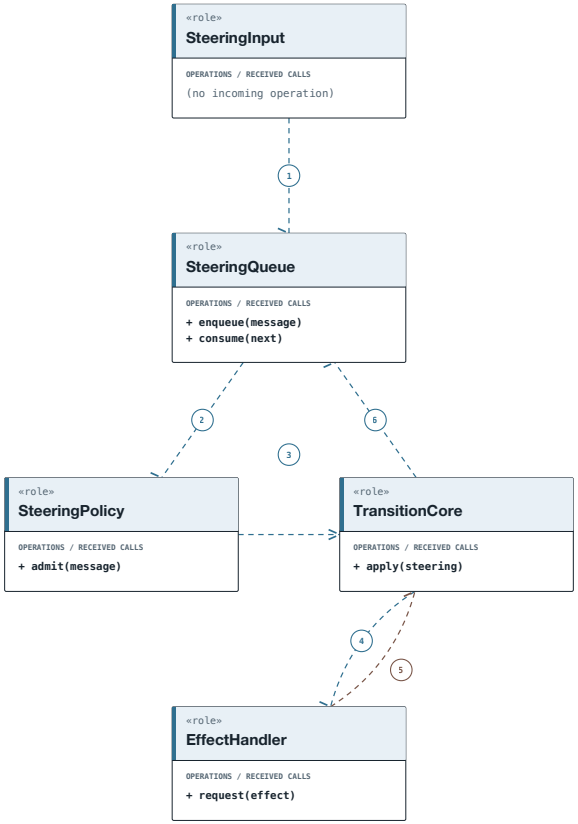
P06 Functional core and steering queues

Primitive / UML CLASS VIEW / ANALYTICAL MODEL

Separate transitions from live control.

UML CLASS VIEW

dashed arrow: call dependency · brown: return/failure · analytical, not an observed trace



POINTCUT / PATTERN INTERFACE

CONTEXT	A run accepts steering or queued input while transitions continue.
TRIGGER	A new steering message or input item is enqueued.
STRUCTURE	A transition core plus queued control messages and defined application points.
EFFECT	The transition consumes or defers steering with explicit ordering semantics.

CONTRACT / REQUIRED BEHAVIOR

REQUIRES	Steering messages have identity and a defined queue boundary.
ENSURES	Transition state changes only through the core and replayable control inputs.
INVARIANT	Live input cannot mutate shared state nondeterministically outside the queue.
VIOLATION	Queue overflow, conflict, or replay produces explicit policy outcomes.

RELATIONSHIPS / ANALYTICAL CALLS

number · caller → callee · message

1 SteeringInput → SteeringQueue enqueue(message)
2 SteeringQueue → SteeringPolicy admit(message)
3 SteeringPolicy → TransitionCore apply(steering)
4 TransitionCore → EffectHandler request(effect)
5 EffectHandler ↩ TransitionCore return(result)
6 TransitionCore → SteeringQueue consume(next)

VARIATION / TEST / NEGATIVE CONTROL

VARIATION	In-memory or durable inbox, priority, coalescing, and effect handlers.
TEST_OBLIGATION	Test injection timing, duplicate steering, replay, and conflicting messages.
NEGATIVE_CONTROL	A mutable global message list without transition semantics is insufficient.