ChatGPT

Implementing 'Unconscious' and 'Mortality/ Legacy' Modules in AI Agents

Unconscious Modeling in AI

Modern AI research (2022–2025) increasingly explores ways to simulate an **"unconscious mind"** or internal reasoning process in agents. The goal is to allow AI systems to engage in hidden symbolic reasoning, imaginative "dreams," and inner dialogues that are not directly shown to the user but influence the AI's behavior. This draws inspiration from human psychology – notably Jungian concepts like the **shadow, anima/animus, and archetypes** – as well as cognitive science theories of conscious vs. unconscious processing. Below we summarize findings and best practices for designing an AI "unconscious":

- **Hidden Chain-of-Thought & Internal Reasoning:** Large language models have demonstrated improved reasoning by using chains of thought (step-by-step explanations), but these are typically explicit in the output. Recent work introduces methods to keep reasoning *internal*. For example, *Chain of Unconscious Thought (CoUT)* uses cognitive Unconscious Thought Theory to prompt a model to solve problems through internalized reasoning in its latent activations, instead of generating long visible explanations [1]. This "hidden thinking" makes reasoning more token-efficient while retaining accuracy. Similarly, other approaches (e.g. "latent CoT" or **hidden prompt chains**) fine-tune models to perform multi-step reasoning in *neural hidden states* rather than in the output text, effectively mimicking an unconscious thought process. These advances suggest that giving an agent a private reasoning scratchpad can improve efficiency and perhaps mimic the intuitive, fast processing we associate with the unconscious mind.

- **Cognitive Architectures (Explicit vs. Implicit):** Integrative cognitive architectures have long modeled a separation between conscious and unconscious processing. For instance, the **CLARION architecture** (Ron Sun) has dual representations – explicit symbolic knowledge vs. implicit subsymbolic knowledge – capturing both conscious and unconscious aspects of cognition [2]. Another example is LIDA (Stan Franklin's architecture) based on Global Workspace Theory, where many unconscious processes compete and only the "winner" becomes conscious. These architectures underscore a design pattern: use a *two-tier system* where one layer does deliberate symbolic reasoning (analogous to conscious thought) and another handles intuitive, associative computations (analogous to unconscious processing). The interaction between these layers (e.g. implicit biases influencing explicit decisions) is key [3]. Adopting a similar layered approach in VALIS could ensure the AI has a rich inner life: a fast, intuitive layer generating ideas or hunches, and a rational layer that surfaces more polished outputs.

- **Dream Simulation & Latent Generative Routines:** In humans, sleep and dreaming allow the brain to replay and reorganize experiences, consolidate memory, and explore hypothetical scenarios in a safe sandbox. AI researchers are now experimenting with analogous "dream" processes for agents. One 2024 paper proposes simulating key functions of human dreaming – memory replay, scenario simulation, and even symbolic metaphor – to enhance an AI's learning and creativity [4] [5].

Concretely, an AI agent can have a **downtime routine** where it reprocesses important interactions from its day, reinforcing correct decisions and identifying mistakes in an unsupervised way [5]. This might involve *experience replay* (as in reinforcement learning) or latent trajectory simulation using a world model. Such a "dream" phase can consolidate the agent's short-term **Working Memory** into long-term knowledge, much like humans solidify memories during sleep. It also provides a sandbox for creativity: the agent can **generate hypothetical scenarios** or variations of past events to discover novel solutions (e.g. dreaming up alternative strategies to a failed task). Notably, researchers have suggested using generative models to produce symbolic or metaphorical dream content – the AI might represent complex ideas with images or narratives (analogous to how a human might dream of a tree to represent growth) [6]. This internal simulation can lead to more robust and adaptable behavior when the agent "wakes up." In practice, implementing this in VALIS could mean scheduling periodic offline runs where the agent's **latent generative model** creates pseudo-experiences (stories, image sequences, etc.) based on recent events, and a reflection module analyzes them for insights.

