# **Comparative Analysis of the Governing Faculty Blueprint (NES) and Contemporary Cognitive Frameworks**

## **Introduction**

The Governing Faculty Blueprint, also known as the Normative Executive System (NES), is a proposed cognitive architecture inspired by the ancient Stoic idea of the hegemonikon or “ruling faculty” – the rational executive of the mind that receives impressions and gives or withholds assent to impulses. In modern terms, NES is conceived as “a single, integrated governance process that receives all incoming impressions and impulses, normatively evaluates them against internalized standards, grants or withholds assent to action, and recursively corrects itself when inconsistencies or errors are detected” . This model aims to fulfill the Stoic picture of decision-making (impressions → assent by ruling faculty → action) using computational and neurocognitive tools. Crucially, NES is not a homuncular “little man” in the head but an emergent system of interacting components (a comparator, an Assent Gate, a conflict resolver, etc.) that together implement executive control. By engineering the hegemonikon as a set of mechanisms, the Blueprint attempts to “demystify” the executive self and make it implementable in a brain or AI.

This analysis will rigorously compare the NES framework against several major contemporary frameworks in cognitive science, neuroscience, artificial intelligence, philosophy of mind, and psychology. We focus on 5–8 influential models that offer contrasting perspectives on how the mind’s “executive” functions operate. In particular, we will examine:

* Global Workspace Theory (GWT) – a cognitive neuroscience model of a global “workspace” for integrating and broadcasting information (e.g. Baars’ and Dehaene’s work on conscious access).
* Predictive Coding and Free Energy Principle – the theory that the brain is a hierarchical prediction engine minimizing prediction error (Friston, Clark), emphasizing distributed processing.
* Dual-Process Models – psychological models distinguishing intuitive/automatic versus analytical/controlled processes (e.g. System 1 vs. System 2 in reasoning).
* Higher-Order Thought (HOT) Theories – philosophical models of consciousness where a mental state is conscious only if a higher-order representation of that state exists (e.g. Rosenthal’s theory).
* ACT-R and SOAR Cognitive Architectures – symbolic AI models of the mind that implement a central executive through production rules and modules (John Anderson’s ACT-R, Allen Newell’s Soar).
* Hierarchical Reinforcement Learning (HRL) – AI frameworks that structure control in layers of managers and sub-agents optimizing rewards (options framework, meta-controller models).
* Self-Regulation and Executive Function Theories – psychological and neurobiological accounts of cognitive control, such as Norman & Shallice’s Supervisory Attentional System, conflict monitoring/ACC models, and related theories of self-control and metacognition.

We will contrast NES with each of these frameworks along several key comparison dimensions that highlight their differences and unique contributions. These dimensions include the nature of central control (unitary vs. distributed), the clarity of their computational specification, how they handle norms/values and decision gating (assent or inhibition of impulses), mechanisms for resolving conflicts (and whether they use recursion or hierarchy), integration of emotion and value with cognition, capacity for learning/adaptation, approaches to the homunculus problem (i.e. avoiding an infinite regress of inner agents), and empirical support/implementability. The goal is to provide a comprehensive, structured analysis of where NES aligns with or deviates from each framework, and to identify NES’s unique contributions in synthesizing or innovating beyond the state of the art.

## **Methodology: Key Comparison Dimensions**

To systematically compare NES with other frameworks, we define the following comparison dimensions. These dimensions target fundamental aspects of an executive system, ensuring an apples-to-apples evaluation of how each framework conceives of cognitive control, decision-making, and normative guidance:

* 1. Nature of Central Governance (Unitary vs. Distributed): Does the framework posit a single, unitary “executive” agent or module at the top of the cognitive hierarchy, or is control an emergent property of many distributed processes? In other words, is there an explicit central governor or does coordination arise from decentralized interactions? This dimension illuminates how each model deals with the idea of a central will or control state and whether it risks a homunculus. NES, for example, reframes the unitary Stoic hegemonikon as an emergent process of multiple sub-components rather than an indivisible soul.
* 2. Algorithmic Specificity and Implementability: How explicitly is the framework specified in algorithmic or architectural terms? Can it be concretely implemented as a computational model or is it primarily a conceptual description? We examine the clarity and technical detail of each theory – e.g. mathematical formalisms, flowcharts, code – and its potential for computational implementation (in AI or as a neural model). NES provides high specificity (e.g. a drift-diffusion evidence accumulator, threshold parameters, and rule-based conflict resolution), aiming to “break down” executive functions into mechanistic steps rather than abstract homunculi. Other frameworks vary from detailed cognitive architectures (ACT-R, HRL) to broad conceptual metaphors (some dual-process and HOT models).
* 3. Treatment of Norms and Values: To what extent does the framework incorporate normative criteria – such as goals, values, moral norms, or task objectives – in its decision-making process? Does it have an explicit representation of internalized rules or preferred outcomes that guide choices? NES is defined as a normative system: it contains a Norm Repository of internalized principles and uses these to evaluate and constrain impulses . We will see that many traditional models (e.g. GWT, HOT) are value-neutral descriptions of information flow or consciousness, whereas others (e.g. reinforcement learning, certain executive function theories) incorporate goals or rewards which can be seen as proxies for values. This dimension highlights how well each framework accounts for why a cognitive system chooses one action over another beyond just mechanistic triggers – i.e. the role of ideals, preferences or learned standards.
* 4. Assent and Withholding Mechanism (Executive Gating): Does the framework include a mechanism analogous to assent – an executive veto or approval that gates whether an impulse leads to action? In Stoic terms, this is the faculty’s ability to withhold consent from a desire. NES explicitly models this via the Assent Gate, which sets a threshold for triggering actions and can actively veto impulses that do not meet normative criteria . We will compare how other frameworks account for inhibitory control or gating: for example, the “central executive” in some models can suppress automatic responses (as in Norman & Shallice’s theory), and dual-process accounts imply that deliberation can override impulsive responses. This dimension examines each model’s capacity to say “no” to an urge – a critical function for self-control and moral agency – and how that is implemented or explained.
* 5. Conflict Resolution and Recursive Processing: How are conflicts between competing inputs, goals, or rules handled? If two processes or motives advocate different actions, does the framework provide a method to resolve this conflict (e.g. weighted competition, a priority rule, a meta-decision)? Moreover, does it allow recursive or iterative re-evaluation (i.e. a feedback loop to resolve difficult stalemates or errors)? NES features a Norm Conflict Resolver that performs weighted voting among norms with possible veto rules , and a Recursive Adjudication process (RAA) that can iterate decision cycles when initial outcomes are indecisive or conflicting . Some frameworks similarly have hierarchical or iterative conflict resolution (e.g. Soar’s subgoaling when impasses occur, or predictive coding’s iterative error minimization), while others may simply pick the strongest impulse or have less structured resolution. We will assess if each model can handle complex internal conflicts (such as moral dilemmas) and whether it employs any form of self-refinement loop to resolve them, as NES does by “thinking twice” on unresolved conflicts .
* 6. Integration of Cognition, Emotion, and Value: Does the framework integrate emotional and value-based signals with cognitive processing in a unified architecture, or does it treat them separately (or not at all)? Human decision-making is influenced by affect and reward values as well as factual reasoning. NES attempts to integrate these by modulating decision thresholds with emotion-like signals (e.g. a serotonin-analogue for restraint, fear or anger adjusting caution) and by weighing impulses against value-laden norms. We will compare how each framework deals with the interplay of cold cognition vs. hot emotion/motivation. For instance, dual-process models explicitly separate an intuitive/emotional system and a rational one; predictive coding accounts can encompass emotion as predictions of bodily states; executive function theories distinguish “hot” vs “cool” control. This dimension shows whether a model can account for phenomena like emotional bias in decision, moral sentiments, or context-dependent threshold changes (e.g. being more impulsive under threat), all of which NES attempts to capture.
* 7. Learning, Adaptability, and Trainability: How does the framework account for learning and adaptation of the executive control processes? Can the system acquire new rules or adjust its control parameters through experience or training? NES is designed to be adaptive – it includes mechanisms for norm acquisition (e.g. learning from authority instructions or from social reward/punishment feedback) and can tune its control thresholds based on context (e.g. urgency, stakes, past failures) . We will review whether each framework has a learning component: e.g. ACT-R learns production utilities and chunked rules; hierarchical RL obviously learns policies from reward; predictive coding continuously updates its internal model; whereas some conceptual frameworks (HOT, some dual-process formulations) have no explicit learning rules. We also consider intervention points: places one could externally tweak or train the system (for example, therapy or training regimes that strengthen executive control, or in AI, adjusting reward functions or adding rules). This dimension addresses each model’s flexibility over time and how new information or normative guidelines can be incorporated.
* 8. Homunculus Problem and Self-Modeling: This overarching dimension examines how each framework addresses the classic challenge of explaining executive control without simply positing another “little mind” inside (avoiding infinite regress). Does the model inadvertently rely on an undefined inner agent, or does it provide a clear decomposition of control tasks? Additionally, does it include a self-monitoring or self-model component that the system uses to regulate itself (e.g. a model of its own state or limitations)? NES explicitly tackles the homunculus critique by “specifying mechanistic processes for each executive function, thus ‘breaking down the homunculus into an army of simpler processes’”. The NES’s executive is not a single ghostly decider but an emergent result of “dumb” pieces (accumulators, comparators, etc.) orchestrated by structured rules. It also features a form of self-monitoring: it watches for conflict signals (analogous to the brain’s ACC) and uses a feedback loop (RAA) to adjust control, “akin to a thermostat”, without requiring a mysterious self to intervene. We will analyze how other frameworks avoid (or fall into) homunculus explanations – for example, GWT distributes control via competition for a global workspace, predictive coding has no central agent at all, and Norman & Shallice’s SAS was criticized as a homunculus until specified otherwise. We will also note if a framework offers an explanation for the phenomenology of having a central self (e.g. Higher-Order theories explicitly attempt to explain the feeling of a self by positing a meta-representational self-model, and the Attention Schema Theory posits the brain’s internal model of attention as the basis of the self). This dimension is closely tied to theoretical/philosophical coherence of the model of agency.

