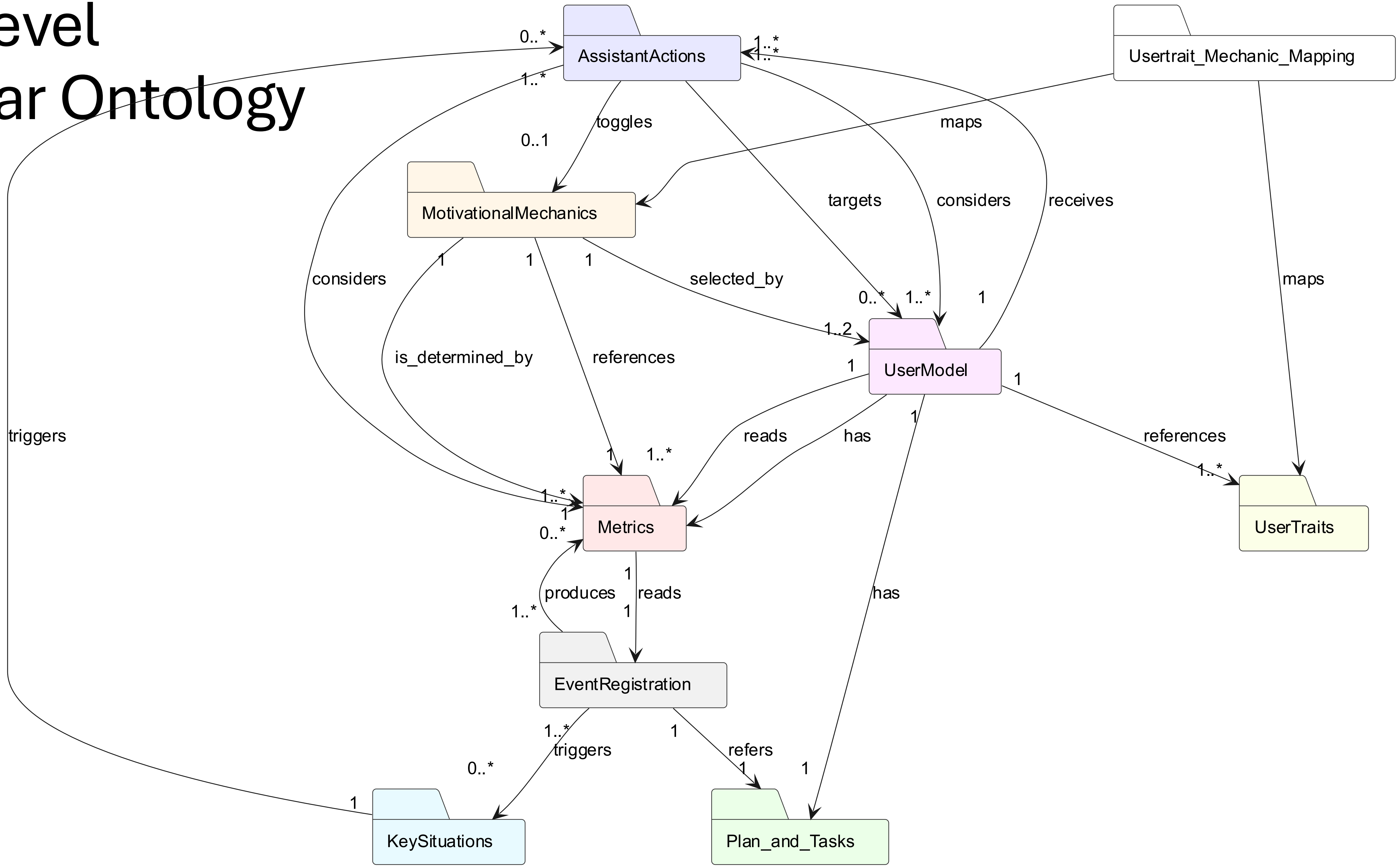
High Level System Architecture



Problem Statement

- We define a domain model for a motivational assistant as an ontology
- Reuse existing ontologies
- Existing ontologies lack domain generalization and practical use for developers
- We provide:
  - Implementation focused guidelines and vocabulary for developers
  - Provide vocabulary for designers
  - Evaluation of an MA to their PA or to evaluate their already existing MA