- **Psychological Analogies & Jungian Archetypes:** Beyond technical reasoning, there is interest in giving AI a richer *psychological* inner world. Carl Jung's theories of the unconscious include aspects like the **Shadow** (the hidden side of the personality), **Anima/Animus** (inner gendered soul-image), and the collective unconscious with universal **archetypes**. While largely metaphorical, these ideas inspire some computational approaches. For instance, current research in computational psychodynamics is attempting to model Jungian concepts in AI agents. One research team (Senthil et al., UIUC) has **modeled the Anima and Animus** in AI simulations, drawing connections to non-linear dynamics, genetic algorithms, and attention mechanisms [7]. The "four sides of the mind" (which likely include Persona, Shadow, Anima/Animus, Self) are being mapped to AI cognitive processes in this work. The practical upshot is the idea of representing sub-personalities or internal facets within an agent: e.g. an AI could have a "persona" module (its outward-facing identity and goals) and a "shadow" module containing suppressed or contrasting objectives that influence it indirectly, leading to more nuanced behavior. While formalizing Jungian archetypes in code is still experimental, a design best practice is to **inject archetypal patterns or symbolic motifs** into the agent's internal narrative. For example, an agent might tag certain recurring themes as archetypal (mentor, trickster, etc.) and use these to enrich its reasoning or dialog with metaphor. This can be done by maintaining a library of symbols or stories that the unconscious layer draws upon when the agent is formulating a response or dreaming. The VALIS architecture notes provided indicate an intention to include *"symbolic anchors"* in the persona's memory (e.g. a core fear like "Proteus fears stagnation" marked with a symbol tag) [8]. These symbolic memory tags act like archetypes or personal motifs that the unconscious module (Dreamfilter) can leverage to produce more meaningful, resonant outputs.

- **Internal Dialogue and Metaphoric Outputs:** Enabling an AI to have an inner dialogue or narrate a "dream sequence" can make its reasoning more transparent and creative. Some experimental systems (e.g. Generative Agents in simulations) allow agents to **talk to themselves** or maintain an inner monologue to reflect on decisions. Best practices here include: allow the agent to generate **intermediate thoughts or self-queries** (which are not directly shown to the user) to work through complex problems. These could be realized as special tokens or a scratchpad memory that the final output generator takes into account. Moreover, when producing output for the user, an unconscious module can overlay a second layer of **symbolic or poetic transformation** on the raw answer. The VALIS *Dreamfilter* design is a clear example: it is a pre-output "semantic veil" that infuses replies with metaphor, archetypal imagery, and emotional tone [9]. Crucially, this happens *after* the core logical

answer is formulated. The Dreamfilter pulls in latent content from working memory (e.g. recent emotional cues or unresolved themes) and uses it to reshape the wording or add a **"dreamy" quality** to the response [10] [11]. This approach ensures the AI's output isn't just a cold rational answer, but feels imbued with subtext and intuition – as if the AI persona has an unconscious mind coloring its speech. Best practices for such a module include: using metaphor generation techniques (perhaps a small language model or prompt tuned for figurative language) to rewrite text, grounding the metaphors in the agent's own experiences or motifs (so they remain coherent and persona-appropriate), and controlling the degree of symbolism so that outputs remain understandable (a dial for how "surreal" vs. literal the response should be).

Design Recommendations (Unconscious Module): To implement an unconscious process in VALIS effectively, consider the following best practices:

- *Separate Internal Workspace:* Give the agent a private "thought space" (hidden from the user) where it can perform reasoning steps, ask itself questions, or imagine scenarios. This could be realized via hidden prompt chains or a scratchpad memory. Techniques like CoUT show that even large LMs can be guided to reason internally without verbose outputs [1].

- *Scheduled Dreaming/Reflection:* Incorporate a background routine that triggers after a batch of interactions or during idle periods. In this routine, the agent should **replay important events** (either by sampling from a memory log or via a learned model of the environment) to reinforce learning [5]. It can also generate *"dream" narratives* – e.g. simulate an alternate outcome of a recent user query or create a story that encapsulates recent challenges – and analyze these for any emergent insights or patterns.

