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

Walkthrough and review

Motivation and Research Gap

- **Problem:**
  - Existing gamification ontologies are too abstract or too domain/ application specific
  - Developers lack reusable, implementation-oriented models
- **Need:**
  - Modular, domain-agnostic ontology powering the design of a motivational assistant

Contribution

• Modular semantic framework for a generalized motivational assistant in OWL and UML that can be applied in any behavior change and self improvement context

• Catalog of 65 mechanics sourced from existing material categorized in 10 groups

•Mechanics ranked through a user survey linking player types, Big Five personality traits, and motivational traits to mechanic preferences

•To be validated in two applications: *MejoraLaMemoria* and *FrailStop*

Context and Scope

Motivational Assistant as a Modular Plugin

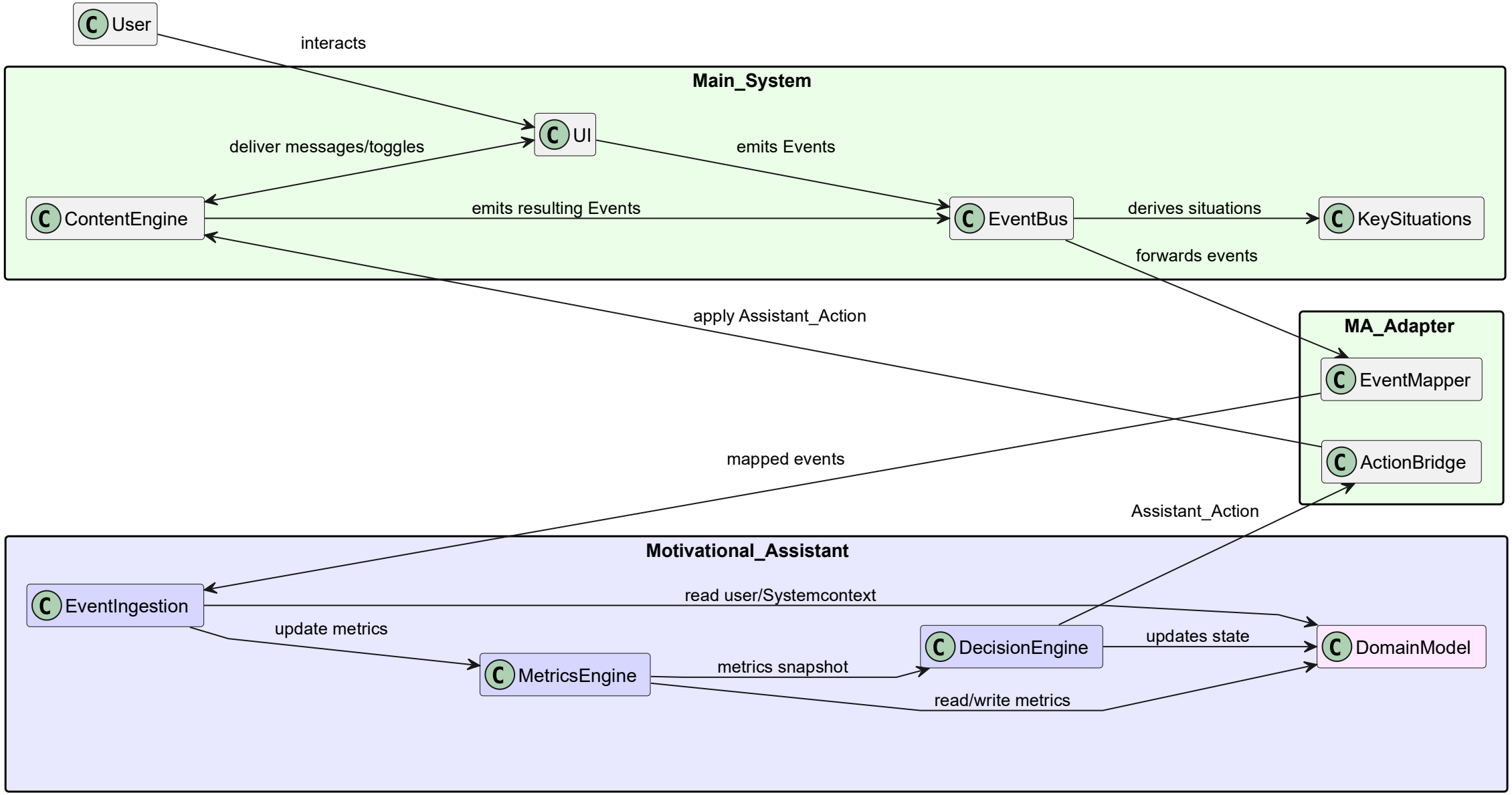
- The **Motivational Assistant (MA)** is a standalone codebase/app/module that is separated from the base system
- The **base system** delivers content, lessons, or activities.
- The **Motivational Assistant module** observes events, evaluates user state, and adapts motivational elements.