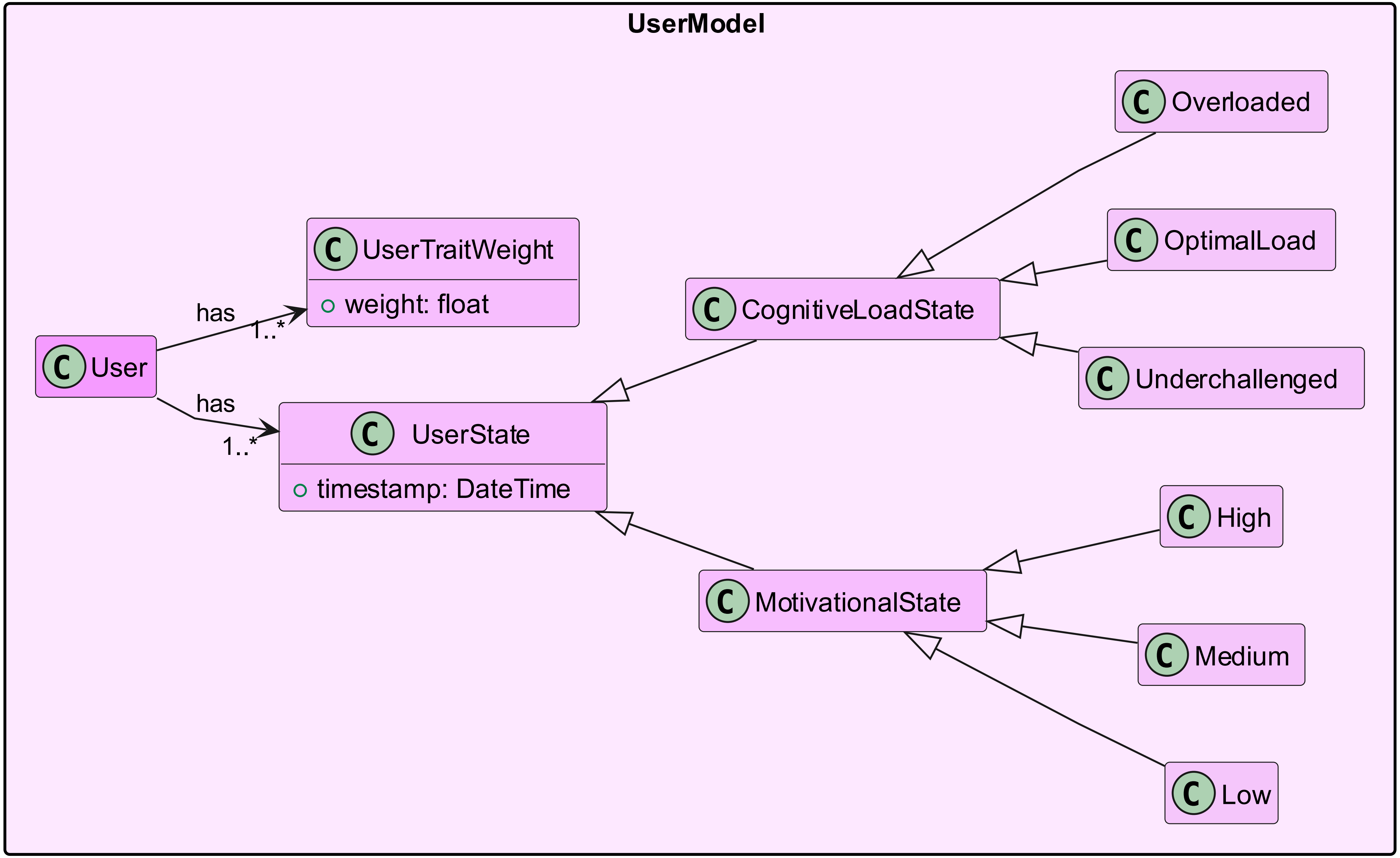
High Level Modular Ontology



UserModel

Holds Usertraits (Static)

Holds UserState (Dynamic)