(Each of these dimensions will be explored in detail for each framework in the following sections.)

## **Frameworks Overview**

Before diving into the detailed comparison, we provide a brief overview of each framework to clarify their core ideas and context:

* Global Workspace Theory (GWT): Originally proposed by Bernard Baars and later expanded neurally by Stanislas Dehaene, GWT likens the mind to a theatre. Many unconscious processes compete for access to a limited-capacity “global workspace” (spotlight of attention), and the information that wins is broadcast globally to many brain systems, becoming conscious and guiding action. GWT emphasizes a central information hub (the workspace) that enables integration and coordination across specialized modules. It is often associated with a “central executive” that directs attention, though modern GWT/GNW (Global Neuronal Workspace) implementations distribute that function across frontoparietal networks. The theory is influential in explaining conscious access and executive attention, and it predicts signatures like widespread brain activation for conscious stimuli.
* Predictive Coding & Free Energy Principle: A theoretical framework (Friston, Rao & Ballard, Clark) that models the brain as a hierarchical Bayesian predictor. Each level of a cortical hierarchy generates predictions about lower-level inputs; mismatches (prediction errors) are fed forward for correction. The system aims to minimize “free energy” or surprise by constantly updating its internal model. In this view, perception, cognition, and action are all forms of prediction error minimization. There is no single executive agent; rather, “control” emerges from the dynamic weighting of prediction errors (which can be modulated by precision estimates akin to attention). The framework is highly computational (formal equations for updates) and has been used to model everything from visual perception to action planning (active inference). It provides a unifying theory for brain function, though it’s abstract with respect to psychological concepts like “will” or “norms.”
* Dual-Process Models: A family of models in psychology (e.g. Stanovich & West, Kahneman, Evans) that distinguish between two modes of processing: a fast, automatic, intuitive mode (often called System 1) and a slow, deliberative, rational mode (System 2). System 1 includes heuristics, habits, emotional reactions; System 2 includes analytical reasoning, rule-based thought, and effortful control. These models explain phenomena like cognitive biases (System 1 yielding a quick but sometimes wrong intuition, which System 2 might override if engaged). They are more of a descriptive framework than a single architecture – typically, no single “homunculus” is identified, but System 2 is implicitly the supervisory process that can intervene on System 1 outputs. In moral psychology, for example, Greene et al. have framed deontological judgments as System 1 (emotion-driven) and utilitarian judgments as System 2 (reasoned), with response conflicts between them. Dual-process theories have broad empirical support in behavioral data, though they often lack a detailed algorithmic implementation of how the two systems interact.
* Higher-Order Thought (HOT) Theories of Consciousness: A set of theories in philosophy of mind (notably Rosenthal’s HOT theory) proposing that a mental state is conscious only if accompanied by a higher-order representation (thought) that one is having that state. For instance, a first-order perception of a tree becomes a conscious experience of “seeing a tree” if one’s mind also has a (possibly unconscious) thought that it is seeing a tree. The “executive” here is not about control of action but about the presence of a meta-level awareness that confers the sense of a conscious self. HOT theories address the self-model aspect: they imply the mind has the capacity to represent its own states (a kind of internal observer). However, they are not detailed computational models of decision-making or control; they mainly aim to explain conscious awareness. Variants include Higher-Order Perception (HOP) theories and the Attention Schema Theory (which in a similar spirit suggests the brain’s internal model of attention produces the feeling of subjective experience). These theories are more conceptual/ philosophical, typically silent on how choices are made or how norms factor in – their focus is explaining the phenomenology of a central self or awareness.
* ACT-R / Soar Cognitive Architectures: ACT-R (Adaptive Control of Thought – Rational) and Soar are classical cognitive architecture frameworks from AI and cognitive psychology, attempting to model human cognitive abilities in a unified system. ACT-R (J. R. Anderson) consists of multiple modules (for vision, memory, goals, etc.) whose information is brought together by a central production system. Productions (if-then rules) fire serially, and a conflict resolution mechanism (based on utility values or specificity) selects which production runs next when multiple are applicable. ACT-R is highly specified (there is a reference implementation) and has been used to simulate human performance (even mapped to brain regions for each module). It includes a goal buffer to represent the current intention and has mechanisms for learning (e.g. adjusting production utilities via reinforcement learning, and chunking new rules). Soar (John Laird, Allen Newell) is another architecture with a similar production-system core; a key feature is that if Soar cannot decide (an impasse), it creates a subgoal and uses problem-solving (potentially recursively) to resolve it, then learns from that via chunking. Both ACT-R and Soar therefore have a unitary but mechanistic executive: a single rule-based decision process that governs action selection, yet it is algorithmic (no mystery homunculus, just programmed rules and learning). They do not inherently encode “norms” or emotions, but they allow modeling of goals/priorities through their goal structures and utility values. These architectures have been extended by researchers to incorporate emotional parameters or moral decision rules, but in their standard form, they focus on cognitive tasks (e.g. solving a puzzle, recalling a fact) under a rational task-oriented goal.