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

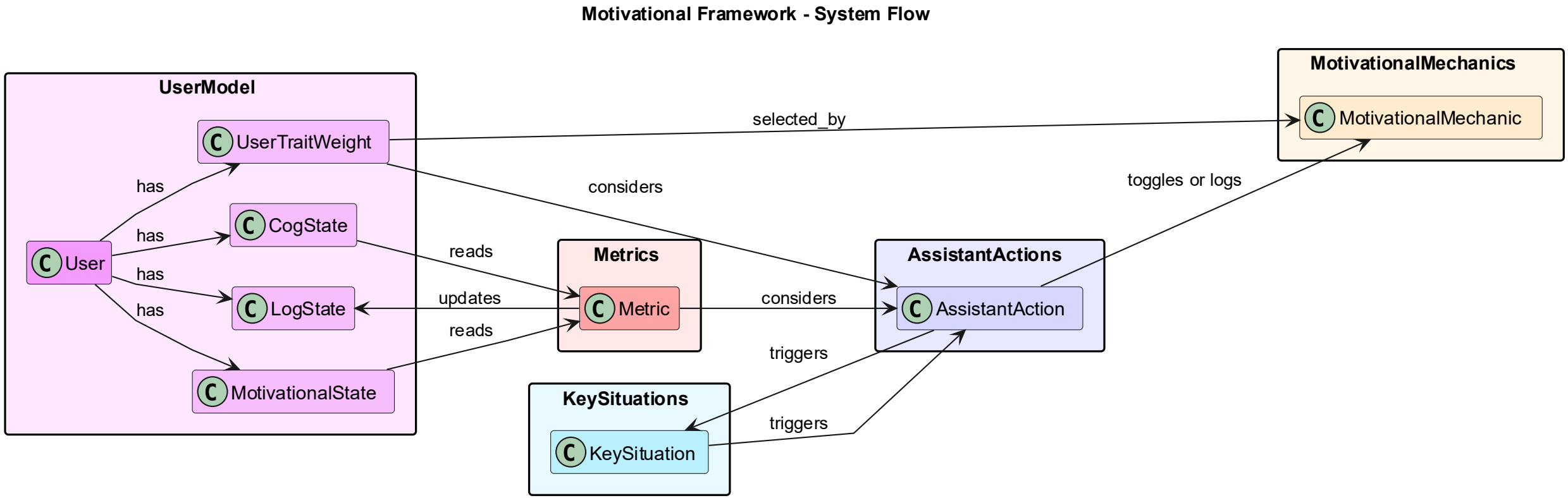
Current Situation and Motivation

- Existing apps: motivation logic deeply embedded → hard to reuse
- Desired: motivational assistant as a *reusable module*.
- Vision: any app can integrate motivation logic via an adapter.

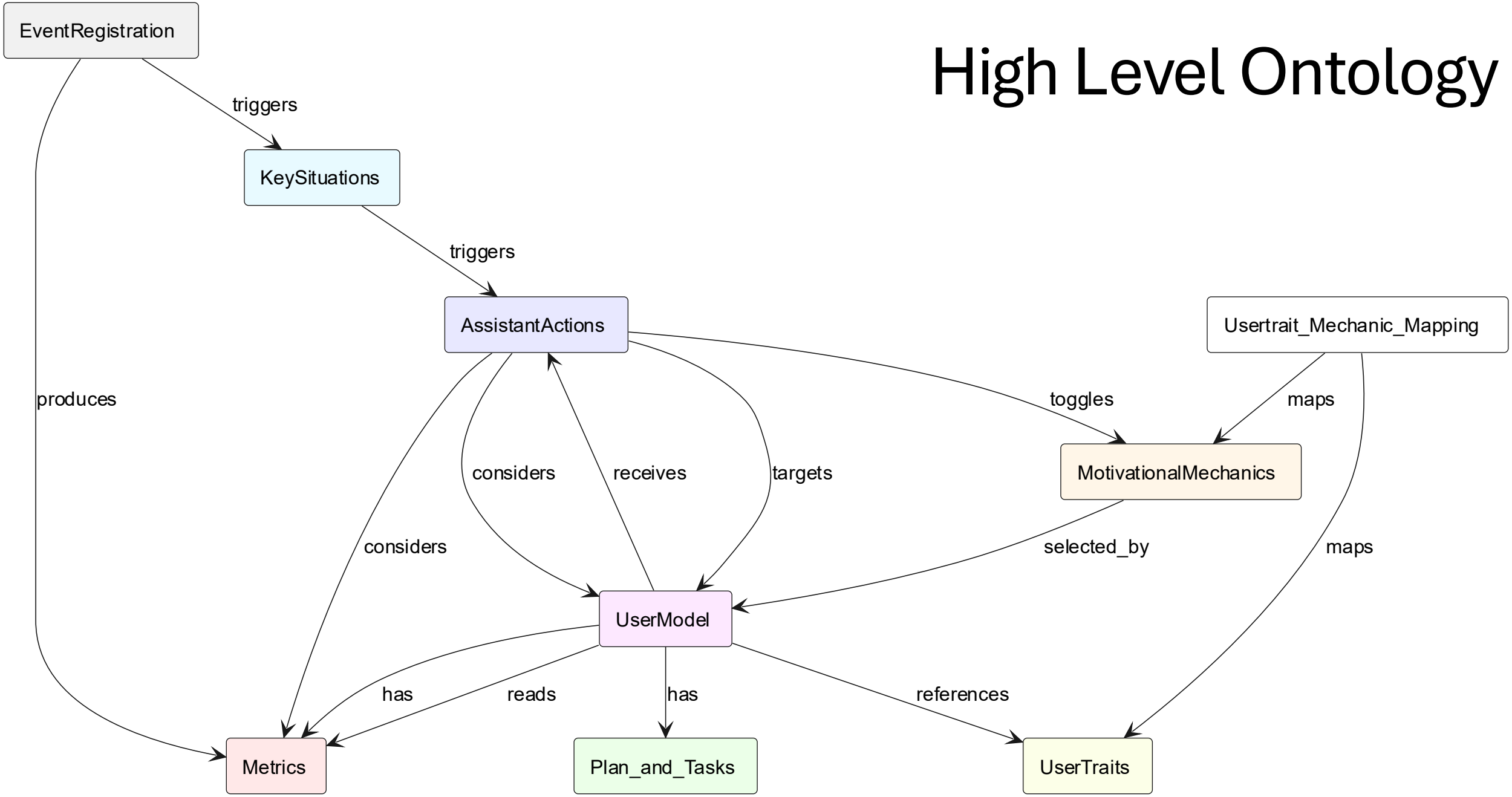
High Level System Architecture



High Level Motivational Assistant Flow



High Level Ontology



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

Ontology Overview

- **Main Modules:**
  - **EventRegistration**:
    - Captures user and system actions as timestamped events
  - **KeySituations**:
    - Occur when one or more important events coincide.
  - **AssistantActions:**
    - Delivered prompts that ecourage user action or change in behaviour such as messages, questionnaires, or toggle of mechanics
  - **UserModel**:
    - Stores user traits, states and relates a single user to metrics and plans
  - **Metrics**:
    - Computes measurements from events and usage.
  - **Plan_and_Tasks:**
    - Defines tasks blocks, repetition requirements and planned user progression
  - **MotivationalMechanics**:
    - Catalog of mechanics that influence motivation.
  - **UserTraits**:
    - Describes stable user characteristics and player types.