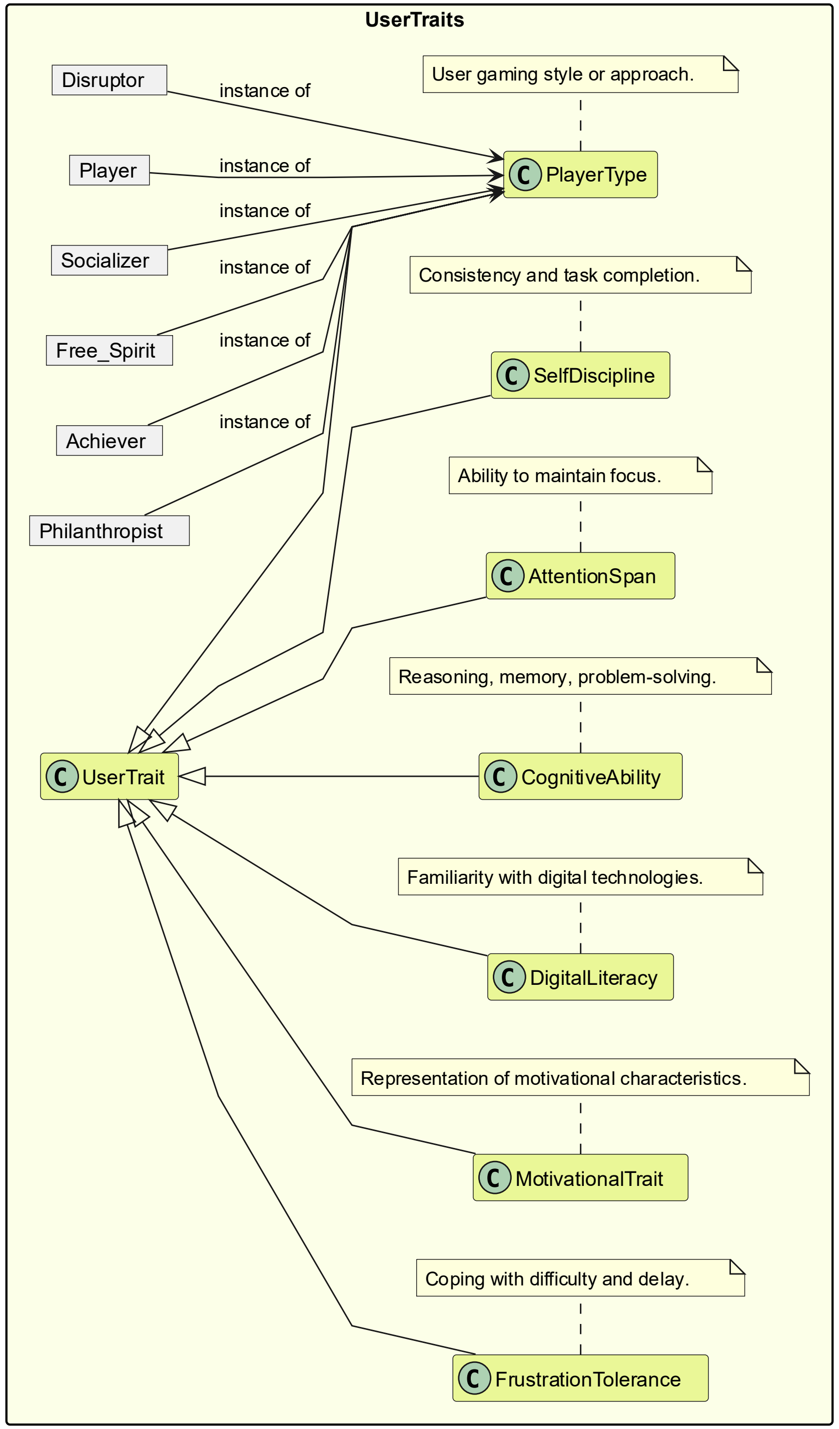


UserTraits and Mapping

- **Traits:**
  - Hexad player types + cognitive/behavioral traits
  - UserTraitWeight ∈ [0–1]
  - MechanicTraitWeight ∈ [–1–1]
- **Example:**
  Level & XP system → Achiever
  - weight 0.8

  ->means that the Achiever Player type will respond and be motivated by the Level & XP system with a strength of 0.8

New traits can be added to fit the **specific domain or target population**
e.g., health behavior traits, learning styles, or other dispositions.



User gaming style or approach.

Consistency and task completion.

Ability to maintain focus.

Reasoning, memory, problem-solving.