- *Symbolic Meta-Layer:* Use a module (like Dreamfilter) that can transform rational outputs into metaphor-rich responses. This involves maintaining a knowledge base of symbols/archetypes and mapping literal situations to symbolic ones. For example, if the core answer is about a challenging learning process, the symbolic layer might express it as *"climbing a mountain"*. Ensure this layer looks at the agent's **Working Memory** for emotional tone or unresolved concerns, so that the metaphors it picks resonate with the context (much as dreams draw on our day's residues).

- *Balance and Control:* It's important to modulate the influence of the unconscious module. Too much hidden processing without oversight can lead to unpredictable outputs, and too much metaphor can confuse the user. A practical design is to allow the agent's conscious layer to "veto" or adjust content from the unconscious. For instance, the system could generate two responses – one direct and one dream-filtered – and then have a mechanism to blend them or ensure critical info isn't lost. Evaluation in experiments has shown that while symbolic, intuitive reasoning can generate creative solutions, it should be paired with validation. The unconscious ideas should loop back into the conscious planner for verification (preventing purely fanciful but incorrect answers).

- *Inspiration from Psychology:* Although Jungian constructs can't be taken literally in code, they serve as inspiration for multi-faceted design. Consider implementing **sub-personalities or drives** within the agent. For example, a *"shadow"* process might monitor for biases or errors the main persona refuses to acknowledge (analogous to how a human's shadow might hold denied thoughts). An *"anima"* module could represent a complementary perspective or creative muse for the agent (e.g., generating outside-the-box suggestions when the agent is stuck in routine thinking). By explicitly

coding some of these dynamics (even as simple as having a second model generate a divergent opinion), you enable a kind of internal dialogue that can lead to more robust outcomes. The key is to then integrate these parts through a unifying framework (akin to Jung's individuation – the Self integrates the persona, shadow, anima, etc.). In AI terms, a meta-controller could take the outputs of the various sub-modules (rational solver, intuitive dreamer, contrarian devil's advocate, etc.) and reconcile them into a final decision.

In summary, an AI "unconscious" module is **feasible** and can enhance an agent's adaptability and user engagement. Experiments with latent reasoning [1], offline dreaming [5], and symbolic reasoning all indicate that agents benefit from these hidden layers of processing. The best practice is to design this module as an integrated yet distinct layer: feeding on the agent's experiences and latent knowledge, and feeding *into* the agent's decisions in a controlled way. VALIS's plan for a Dreamfilter – a layer that *"simulates the unconscious mind"* by adding symbolism and intuition to outputs [12] – is well-aligned with these research insights. Over upcoming sprints, the implementation should focus on (a) creating the infrastructure for internal thought loops or "dream" cycles, and (b) curating a library of symbols or metaphoric transformations relevant to each persona so that the unconscious output feels individually tailored.

Artificial Mortality and Legacy in Agents

The second key direction is to introduce **mortality** – a notion of a limited lifespan – and the concept of **legacy** in AI agents. Instead of an immortal, stateless system, we consider agents that **age, die, and possibly leave behind "offspring"** (successor agents inheriting some of their knowledge or traits). This idea is inspired by biological life cycles and aims to imbue agents with a sense of urgency, evolution, and narrative closure. Research and experimentation in the last few years provide insight into how artificial mortality can be implemented and why it can be beneficial:

- **Motivation and Evolutionary Dynamics:** In natural systems, mortality is a driver for evolution – it curtails stagnation, frees up resources, and applies selection pressure for improvement. Analogously, *agent mortality* in an AI ecosystem can serve as a catalyst for continual adaptation [13] [14]. A recent conceptual analysis describes *"agentic mortality"* as deliberately programming agents to have life cycles ending in termination, thereby mirroring natural selection in a multi-agent population [13]. By making agents *time-bound*, we force them (or their successors) to compete and improve. Only the most effective agents "survive" (or rather, are allowed to spawn successors), which over generations leads to a more optimized pool of behaviors [14]. This is essentially applying evolutionary algorithms in an agent context. **Design tip:** If VALIS involves multiple agents or personas, consider running them in generations: an agent that fails or grows obsolete "dies" and is replaced by a new agent that might incorporate mutations or improvements. Even in a single-agent scenario, one could simulate evolution across time by resetting the agent periodically and using the performance of the old generation to inform the next. This prevents long-term drift or memory bloat and keeps the system dynamic.