Main relationships:

- **triggers**: Source causes the target to occur.
- **reads**: Target is queried to inform the source.
- **considers**: Source uses target as an input to decide.
- **selected_by**: Mechanic choice depends on the target weights.
- **maps**: Links two classes and describes the strength of their relationship
- **toggles**: Enables or disables a mechanic or feature.

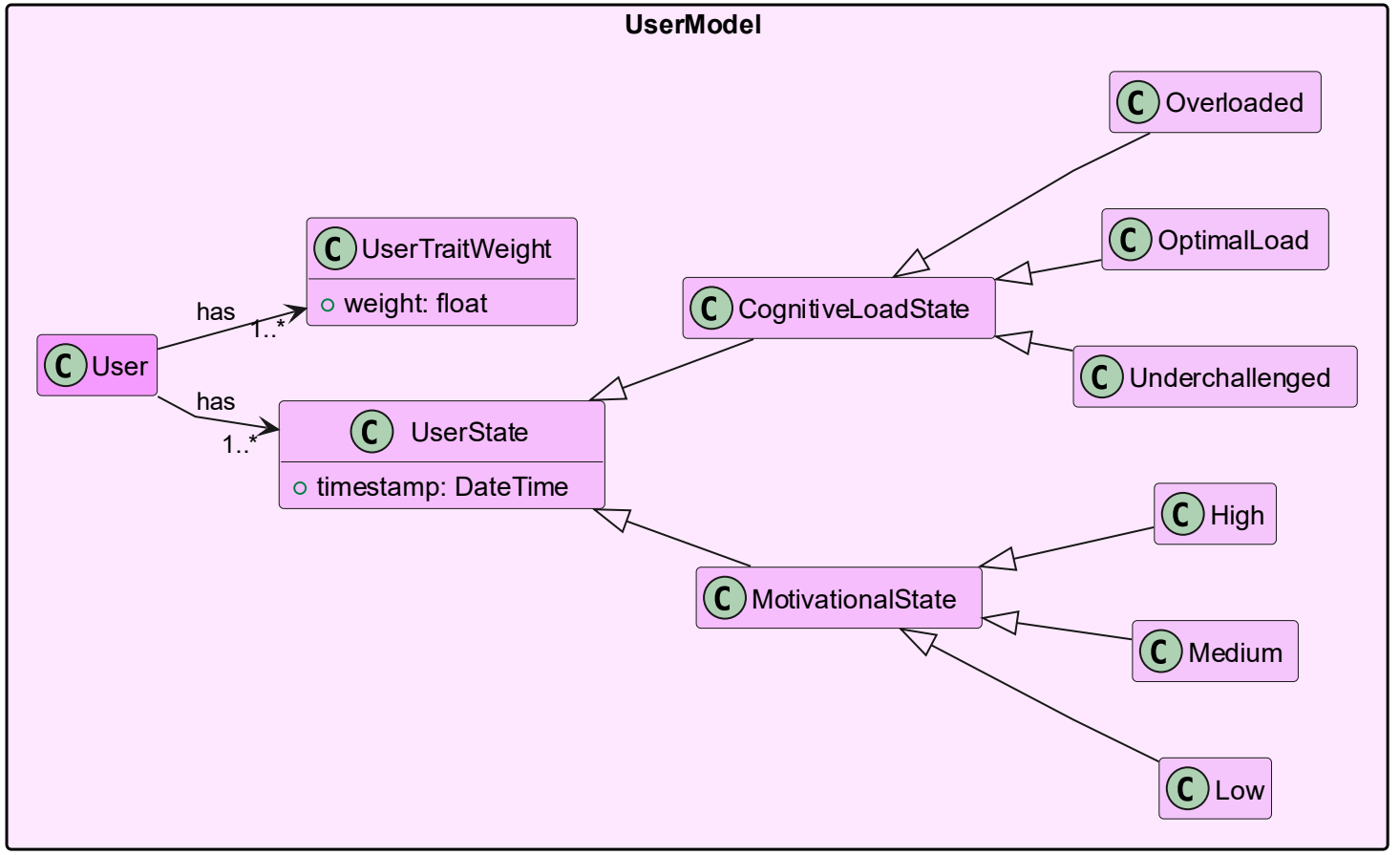
UserModel

Key Concepts:

- User, UserState, MotivationalState, CognitiveLoadState
- Temporal smoothing by using an average of values to avoid abrupt changes
- Provenance for baseline questionnaires

MotivationalState: Low / Medium / High
CognitiveLoadState: Underchallenged / Optimal / Overloaded

Users Preferences are **continuously measured** and saved as metrics via actual usage (e.g., MechanicUsageRate).