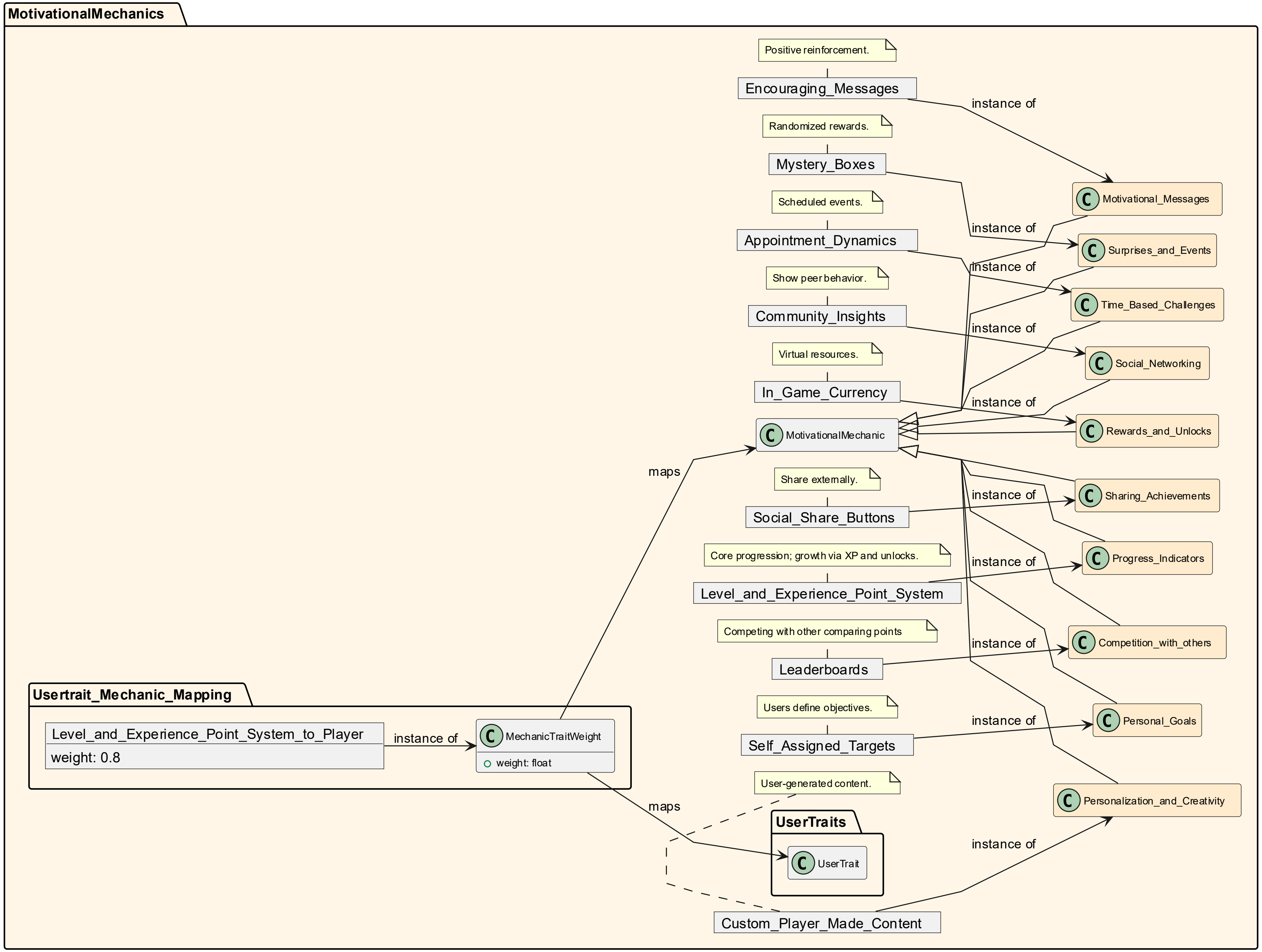
Familiarity with digital technologies.

Representation of motivational characteristics.

Coping with difficulty and delay.