* Hierarchical Reinforcement Learning (HRL): In AI and computational neuroscience, hierarchical RL refers to models where control is layered: higher-level policies decide abstract actions or subgoals, which are then carried out by lower-level policies. This addresses complex tasks by breaking them into subtasks. Examples include the options framework (Sutton et al.), where an agent has available “options” (multi-step actions) and a policy to choose among them, or feudal RL (Dayan & Hinton) with manager-worker relationships. A contemporary variant is deep RL agents with hierarchy or meta-controller modules (e.g. DeepMind’s hierarchical DQN or policy networks that output subgoal representations). In these frameworks, there isn’t a single permanent “executive”; instead, each level’s policy serves as the executive for the level below. A top-level policy might choose a goal (analogous to an executive decision), but it is itself just another learned function maximizing cumulative reward. Values enter naturally as reward functions – the agent’s objective – but these are typically task-specific (e.g. maximize points, minimize time) and not normative in the human sense unless the reward is shaped to encode norms. HRL provides clear algorithms (it’s a subfield of machine learning with mathematical formalism) and is fully implementable. It also raises interesting points about conflict (if sub-policies suggest different actions, the higher policy’s choice prevails) and learning (the agent must learn both low-level skills and the high-level policy). Comparatively, HRL is a strong model for how complex skills can be organized, but it does not explicitly address conscious oversight or “vetoing” actions except insofar as higher layers can choose to not invoke certain lower options. It’s a distributed control scheme in a sense, though often the hierarchy is pre-designed (which could be seen as an external homunculus deciding the task decomposition).
* Self-Regulation and Executive Function Theories: This category includes psychological and neurocognitive models of how people control their behavior, attention, and impulses. One influential model is Norman & Shallice’s Supervisory Attentional System (SAS), which posits that while many actions are run by automatic schemas (habits) via a contention scheduling mechanism, a higher-level SAS can intervene in non-routine situations to bias or inhibit the automatically activated schemas. SAS is essentially a top-down attention controller – conceptually like a central executive – that ensures behavior aligns with goals when autopilot would be inadequate. Another line of work is the conflict monitoring model (Botvinick et al.), where the anterior cingulate cortex (ACC) detects conflict (e.g. between responses in a Stroop task) and signals the need for increased control, which the prefrontal cortex then implements – a feedback loop regulating control allocation. This led to the Expected Value of Control (EVC) theory (Shenhav, Botvinick, Cohen), which proposes that the brain computes the cost vs. benefit of exerting control and engages control accordingly; it’s an attempt to formalize SAS’s intervention in rational terms. In terms of emotion, there are concepts of “hot” vs “cool” executive function – “hot” involving emotion-laden decisions (orbitofrontal cortex) and “cool” being purely cognitive (dorsolateral PFC). Also relevant are concepts like willpower or self-control strength (Baumeister’s ego depletion model, though empirically contested) and metacognition, where one monitors and regulates one’s own cognitive processes. These theories, taken together, often assume some centralized executive function system in the frontal lobes, but modern approaches try to break it into specific components (inhibition, updating working memory, shifting attention) rather than a single entity. They have extensive empirical backing in neuropsychology (e.g. patients with PFC damage have dysexecutive syndrome, deficits in planning, inhibition, etc., supporting the idea of an executive system in the brain). However, many of these theories are descriptive or computational only in narrow scopes (e.g. mathematical models of conflict monitoring exist, but there isn’t a single unified algorithm comparable to ACT-R or an RL agent that captures all executive functions). They also typically don’t explicitly include moral norms—though “internalized standards” are often discussed in self-regulation (e.g. psychology of conscience, or the idea that people have personal goals and ideals that guide self-control). Thus, they cover the functions NES is interested in (like inhibition, conflict resolution, goal maintenance), but usually not in a single integrated model with normative evaluation.