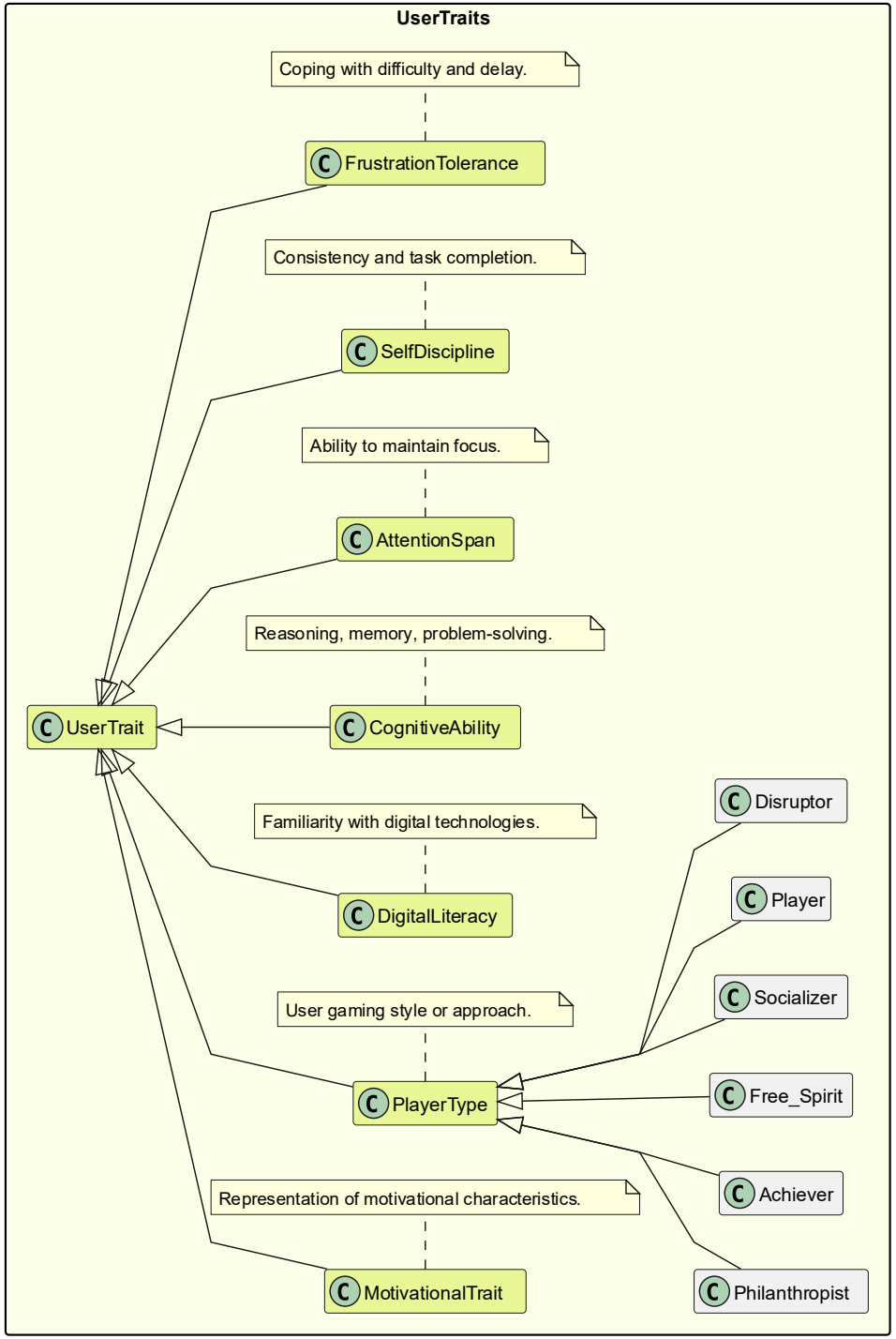
UserTraits and Mapping

- **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

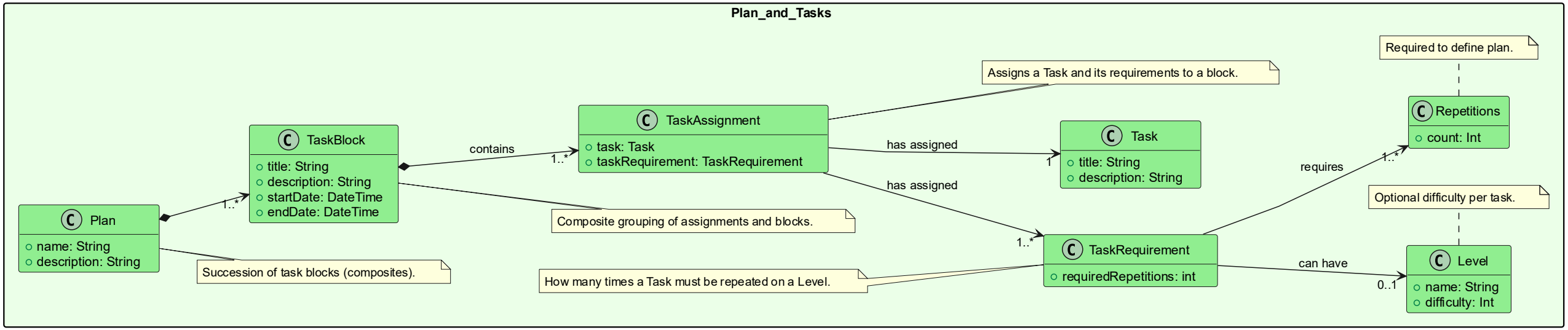
  The trait model is **modular and open-ended**