Motivational Mechanics Catalog

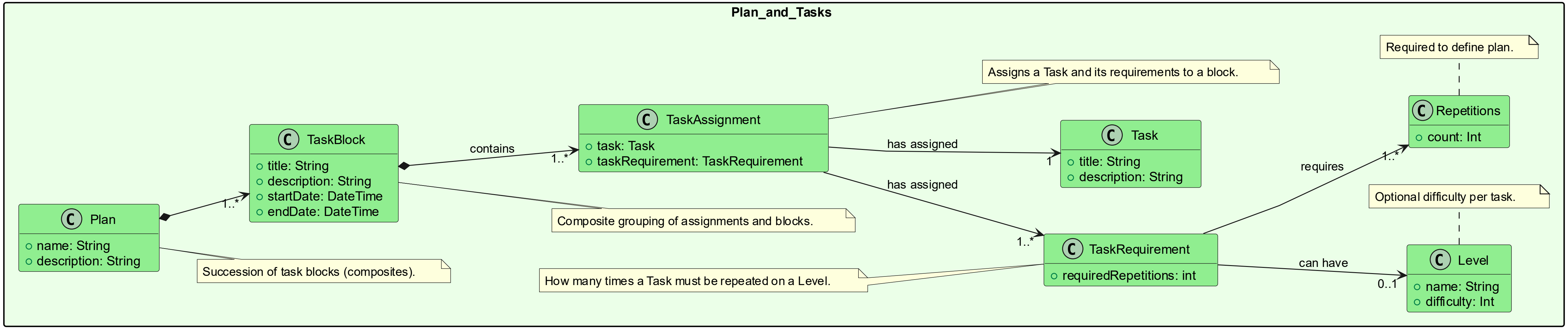
- 10 categories with one mechanic each as example
- MechanicTraitWeight maps a mechanic to usertrait and gives it a weight



Plan and Tasks

Purpose:
Defines progression timeline, difficulty curve, tasks and baseline goals for the user.

Serve as base for adherence calculation



Plan_and_Tasks

Plan
- name: String
- description: String

Succession of task blocks (composites).

TaskBlock
- title: String
- description: String
- startDate: DateTime
- endDate: DateTime

Composite grouping of assignments and blocks.

TaskAssignment
- task: Task
- taskRequirement: TaskRequirement

Assigns a Task and its requirements to a block.

Task
- title: String
- description: String

TaskRequirement
- requiredRepetitions: int

How many times a Task must be repeated on a Level.

Repetitions
- count: Int

Required to define plan.

Level
- name: String
- difficulty: Int

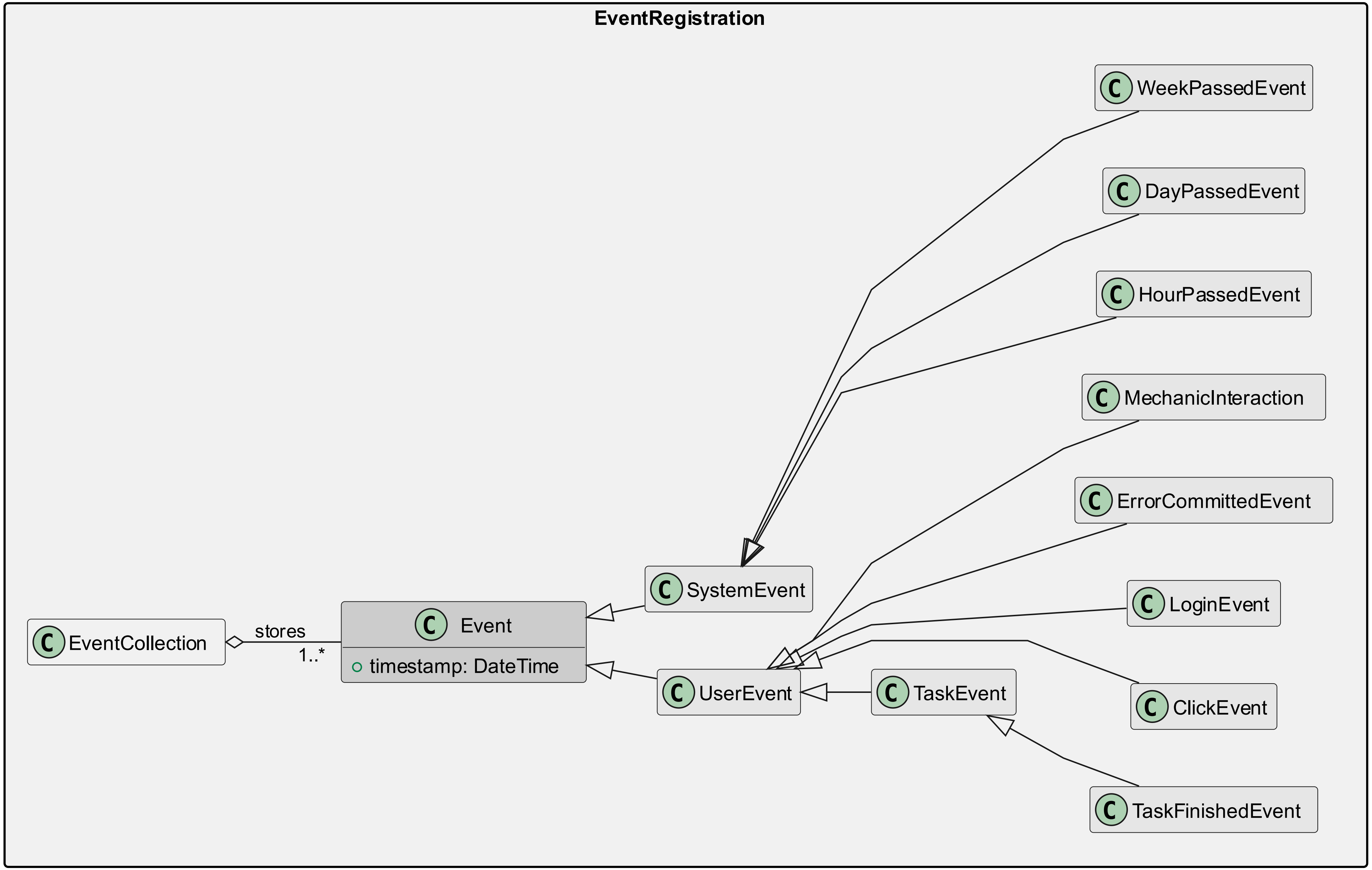
Optional difficulty per task.