With these frameworks introduced, we now proceed to compare each of them with the Normative Executive System along the defined dimensions, highlighting both similarities and divergences.

## **Detailed Comparison by Dimension**

### **1. Nature of Central Governance: Unitary vs. Distributed Control**

NES (Normative Executive System): NES is explicitly designed as a distributed yet integrated control process rather than a single monolithic decider. The blueprint describes NES as an emergent coordination of sub-components (comparator, Assent Gate, conflict resolver, etc.) that collectively implement executive functions. There is no singular “boss” module with free rein; each part has a defined simple role (accumulating evidence, gating output, arbitrating norms) and the ruling faculty emerges from their interaction. This is in contrast to the Stoic hegemonikon’s unitary soul – NES reconceptualizes that unity as arising from structured interactions of parts. By organizing “armies of such idiots to do the work” (in Alan Turing’s spirit), NES claims to avoid positing a little person in the head. In essence, NES’s governance is functionally unified (it produces a single decision outcome at a time) but structurally distributed. It does map to a kind of hierarchy – e.g. RAA (Recursive Adjudication Algorithm) overseeing lower processes – but even RAA is a rule-based meta-process, not an autonomous homunculus.

Global Workspace Theory: GWT posits a central workspace that has a functional singularity: at any moment, one coalition of information is “in the spotlight” globally. In that sense, it has a temporary unitary focus (the content in the global workspace) which can be seen as akin to a central stage of processing. However, GWT does not identify a persistent single executive agent. Instead, multiple unconscious processors compete or cooperate to put content into the global workspace; once there, that content is broadcast to the rest of the system . The “executive” in GWT is often identified with attention control mechanisms that bias this competition – e.g. top-down attention from frontal regions helps select the winner. In Baars’ original theory, a Contextual Supervisor might shape the competition, sounding like a homunculus, but in modern neuroscience implementations (Global Neuronal Workspace), this is realized by distributed circuits (e.g. dorsal attention networks, basal ganglia gating). Thus, GWT offers a compromise: it has a central bottleneck (only one conscious content at a time, implying a momentary unitary control of action/thought), but the control of that content is distributed across many modules. Compared to NES, GWT lacks a permanent central decision module – the unity comes from the broadcasting mechanism. NES aligns with GWT in that when NES reaches a decision (approving an impulse), that decision and its justification could become the globally broadcast content of consciousness . In fact, the Blueprint suggests the NES’s output corresponds to the conscious will entering the global workspace . In short: NES and GWT both affirm a unified decision moment, but NES explicitly details an internal decision governance system leading up to it, whereas GWT largely abstracts that as “whatever wins the competition gets broadcast.” GWT’s governance is thus more emergent and distributed in competition, whereas NES injects a structured arbitration process before global broadcasting.

Predictive Coding: Predictive coding frameworks feature fully distributed governance. Control emerges from the continuous reciprocal interaction of predictions and prediction errors across hierarchical levels; there is no single executive module deciding what the system does. Each level tries to minimize its local error, and high-level predictions can suppress or modulate lower-level activity by providing top-down expectations. In this paradigm, every unit is “mindless” (just computing errors or predictions), and intelligence arises from the network dynamics – a principle quite consonant with NES’s philosophy of breaking tasks into simple units. However, predictive coding doesn’t carve out a specific executive function or “central controller”; instead, it might interpret executive control as just another prediction task (e.g. predicting the need for control, or having a high-level prior that one should behave according to certain rules). Some active inference models incorporate something like “policies” as part of the hierarchical model – effectively giving a probabilistic form of action selection – but it’s still distributed (no single policy node that governs all). NES vs Predictive Coding: NES does propose a top-level process (RAA) that oversees conflicts and termination of deliberation, which might look more centralized than anything in standard predictive coding. On the other hand, NES’s reliance on feedback loops and signal modulation has a flavor of predictive regulation – e.g. NES monitors if no decision is reached after a while (similar to noticing a persistent prediction error) and then adjusts parameters or loops, which is analogous to an error-driven update . But fundamentally, predictive coding is egalitarian in governance (any level could influence outcome if error is large enough, and the equilibrium of the whole system determines action), whereas NES imposes a decision hierarchy (impulses first accumulate evidence, then norms can veto, then a meta-layer can repeat or stop the process). Predictive coding would interpret something like “norms” as just priors that bias the predictions of reward or outcomes – it would not create a separate conflict resolution committee for them. Thus, predictive coding’s control is emergent and implicit, while NES’s control is explicitly structured (even though it’s multi-component).

Dual-Process Models: Dual-process theories often imply a sort of two-tier governance: an automatic pilot (System 1) and a supervisory system (System 2) that can step in. In many interpretations, System 2 is the unitary executive – essentially the mind’s CEO that can override or modulate the myriad heuristic responses of System 1. However, System 2 is not usually fleshed out as an algorithm; it’s more a placeholder for “all the brain’s resources for deliberate reasoning and control,” which in practice might correspond to frontal lobe functions, working memory, etc. This runs the risk of homunculus unless specified, but authors like Stanovich have broken System 2 into an “algorithmic mind” (basic cognitive computations like those needed for abstract reasoning) and a “reflective mind” (higher-level goal setting). Still, in everyday terms, dual-process evokes a single decider that can say stop or think harder. So dual-process models lean unitary at the top (System 2 = one thing in charge when active) and distributed at the bottom (System 1 = many intuitive subsystems). NES vs Dual-Process: NES actually provides a possible mechanism for System 2. For example, in moral decision scenarios, NES’s Norm Conflict Resolver and recursive deliberation mimic the interaction of an emotional intuition and a rational counter-consideration . The intuitive veto (deontological norm) and the deliberative evidence accumulation (utilitarian reasoning) compete, and NES might take extra time and cycles to resolve this – paralleling System 1 vs 2 conflict where the “executive” is engaged to adjudicate . NES doesn’t explicitly label one part “intuitive” and another “rational,” but functionally the Assent Gate with its threshold and the Norm Resolver could be seen as enacting the controlled, reflective process when simple automatic accumulation doesn’t yield a clear go. So NES’s architecture embeds a two-mode behavior: quick decisions when an impulse sails through with no conflicts (akin to System 1), versus slower recursive decisions when norms conflict (akin to System 2 engagement). Unlike generic dual-process theory, NES dissects the reflective layer into concrete operations rather than a singular will – so it is more distributed in implementation. Dual-process accounts, lacking a detailed model, implicitly centralize executive control in the vague System 2. NES provides that System 2 with a blueprint of interacting parts, aligning with dual-process outcomes but with a distributed internal design.