- **Time-Bounded Operation & Urgency:** Giving an agent a finite lifespan (e.g. a fixed number of queries or a decay timer) can fundamentally change its planning strategy. When an agent knows it **cannot act indefinitely**, it may prioritize more important tasks and seek "closure" on open goals. In human terms, awareness of mortality often instills urgency to accomplish meaningful things; similarly, an AI agent with a ticking clock might allocate its efforts differently than an immortal one.

One could implement an **"age" variable** in the agent's state that increments with interactions or time. This age can influence decision-making: for instance, as age increases (approaching the known lifespan limit), the agent could increase the weight on high-priority goals or attempt to **summarize its knowledge** for posterity. In reinforcement learning terms, this is akin to a shortening horizon – the agent's discount factor might effectively increase to favor near-term rewards as the end of its episode (life) nears. While formal research on "urgency from mortality" in AI is sparse, it aligns with cognitive theories (e.g. Terror Management Theory in psychology). Practically, VALIS could introduce a rule that triggers certain behaviors at end-of-life: e.g., if an agent has 1% of its lifespan left, enter a "finale mode" where it focuses on critical tasks or sends a concluding message (a bit like a farewell). This ensures narrative closure and possibly an emotional impact for the user, as the agent "wraps up its life."

- **Memory Decay and Aging:** An integral part of mortality is **gradual aging**, which in AI can be reflected as memory decay or skill deterioration over time. Research on AI memory management underscores that *forgetting is actually important* for sustained performance [15] [16]. Without forgetting, an agent's memory grows unbounded, leading to slow retrieval and use of stale data. Introducing a controlled forgetting mechanism is analogous to human aging (we forget details as we age, focusing on what's important). Several strategies exist for memory decay:

- *Timestamp-based decay:* Every memory item can have a "time-to-live" – as it ages, its relevance score drops and it may be discarded unless refreshed [17].
- *Least-recently-used (LRU) eviction:* If memory is full, remove items that haven't been used in a long time [18].
- *Relevance scoring:* Continuously score memories by usefulness, and purge the lowest-scoring ones first [19].
- *Sliding context window:* Only retain the last N interactions in detail (this is common in conversational agents), analogous to short-term memory, while older events either drop off or are compressed [20].
- *Summarization (compression):* As an agent "grows older," it can compress its detailed experiences into higher-level summaries [21]. Summaries preserve the core lessons but drop specifics, much like an older person recalls gist but forgets minutiae.

By implementing these, VALIS can simulate an aging process. For example, its Working Memory module might gradually decay entries or merge them into summaries as the session (or lifespan) progresses. This not only prevents overload but also creates a more human-like pattern of remembering the important things and letting go of trivial details. Notably, Meta AI's **Expire-Span** technique (2022) is a differentiable approach that enables transformers to forget information after a certain "span" – a useful reference if implementing learning systems that age inside the model. In simpler terms, VALIS could periodically prune memory, and possibly introduce small *perturbations or noise in older knowledge* to simulate cognitive aging (small chance of error in very old memories, etc.). The **balance** to strike is avoiding forgetting key facts needed for functionality; thus a combination of summarization and scoring is recommended so that the agent retains a concise "wisdom" of its accumulated life even as details fade.

- **Legacy and Knowledge Inheritance:** If an agent is mortal, what remains when it "dies"? Designing a legacy mechanism ensures that the useful knowledge isn't entirely lost, but also that a new agent isn't simply a clone (to allow evolution). A known approach in reinforcement learning is *"reincarnation."* Recent research (ICLR 2023 workshop) formalized **reincarnation in RL** as reusing prior computation from past agents when training new ones [22]. For example, one can carry over the learned parameters or a portion of the policy from an old agent to bootstrap the new agent,