EventRegistration

Provides system events that happen without user action

User events when user interacts with application

Events are logged in event collection over time



KeySituations

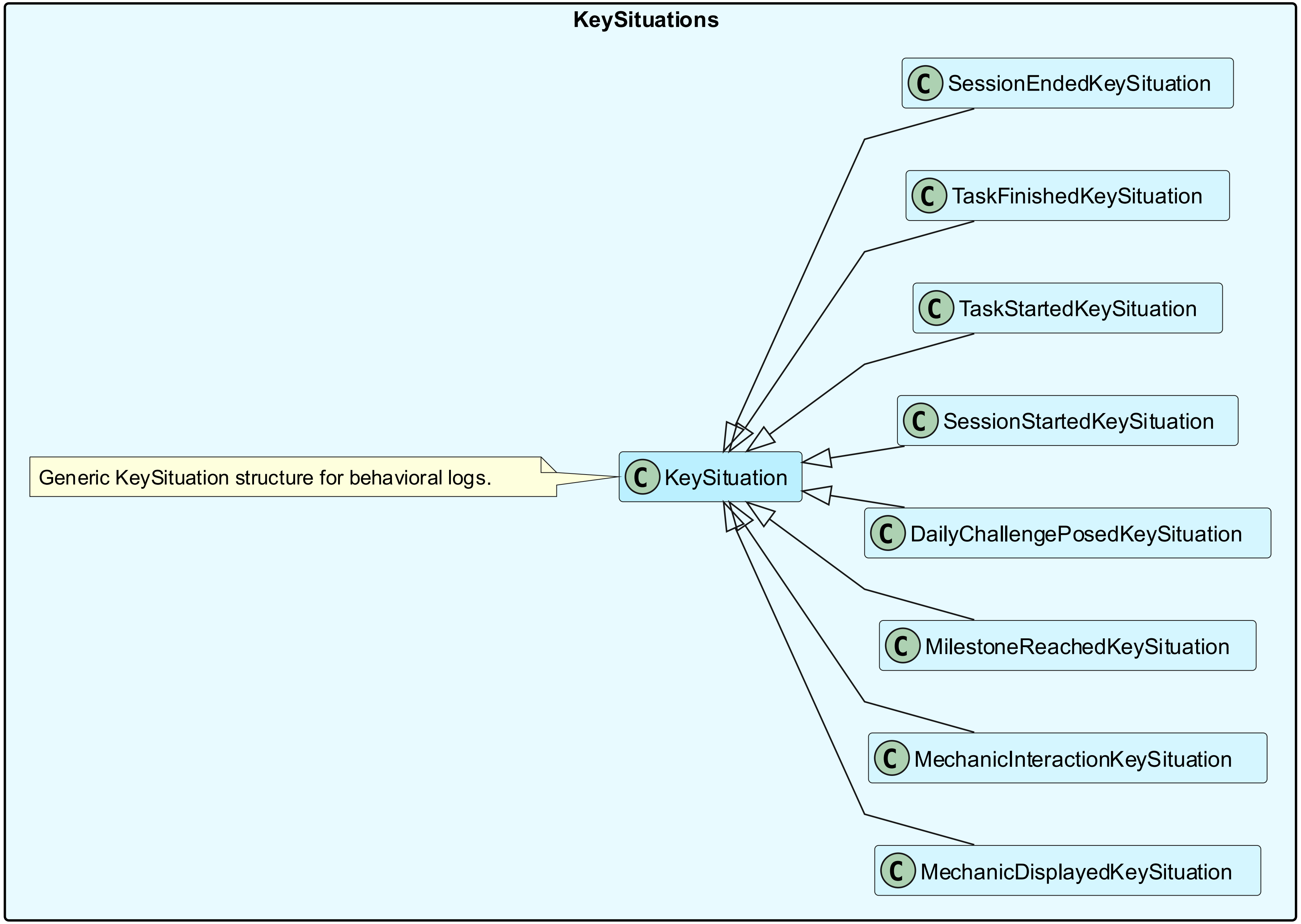
Are derived from events and define important moments like