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



Plan and Tasks

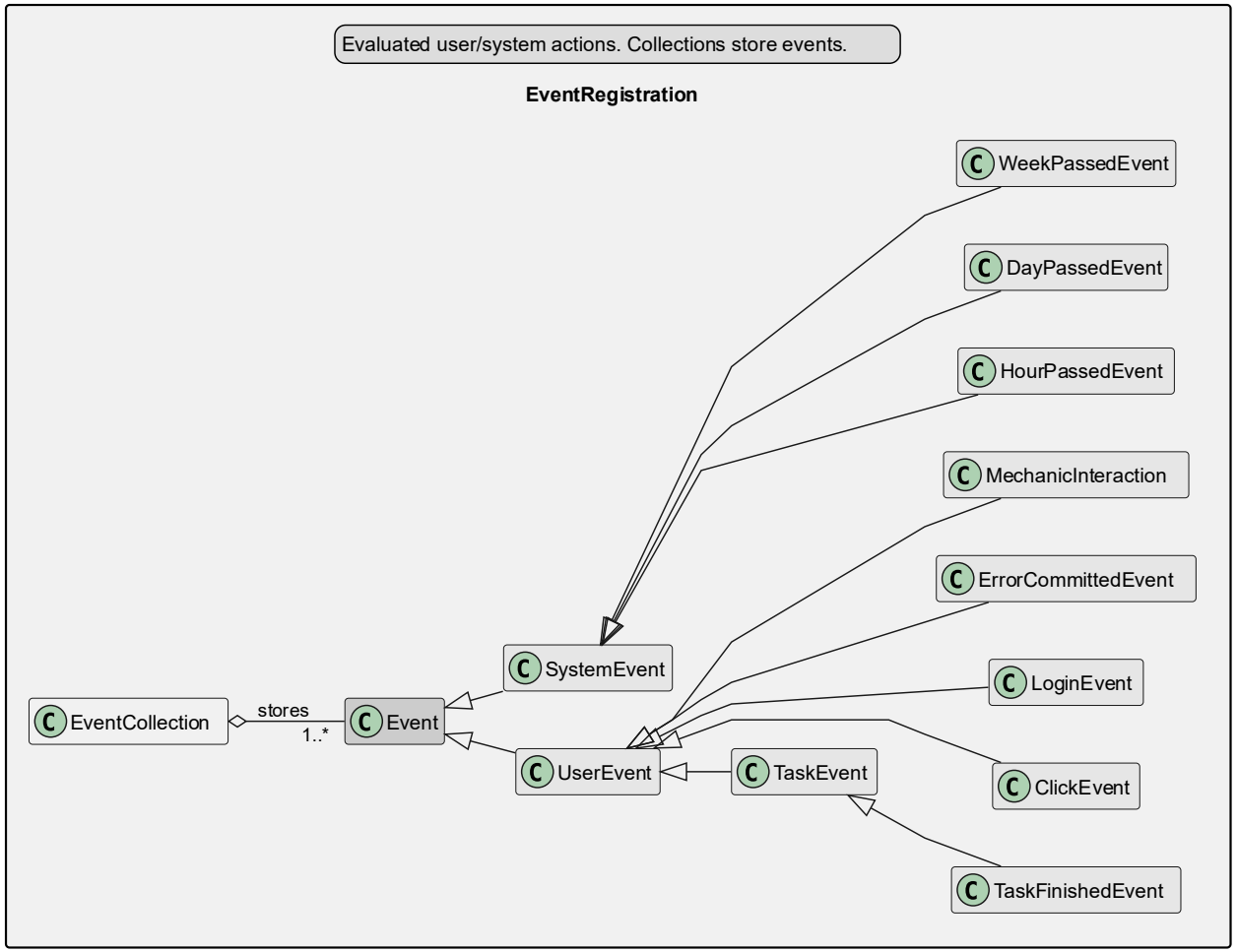
- **Structure:**
  Plan → TaskBlocks → TaskAssignments → TaskRequirements (Level, Repetitions)
- **Purpose:**
  Defines progression, adherence baseline, difficulty adaptation.



EventRegistration

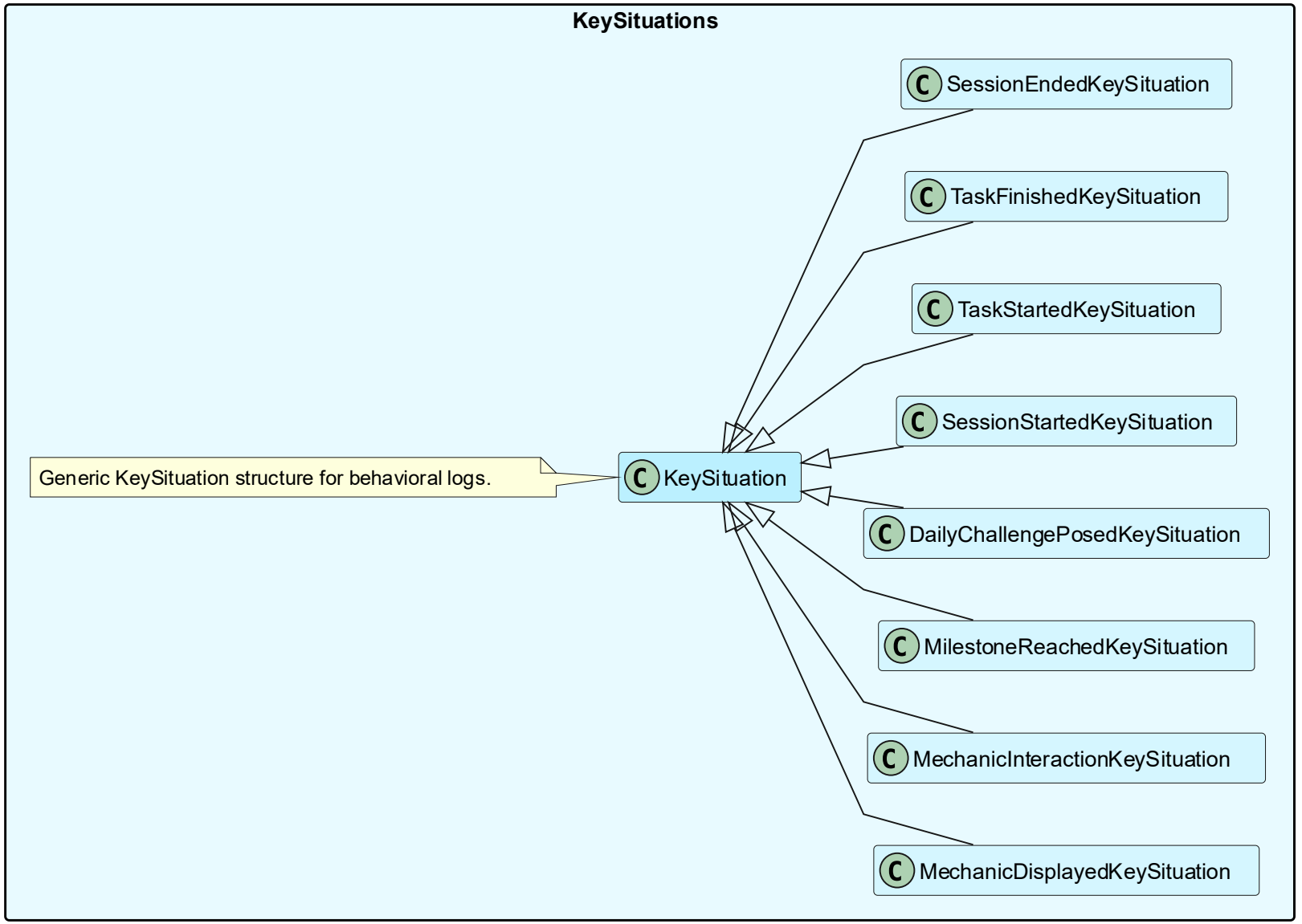
- Captures both user and system actions
- Event(timestamp, element) → UserEvent / SystemEvent / TaskEvent
- Examples:
  - ClickEvent,
  - LoginEvent,
  - TaskFinishedEvent,
  - ErrorCommittedEvent
  - Hour/Day/WeekPassedEvent,
- MechanicInteraction