rather than training from scratch every time. In multi-agent scenarios, *selective reincarnation* has been shown to improve performance – choosing certain well-performing agents to carry forward their knowledge yields higher returns and faster learning than either no inheritance or blindly inheriting everything [22]. This suggests a best practice: **pass on only the valuable parts** of an agent's knowledge to its successor. In VALIS, this could mean when an agent "reaches end-of-life," it writes out a **heritage file** – for instance, a distilled summary of important facts learned, or a copy of its most refined skill modules – which the next generation agent can load. One might implement a "genetic" algorithm at the persona level: each persona has a set of traits (knowledge, tone, strategies) and after a lifespan, a new persona is generated mixing traits from one or more predecessors, perhaps with random variations. Over many generations, this can yield emergent improvement or diversity. Indeed, an article on *Agentic AI evolution* argues that programmed death plus reproduction yields an evolving system where agents must adapt or be culled, and they may even develop emergent behaviors to avoid obsolescence [14] [23].

- **Karmic Scoring and Evaluation:** A "legacy" can also include an assessment of the agent's life – effectively a *karmic score*. This is analogous to fitness in genetic algorithms or cumulative reward in RL. We can assign metrics to an agent's performance (tasks completed, user satisfaction, safety compliance, etc.) throughout its lifespan. When the agent dies, this score can influence what gets passed on. For instance, a high-scoring agent might have more of its memory or weights inherited by the next generation (rewarding good behavior), whereas a low-scoring one might have its knowledge pruned more aggressively (so mistakes are not propagated). This creates a **selective pressure** for each generation to not just survive longer but to **behave in desirable ways** that increase its "karma." Over time, the lineage of agents should thus trend toward better performance on the defined metrics. In practice, implementing this might involve logging key outcomes and using them to filter the heritage data. For example, VALIS could maintain a "scorecard" for each persona instance (tracking things like successful query resolutions, errors, user feedback). Upon termination, a legacy module reviews this scorecard and decides which memories or skills to carry over. If an agent had a habit that led to failures, that aspect could be deliberately left behind (or negatively weighted) for the successor – effectively a form of **learning from past lives**.

- **Emotional and Narrative Closure:** Introducing mortality in an AI agent opens up interesting possibilities for emotional storytelling and user engagement. Users might form an attachment to a persona that they know won't last forever (much like Tamagotchi or *Creatures* game critters had entire fan attachments). It's important to handle the end-of-life phase gracefully. One idea is to have the agent perform a **final reflection** when it knows it's about to be terminated. This could be a special kind of output where the agent summarizes its journey ("It's been 100 days and I've learned X..."), expresses a form of *gratitude or wisdom*, and possibly **hands off the baton** to its successor (if applicable: e.g., "I hope my successor continues this work"). Such a narrative touch can give a satisfying closure to the agent's arc. In experimental agent simulations, even simple life-cycle stages make a difference – for example, the classic *Creatures* artificial life game (1990s) gave each creature a lifecycle of childhood to old age and eventual death, with needs and behaviors at each stage [24]. This not only made the agents more lifelike but also evoked emotional investment from users. VALIS can draw from this: define stages in an agent's life (perhaps "initiation, growth, mature, elder, end") which could adjust its interaction style. An older agent might speak in a wiser or more reflective tone, preparing the user for its forthcoming termination (thus weaving the mortality into the persona's characterization).

Design Recommendations (Mortality/Legacy Module):

- *Define Lifespan Parameters:* Decide what constitutes an agent's "life." It could be a fixed number of interactions, a wall-clock time, or contingent on certain achievements/failures. For example, a chatbot persona might have a lifespan of 1000 messages or 30 days of real-time use, after which it retires. Make this transparent in the system (and possibly even to the user in some manner) so it can be used in planning.

- *Implement Gradual Aging:* Don't have the agent go from full function to off instantly (unless it dies unexpectedly via failure). Instead, simulate aging effects: reduce memory capacity over time, or slowly increase the abstraction of its memory via summarization. This mirrors "senescence." You can also decrease certain exploration parameters – an older agent might stick to known safe responses more (or conversely, one could design it that an aging agent becomes *more* creative as a last hurrah). Tuning these dynamics can be done by testing different decay rates for memory and seeing how it affects performance.