-First login

-Session ended

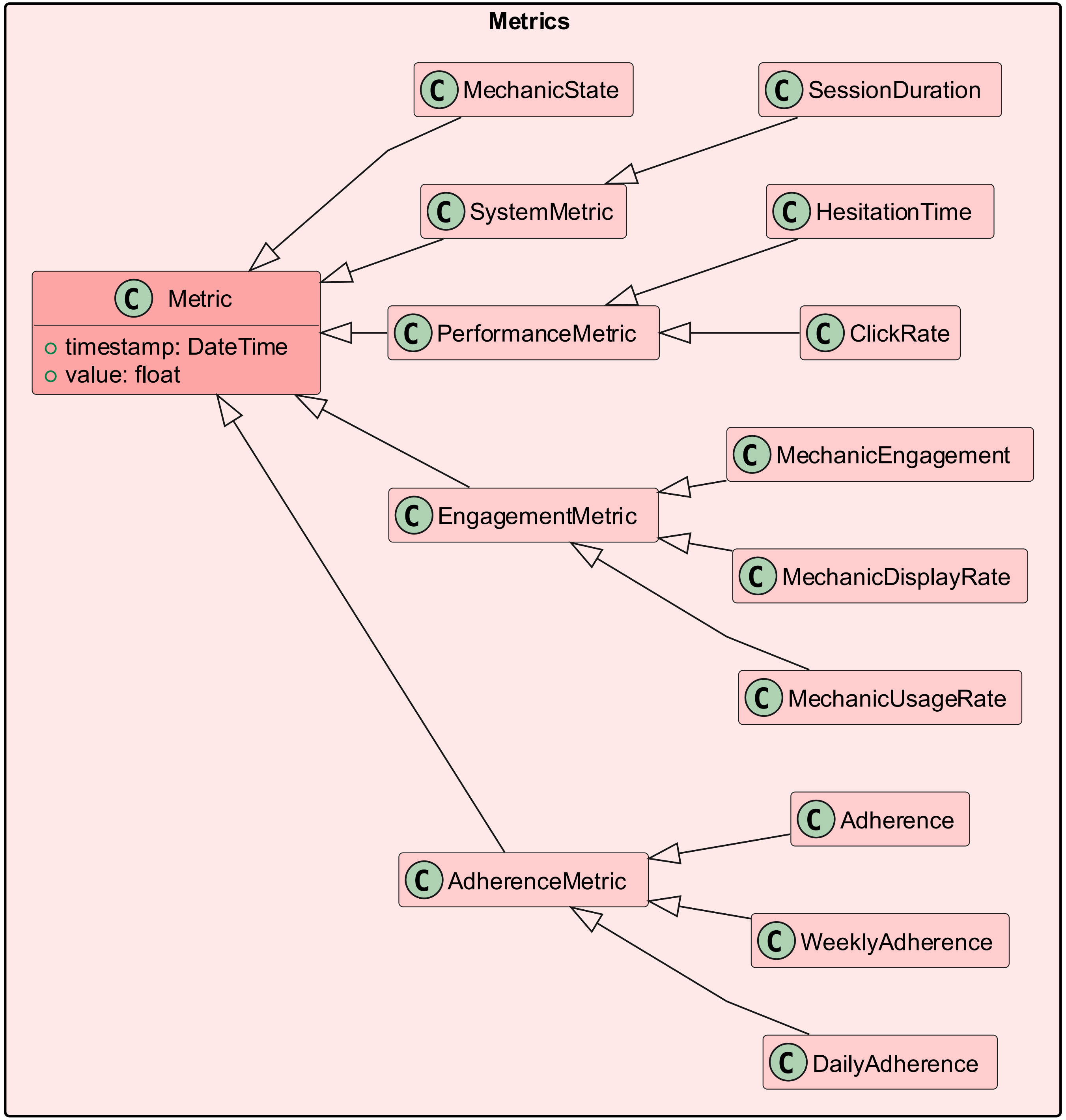
Are composed or one or more events

For example: End of Session and end of first week


KeySituations
Generic KeySituation structure for behavioral logs.
C KeySituation
C SessionEndedKeySituation
C TaskFinishedKeySituation
C TaskStartedKeySituation
C SessionStartedKeySituation
C DailyChallengePosedKeySituation
C MilestoneReachedKeySituation
C MechanicInteractionKeySituation
C MechanicDisplayedKeySituation

Metrics

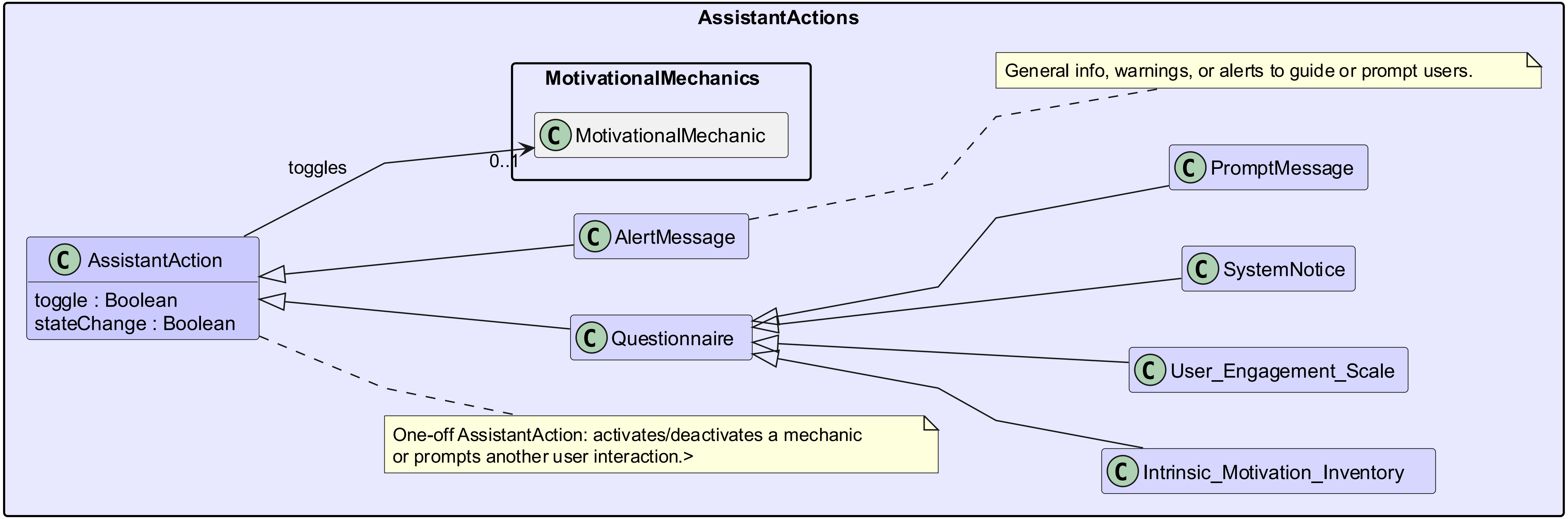
- Base class: Metric(timestamp, value)
- Computed from event data



AssistantActions

An action targeting the user and prompting interaction

Can be a questionnaire, an alert message or turning on/off a motivational mechanic


AssistantActions
MotivationalMechanics
MotivationalMechanic
General info, warnings, or alerts to guide or prompt users.
toggles
0..1
AssistantAction
toggle : Boolean
stateChange : Boolean
AlertMessage
Questionnaire
PromptMessage
SystemNotice
User_Engagement_Scale
Intrinsic_Motivation_Inventory
One-off AssistantAction: activates/deactivates a mechanic
or prompts another user interaction.>