- EventCollection aggregates multiple events for analysis



KeySituations

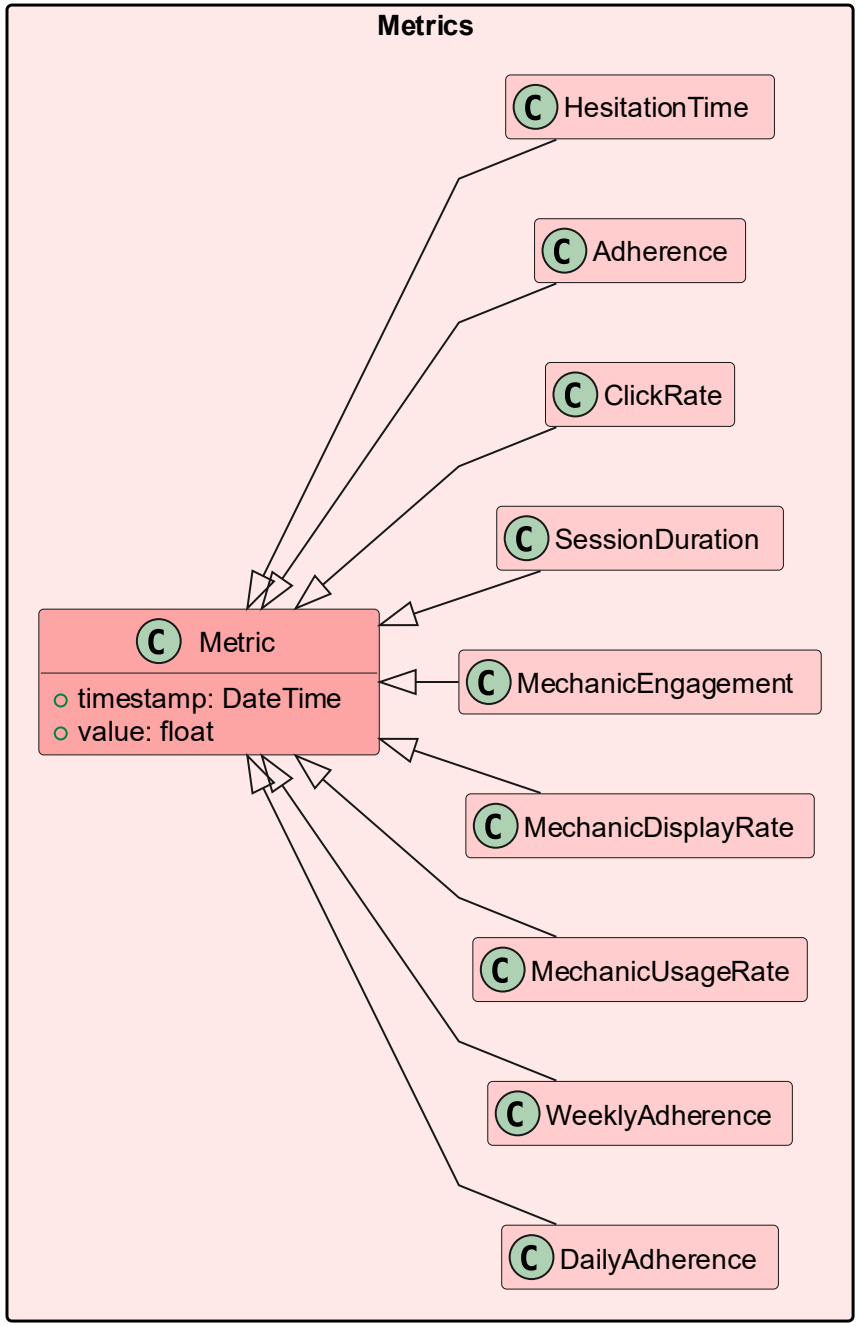
- MechanicSituations:
  - MechanicDisplayed,
  - MechanicInteraction,
  - MilestoneReached
- DailyChallengePosed,
- TaskStarted,
- TaskFinished,
- SessionStarted, SessionEnded
- Represent meaningful behavioral checkpoints
- Trigger Interjections or MotivationalMechanics