Higher-Order Thought Theories: HOT theories do not focus on control of behavior, so they don’t present an “executive” per se. Instead, they involve a meta-level representation (the higher-order thought) that confers awareness. In terms of governance, one might say HOT introduces a kind of inner observer – a higher-order system that monitors first-order mental states. This could be interpreted as a form of central agent (the thing that has higher-order thoughts about everything else). However, HOT theorists typically resist the homunculus label by insisting the higher-order thought is just another mental state generated by normal cognitive processes, not a persisting inner ego. There isn’t a centralized decision-making in HOT, just a dual-layer architecture (mental state vs. a meta-state about it). So if we talk about governance in HOT: it’s unitary in terms of awareness (the content of the HOT is a single integrated representation of the lower state, suggesting a unified self perspective), but parallel/distributed in generation (various brain systems produce lower states and also higher-order representations possibly distributed in prefrontal cortex, etc.). NES vs HOT: NES is about decision control, HOT about conscious awareness – different aims. NES does include a notion of self-monitoring (the system checking itself) which could be likened to a simplified higher-order representation of its decision process (the blueprint even notes an “attention schema”-like internal model of the decision state for self-regulation ). That is analogous to the spirit of HOT (a system modeling itself). But NES uses that self-model strictly for control (to detect conflicts, etc.), not to explain subjective experience. So governance-wise, HOT doesn’t give us an executive mechanism, and it remains agnostic or compatible with many control architectures. You could embed HOT on top of NES (NES could generate higher-order thoughts about its own states to produce conscious feeling). In summary, HOT posits a unitary self-model (the content of the higher-order thought is typically a single integrated sense of “I am doing/feeling X”), but it doesn’t manage actions. It avoids homunculus by claiming the higher-order thought itself need not be observed by yet another process (one-stop higher-order). NES avoids homunculus by mechanizing the decision process. They tackle different “central entity” problems – HOT for the self/experiencer, NES for the decider/controller. For our purposes, HOT doesn’t provide a model of executive governance, so it neither competes with nor informs the structure of NES’s governance, except to note that NES’s inclusion of self-monitoring aligns with the idea that an executive system can have a model of itself (attention schema) without invoking a soul.

ACT-R / Soar: These architectures implement a unitary executive in practice: a single production matching mechanism that determines the next action at each step. In ACT-R, at any moment one production rule fires (selected by conflict resolution if needed) – effectively a single-threaded cognitive controller. Soar similarly has a single decision procedure that can stack subgoals but ultimately one decides at a time. This unitary decision-making is mechanistic (algorithmic selection, not an obscure will), but it is still a bottleneck – a clear locus of control in the model. The architectures are modular (different knowledge sources), but not distributed in control: only the production system “chooses.” Compared to NES, ACT-R/Soar look more centralized. NES could be seen as adding a pre-processing layer before a final action execution, whereas ACT-R’s productions directly decide actions. However, ACT-R’s production system could be used to emulate NES: e.g. one could write production rules that check norms and only allow an action if no rules forbid it, etc. In other words, ACT-R is flexible enough that the modeler can incorporate distributed-like control (by encoding intermediate steps), but the framework itself doesn’t impose that – it gives a single sequence of rule firings. NES vs ACT-R: NES’s RAA+Assent Gate is conceptually similar to a cycle of rule-based evaluations culminating in either doing or not doing an action. The difference is NES explicitly separates components (accumulator vs gate vs resolver), whereas ACT-R lumps everything into the production rule engine (one could write a rule that says “if conflict X and norm Y then set goal to resolve conflict…” etc. – but that’s up to the modeler). So we might say ACT-R has a unitary control locus by design, and any distribution is simulated within the rule set, whereas NES has an explicitly multi-component control locus. Soar’s subgoal mechanism does introduce a hierarchical control: a meta-level is invoked on impasse. This is analogous to NES’s recursion. But again, Soar’s meta-level is basically the same rule engine operating at a goal of “resolve impasse,” so it’s the same central machinery applied recursively. NES in contrast delineates a distinct conflict-resolution module. In terms of homunculus: ACT-R/Soar consider the production system as the “executive,” but it’s fully specified (no mystery), so they claim no homunculus in principle – it’s just a program. NES similarly specifies its executive in parts. So all three avoid an unexplained decider by providing an algorithm. The key governance difference: NES’s emergent control vs. ACT-R’s single decision loop. Practically, ACT-R is centralized sequential processing; NES, while also sequential in outcome (one decision at a time), conceptually spreads the decision across parallel evaluators (accumulators, norm votes) that then feed into the final gate. Cognitive architectures historically assume serial bottlenecks (consistent with psychological refractory period, etc.), whereas NES is trying to marry parallel evaluation (multiple norms and factors at once) with a serial decision act. Thus, NES is a bit more hybrid on this dimension, but leans distributed relative to ACT-R.