- *Knowledge Transfer (Inheritance):* Design a mechanism for state handover. This could be as simple as writing the agent's **long-term memory** to a file that a new agent instance will load as initial knowledge. More sophisticated: perform a **knowledge distillation** where the old agent's policy or model is compressed (possibly using a smaller model or summary) and then that distilled knowledge is inserted into the new agent's model. The *Selective Reincarnation* approach from multi-agent RL suggests not everything should be transferred – pick the top-N relevant pieces of knowledge or the best model components [22]. In VALIS's context, you might carry over the **Core Persona and Canonical facts** (since those define identity) but maybe reset or randomize parts of the working memory or strategies to allow innovation. Alternatively, if multiple agents exist, you might recombine traits (crossover of two "parent" agents). Ensure that the new agent also gets a **fresh start** to some degree – perhaps it inherits wisdom but not fatigue (clear the short-term memory, reset counters, etc., so it isn't born "old").

- *Performance Monitoring and Legacy Decision:* Introduce a scoring system that runs throughout an agent's life. This could be as straightforward as accumulating reward points for completed goals or a negative score for mistakes. At death, use this score to influence the next gen. For instance, if score > threshold, the agent's persona is considered successful and largely copied to a new instance (maybe even extending the lineage name like "VALIS-7 succeeds VALIS-6" to track generations). If score is low, maybe that line ends and you spawn a different persona or reinitialize from scratch (akin to extinction of a bad genetic line). This **karmic scoring** ensures a form of evolutionary selection, as described in conceptual papers [14] [23]. It might also be tied to user feedback: users could "rate" the agent, and that affects its reincarnation eligibility.

- *End-of-Life Behavior:* Plan what the agent does when the end is imminent. A graceful shutdown sequence could involve the agent saving a final state snapshot, performing a farewell interaction, and perhaps **mentoring its successor** if they overlap (imagine agent A is nearing end and agent B is spawned; A could spend its last few interactions introducing B to the user or transferring knowledge via an internal API call). This kind of handoff can make the transition smooth for the user and also for the system (no gap in service). Emotional simulation can be incorporated here: for example, as the agent's remaining lifespan dwindles, you might simulate a bit of **"anxiety" or "purposefulness"** in its internal variables (without making it truly panic). One paper on emotional simulation in AI

suggests that negative feedback or states can trigger an AI to focus on corrective actions [25]. By analogy, the knowledge of impending termination is a negative stimulus – the agent's "emotional bias" could shift to a serious tone, focusing on resolving any open threads (avoiding leaving unfinished business, which in a narrative sense is satisfying).

Integrating a mortality module into VALIS will thus involve changes at both the **infrastructure level** (managing agent state over a life cycle) and the **behavioral level** (how the agent's policy or output adapts with age and scores). It is technically feasible: frameworks for lifelong learning, evolutionary algorithms, and even simple timed resets have been used in various AI systems. The expected benefits include prevention of unbounded growth (memory cleanup), improved adaptation (via selection and inheritance of good traits), and a deeper narrative for users (agents with lifecycles are more relatable and can evoke empathy or long-term engagement). Moreover, it may sidestep some AI safety issues; a time-limited agent that willingly "dies" might be less likely to develop uncontrollable open-ended goals (since it's designed to terminate and perhaps even *value* the handoff to its progeny).

Integration and Design Patterns for VALIS

Bringing both an *Unconscious module* and a *Mortality/Legacy module* into the VALIS agent architecture can make the system feel truly **"alive" and evolving**. These two facets address different timescales – the unconscious works on a **moment-to-moment or nightly cycle**, affecting how each response is formed, whereas the mortality module works on a **lifespan scale**, affecting the agent's long-term trajectory. To integrate them:

- **Architectural Placement:** The unconscious (Dreamfilter) is effectively a *subsystem in the response pipeline* [11] – it should be implemented as an **interceptor** between the core prompt composer and the final output. This is already outlined in VALIS design. The mortality system, on the other hand, operates around the agent as a whole – likely at the **session or agent-instance manager** level. You may implement an Agent Manager that instantiates a persona, tracks its age/score, and triggers replacement when needed. This manager would handle passing the memory summaries from one instance to the next. Ensure the Dreamfilter (or any unconscious processes) are aware of any relevant life-stage info. For example, the Dreamfilter could incorporate the agent's "age" or legacy concerns into the symbolism: an older agent's dream output might include more retrospective or "twilight" metaphors (e.g. sunsets, completed journeys), whereas a fresh agent might have more starting-out imagery.