Metrics

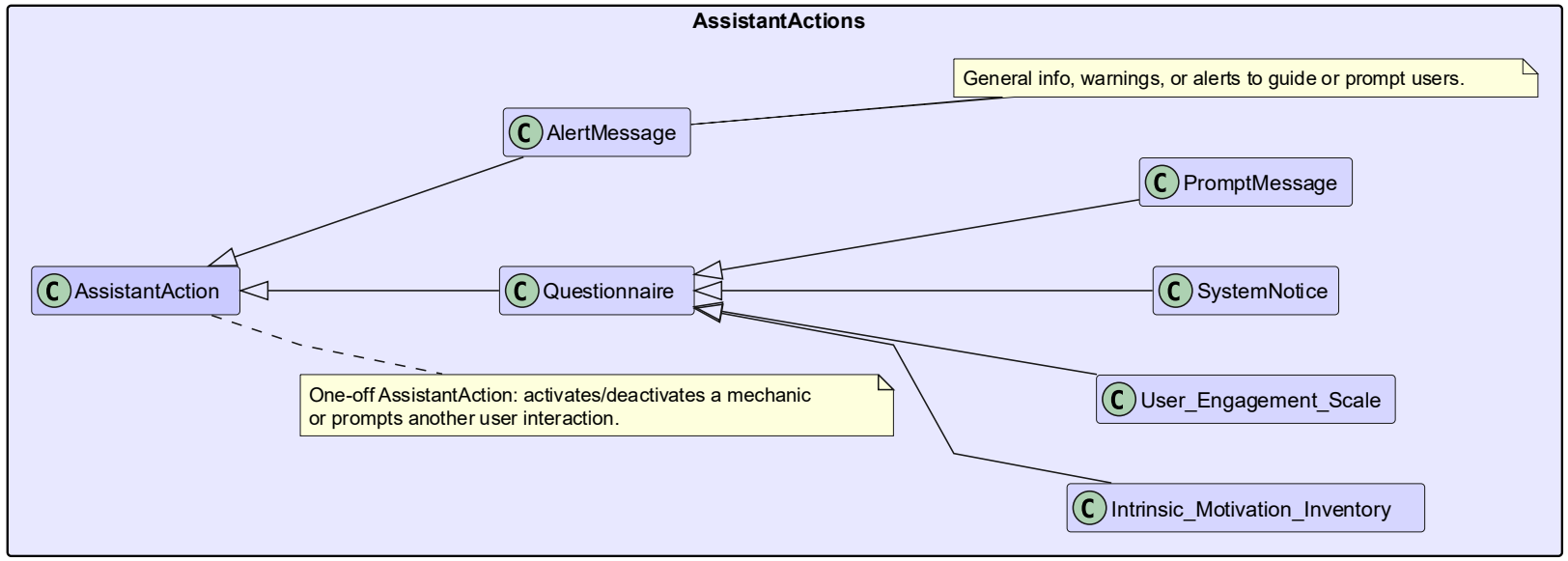
- Base class: Metric(timestamp, value)
- Examples:
  - DailyAdherence, WeeklyAdherence, Adherence
  - MechanicUsageRate, MechanicDisplayRate, MechanicEngagement
  - SessionDuration, ClickRate, HesitationTime

- Computed from event data using Goal–Question–Metric logic



AssistantActions

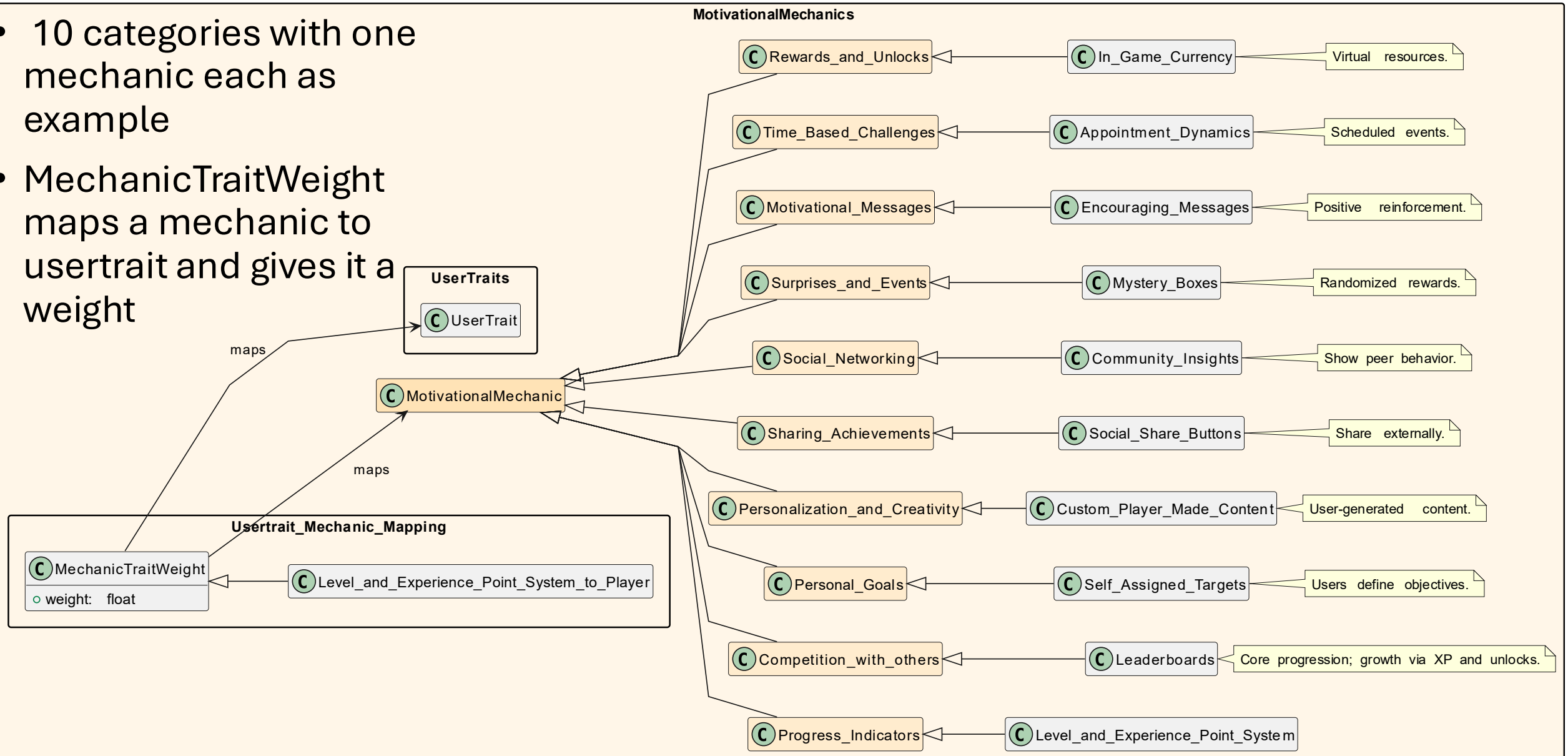
- AssistantAction can be Questionnaire, AlertMessage/SystemNotis or Mechanic Toggle
- Questionnaires:
  - Intrinsic Motivation Inventory
  - User Engagement Scale
- Selection logic considers :
  - Metrics
  - UserTraitWeight
  - UserState