Hierarchical RL: In HRL, control is hierarchically distributed. There isn’t one monolithic executive; instead, each level’s policy is like the boss of the level below, forming a chain of command. For example, a top-level policy decides “I want to achieve goal G” and delegates to a mid-level option which then invokes low-level actions. At each junction there is a decision-maker, but it’s local to that layer. In effect, HRL has multiple executives at different scales, each operating on their own timescale or abstraction. No single layer knows everything: the top doesn’t micromanage the details, the bottom doesn’t see the big picture. This is a distributed governance via delegation. If one considers the whole hierarchy, it forms a structured control system, but not a unitary agent in the classical sense. This resonates with NES in that NES also has layers (impulse evaluation, norm evaluation, meta-conflict resolution). One could map NES’s structure onto a two- or three-layer hierarchy: e.g. lower-level impulses propose actions, a higher-level normative layer decides whether to permit them (like a manager approving or vetoing subordinate suggestions). Indeed, NES’s Norm Conflict Resolver and RAA behave like higher-level policies that ensure the chosen action meets certain criteria, analogous to a top-level RL agent whose “reward” is alignment with norms rather than just immediate gains. However, standard HRL typically optimizes a single reward function consistently across layers (just decomposed for efficiency), whereas NES has a distinct notion of normative priorities that can override utility – more akin to multi-objective RL with a hard constraint. NES vs HRL: Both share the idea of hierarchical control, but differ in purpose. HRL’s hierarchy is about temporal abstraction (long-term goals vs short-term actions), not explicitly about norm-based veto. NES’s hierarchy is about value abstraction (moral/meta values vs reactive impulses). In governance terms, HRL’s top policy is still ultimately in charge of achieving the reward – it might suppress some sub-behaviors if they don’t lead to good reward long-term, similar to how NES suppresses an impulse violating a high-level norm. But HRL’s top policy itself is learned and shaped by reward, whereas NES’s top layer (norms) is somewhat fixed or explicitly given (though it can learn norms, it treats them as constraints/principles more than just accumulated reward). The distributed nature of HRL means no homunculus – the “executive” is implemented as a set of policies each solving a subproblem. NES also distributes executive function (comparator for evidence, conflict resolver for norm arbitration, etc.), but it does have a concept of a highest layer (RAA) that stops recursion and commits to a decision. HRL could theoretically have a “top policy” that decides when to stop deliberating too (if meta-decisions are part of the hierarchy). In sum, HRL and NES both exemplify hierarchical, multi-layer governance as opposed to a single flat executive. NES is more prescriptive about what each layer does (since it’s hand-designed), whereas HRL’s layers are typically trained or abstracted from the task.

Executive Function Theories (SAS/ACC/etc.): Early executive function theories (like SAS) leaned toward a unitary supervisory system concept – the SAS is one entity (some sort of central attention controller) overseeing the contention scheduler. Baddeley’s model similarly had a “Central Executive” as a single box in his model of working memory. These are clear homuncular placeholders, though Norman & Shallice argued it’s not a ghost in the machine if you specify conditions under which it operates (they did not fully specify how it chooses though). More modern theories fragment this: e.g. Miyake et al. identified multiple executive subfunctions (inhibition, shifting, updating), suggesting there isn’t literally one executive module but a set of control processes. The conflict-monitoring model (ACC-PFC loop) is a distributed two-part system: ACC detects conflict, PFC adjusts control – no singular “executive,” just an interaction of two specialized systems. EVC theory goes further to say the brain as a whole computes a cost/benefit and deploys control accordingly, distributing the “decision” between evaluative (ACC/BG) and implementation (PFC) components. Thus, in contemporary cognitive neuroscience, executive control is often depicted as an emergent property of a network (frontal cortex, cingulate, basal ganglia, etc.), not a single homunculus node. This network can be thought of as distributed governance with different regions handling monitoring vs. applying control vs. storing goals. NES vs Executive Theories: They are very much aligned in spirit. The NES maps its modules to such brain functions – e.g. conflict signals analogous to ACC monitoring, and the ability to loop or not loop analogous to computing expected value of more control . In effect, NES takes what executive function theories describe and packages them into a coherent architecture. It answers Norman & Shallice’s call to specify operations to avoid a homunculus by providing a model for inhibition (the Assent Gate’s threshold and veto) and conflict resolution (explicit norm weighting rules). So in terms of governance: NES, like these theories, breaks “executive” into parts – monitor, evaluator, implementer – i.e. a distributed team rather than a single decider. One could say NES is a working instantiation of a Supervisory System that is not unitary but a system-of-systems. Both avoid an unexplained central agent by giving mechanistic accounts. It’s worth noting that executive function theories often still assume a unity of purpose – e.g. a single top-level goal or task set that biases everything – which is somewhat unitary. NES assumes a single integrated goal system too (the Norm Repository provides a common set of standards for all decisions, implying the agent has a unified value structure to reference ). Thus, while structurally distributed, the governance in NES and in these theories is functionally unified by top-level goals/norms.