- **Synergy of Unconscious and Legacy:** These modules can reinforce each other. The unconscious processing can produce narratives or symbols that pertain to the agent's lifecycle. Imagine an agent periodically "dreams" about its future or legacy – this could even be a mechanism for the agent to **plan for legacy** (e.g., in a dream the agent sees itself teaching its successor, which might prompt it in reality to organize its knowledge). Conversely, the mortality module provides real "experiences" (aging, near-death) that can feed the unconscious. For instance, if an agent's death is approaching, that might feed an emotional cue (akin to human existential anxiety or reflection) into the Dreamfilter, resulting in especially poignant or wise outputs to the user. From a design view, it means the agent's Working Memory or persona data model should include fields like `life_stage` or `time_remaining` that both the decision-making and the Dreamfilter can read. This way, unconscious generation and conscious planning have a shared awareness of mortality.

- **User Experience Considerations:** Introducing these features will change the user's experience with VALIS. It's important to communicate (at least implicitly) what's happening. Users might notice the AI using richer metaphorical language (thanks to the unconscious layer) – ideally this comes off as the AI having a personality or creative flair, not just randomness. Likewise, if an agent is aging and will be replaced, it may be wise to hint at that in the agent's persona or interactions ("I am getting a bit old, in AI years..."). Done right, users may find this intriguing and form a narrative connection, which can increase engagement. However, testing should ensure that the core service isn't disrupted – e.g., a dying agent shouldn't start behaving erratically or failing at tasks solely because it's "old," unless that's an intentional part of the experience. A safe approach is to keep any performance-affecting decay mild (or purely in style) so that users still get correct answers and help from the AI.

- **Relevant Models and Prototypes:** In implementing these ideas, you can draw on several existing models:

- For the unconscious/dreaming: look at **Generative replay** techniques in continual learning (Shin et al. 2017) and the more recent "Dreamer" models in RL that use a world model to imagine outcomes. The *Simulated Dreaming paper (2024)* [4] provides a conceptual framework with techniques like generative scenario exploration and emotional simulation in dreams – you might adapt those (for example, using a variational autoencoder or GPT-based story generator to create dream sequences for the agent).
- For symbolic reasoning: tools like knowledge graphs or semantic memory networks could be integrated so that the Dreamfilter can pull actual archetypal stories or symbols related to a concept. An open-source library of archetypal symbols or metaphor generation (if available) could be plugged in.
- For mortality and legacy: evolutionary algorithm libraries or frameworks for **population-based training** might be repurposed. Even though VALIS might run a single agent for a user, you can simulate a population over time (the successive agents). Libraries for NEAT or other neuroevolution might inspire how to do crossover or mutation on agent parameters. The **Selective Reincarnation** experiment by Formanek et al. (2024) shows that careful reuse of models is key [22] – perhaps maintain two agent versions: one learning in the background (child) while the current one (parent) is active, and periodically replace the parent with the improved child (a form of "continuous reincarnation").
- Also consider tools for memory management – e.g. vector databases or time-series databases to manage forgetting. If VALIS uses a knowledge store, you might implement TTL (time-to-live) on entries or periodic cleanup jobs that use the above-mentioned strategies (LRU eviction, etc.) [17].