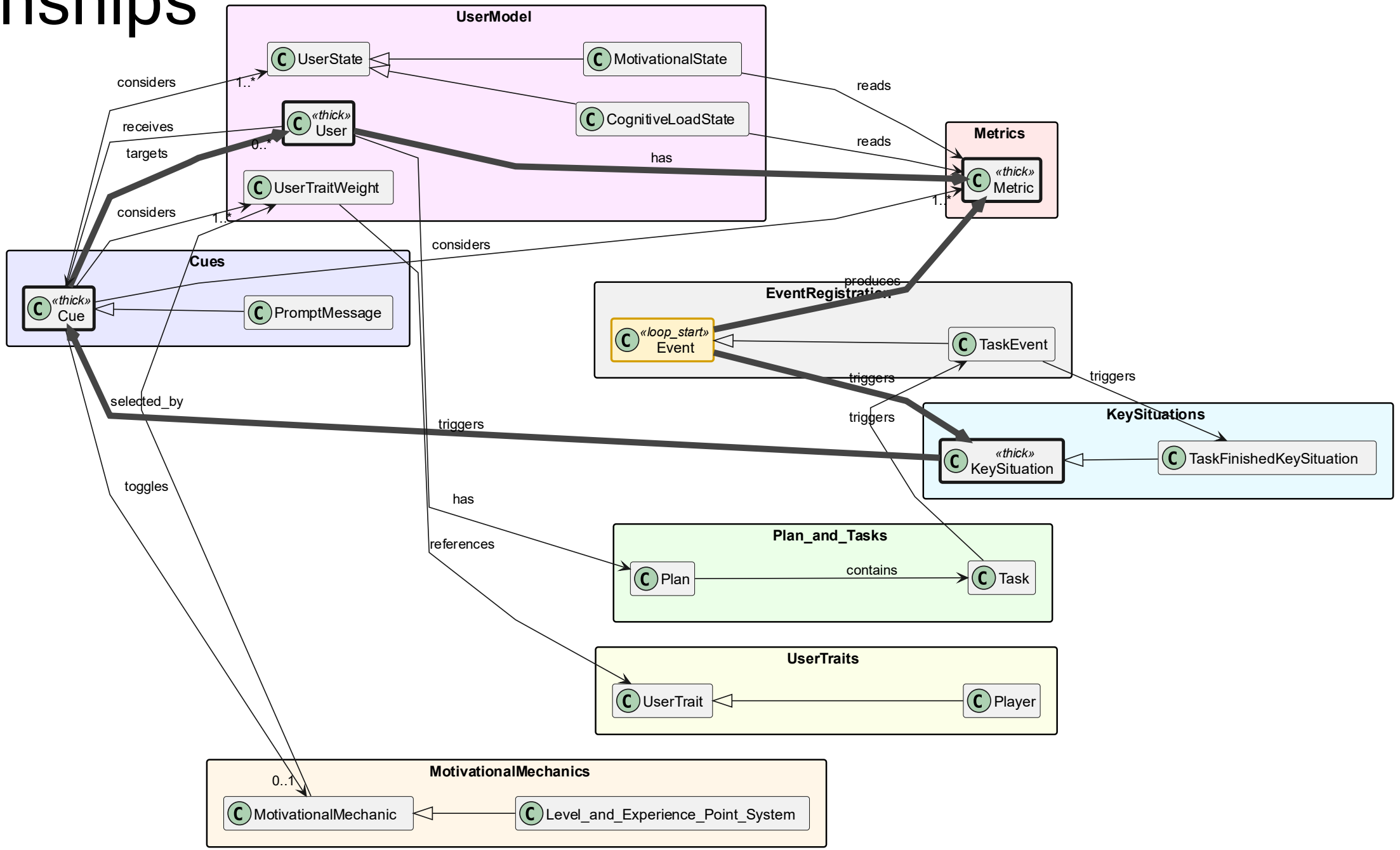
- Targets a User and can toggle Mechanics on/off

MotivationalMechanics (Catalog)

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



Relationships



Highlighted adaption pipeline

Decision and Monitoring Pipeline

- **Flow of Interaction and Adaptation**
- **Overview:**
  The system continuously monitors the user and reacts trigger relationships
- **Flow:**
  - **User performs actions** → captured as **Events**
  - **Events trigger KeySituations** (e.g., task finished, challenge posed)
  - **KeySituations update Metrics** (adherence, engagement, usage rate)
  - **Metrics inform decision logic** to adapt Mechanics or Actions and Score and select MotivationalMechanics.
  - **Deliver an Action** (message/questionnaire/toggle) via the chosen mechanic
  - **Learn** by Updating engagement Merics (e.g., MechanicUsageRate). Smooth over time.
- **Outcome:**
  A event driven feedback pipeline where user behavior drives metric updates, and metrics determine personalized motivational responses.