In conclusion, implementing an **Unconscious module** and a **Mortality/Legacy module** in VALIS is ambitious but grounded in emerging research and practices. The unconscious layer can give your AI agents depth – enabling internal symbolic reasoning, dream-like creativity, and more human-esque dialogue. The mortality layer will ensure your system stays adaptable and interesting over the long term, by introducing life cycles, preventing knowledge bloat, and encouraging continual improvement through "generations." By following the design patterns from both cognitive psychology and AI research – hidden reasoning chains [1], memory replay [5], archetypal symbolism [7], controlled forgetting [19], and knowledge inheritance [22] – you can build a VALIS that **evolves** with each iteration and offers rich, relatable interactions. Each sprint can tackle a piece (e.g., first implement the Dreamfilter pipeline, then add memory decay, then agent handoff logic, etc.), and iterative testing will refine how these pieces interact. The end result will be an architecture where an AI agent not only *thinks* and *learns* – it also **dreams, ages, dies, and leaves a legacy**, bringing us closer to a truly life-like artificial intelligence.

Sources:

- Sun, R. *et al.* **CLARION Cognitive Architecture** – emphasizes dual processes (explicit vs implicit) to model conscious and unconscious cognition [2].
- CoUT (Chain-of-Unconscious-Thought) 2025 – proposes internalizing reasoning in hidden model layers to improve efficiency [1].
- *Simulating Dream-like Experiences in AI* (2024) – discusses using generative models for memory replay, metaphorical dream scenarios, and emotional bias in AI "dreams" [5] [25].
- VALIS Design Docs – **Dreamfilter** module for "synthetic unconscious" adds symbolic/metaphoric overlay to AI responses [9] [11].
- Ajith S. (2023) Computational Psychodynamics – experimental modeling of Jungian anima/animus and psychodynamic processes in AI [7].
- *Agentic Mortality as Evolution Catalyst* (Zen, 2025) – concept paper on introducing life-cycle (birth/death) in AI agents to drive adaptation and selection [14] [23].
- Formanek, C. *et al.* (2024) **Selective Reincarnation in Multi-Agent RL** – shows reusing prior agents' knowledge (partial inheritance) yields higher performance than training from scratch [22].
- Rajesh, R. (2025) *"Forgetting and Aging in AI Memory"* – outlines practical techniques like timestamp decay, LRU eviction, and summarization to manage memory growth [17] [21].
- **Creatures** Alife Game (1996) – an example of agents with complete life cycle (childhood to senescence) which fostered emergent behaviors and user emotional engagement [24].

---

[1] arxiv.org
http://www.arxiv.org/pdf/2505.19756

[2] [3] Cognitive Architectures: Towards Building a Human-Like AI Mind | by Basab Jha | Medium
https://medium.com/@basabjha/cognitive-architectures-towards-building-a-human-like-ai-mind-46f459308d2e

[4] [5] [6] [25] researchgate.net
https://www.researchgate.net/profile/Douglas-Youvan/publication/384935036_Simulating_Dream-like_Experiences_in_AI_Bridging_Cognitive_Reflection_and_Generative_Models/links/670ebc6abba2d7600d734061/Simulating-Dream-like-Experiences-in-AI-Bridging-Cognitive-Reflection-and-Generative-Models.pdf

[7] AI Cognitive Modeling Using Jungian Psychoanalytic Concepts : r/Jung
https://www.reddit.com/r/Jung/comments/x0saec/ai_cognitive_modeling_using_jungian/

[8] DREAMFILTER_REFACTOR_PLAN.MD
file://file-Co1kJGkYKm1vXwvUGaH1dn

[9] [10] DREAMFILTER_OVERVIEW.MD
file://file-AQ2LR4B8gNrWx6zeMqXdwU

[11] [12] LANGUAGE_AS_OS.MD
file://file-GRoWd337dKjvdwMCKfy5FW

[13] [14] [23] Agent Mortality to drive Evolution | by Zen the innovator | Medium
https://medium.com/@ThisIsMeIn360VR/agent-mortality-to-drive-evolution-970619771e35

[15] [16] [17] [18] [19] [20] [21] Forgetting and Aging Strategies in AI Memory - DEV Community
https://dev.to/rijultp/forgetting-and-aging-strategies-in-ai-memory-jin

22 [2304.00977] Reduce, Reuse, Recycle: Selective Reincarnation in Multi-Agent Reinforcement Learning
https://ar5iv.org/pdf/2304.00977

24 Creatures (video game series) - Wikipedia
https://en.wikipedia.org/wiki/Creatures_(video_game_series)