Summary for this dimension: NES positions itself in between extremes: it rejects a unitary homunculus by dividing control into modular processes, yet it coordinates them to act as a coherent governor (the “system” in Normative Executive System). Global Workspace and traditional dual-process also envision a unified moment of control (conscious decision or System 2) but do not detail its internals, whereas NES provides that detail. Predictive coding and HOT represent the other end – highly distributed processes with no single decision-maker (HOT’s “self” is an inference, not a decider). ACT-R and Soar exemplify a clearly defined single executive process (a production matcher) – NES differs by having multiple checks (accumulation, gate, conflict resolution) rather than one rule selection step. Hierarchical RL and modern neuroscience theories show how control can be multi-layered; NES mirrors that idea by introducing hierarchical checks (norms above impulses). In sum, NES’s approach to central governance is hybrid: no single all-powerful module, but a structured coalition that yields a single decision. This sets it apart from frameworks that assume an undefined central executive, while aligning it with computational architectures that explicitly implement decision control.

### **2. Algorithmic Specificity and Computational Implementability**

NES: The Governing Faculty Blueprint strives for a high degree of algorithmic specificity. It doesn’t remain a metaphor – it lays out a pseudo-code of how decisions are made. Key elements include a Comparator modeled as a drift-diffusion process accumulating evidence for and against an impulse, an Assent Gate that applies a threshold (potentially dynamic) to decide if the accumulated evidence warrants action, and a Norm Conflict Resolver that uses a weighted voting algorithm with veto to adjudicate between conflicting norms . The recursion (RAA) is defined with stopping criteria (e.g. limit of cycles or default to inaction after N loops). The blueprint even invokes analogies to neural mechanisms: e.g. the threshold can be modulated by a serotonin-like parameter to mimic inhibition control . These details indicate NES is specified at a mechanistic level granular enough to be implemented in code or a computational model. Indeed, the document provides toy simulations and examples (like a scenario with Norm1 “don’t murder” weight 9, Norm2 “save lives” weight 8, showing how a veto works) to illustrate the algorithm in action . Therefore, NES has a clear architecture diagram and step-by-step decision procedure. One could program an NES agent by following the blueprint’s design – making it highly implementable. The existence of modules and signals suggests it could be translated into a block diagram or even a neural network schema (with accumulators as nodes, thresholds as tunable parameters, etc.). In short, NES’s creators have given it the rigor of a cognitive architecture rather than leaving it as a conceptual model.

It is important to note that while NES is quite detailed internally, it is not yet an industry-standard implemented system. It’s a proposal (albeit with simulation prototypes). So in terms of maturity of implementation, it’s at the blueprint stage, not a widely tested platform like ACT-R or an off-the-shelf algorithm like Q-learning. But the blueprint clearly has implementability in mind – even mapping components to possible brain substrates suggests they envision practical instantiation.

Global Workspace Theory: GWT initially was formulated in a descriptive manner (the “theater of consciousness” metaphor). Bernard Baars described it in flow diagrams and functional terms, but not as an exact algorithm. However, GWT has since been instantiated in various computational models. For example, Stan Franklin’s IDA and LIDA cognitive architectures implement a global workspace idea with code: numerous codelets compete, and a “broadcast” occurs to update all modules when a winner is chosen. Dehaene’s Global Neuronal Workspace (GNW) has been simulated in neural networks (even spiking neuron models) showing ignition of activity when a threshold of excitation is reached. These implementations indicate GWT is computationally implementable, though it’s a framework that can have multiple implementations rather than a singular algorithm. It lacks the specificity of saying exactly which information gets in – that is often left to an attentional mechanism or “relevance” heuristic that is not fully specified in the theory itself. So relative to NES: GWT is less algorithmically specific about the executive process. It specifies the architecture (a set of distributed specialists and a global blackboard they can write to) and a general principle (broadcast when threshold is reached), but not the nitty-gritty of how conflicts are decided beyond “the most activated coalition wins.” GWT doesn’t include explicit code for e.g. norm evaluation or exactly how attention biases inputs (though neuroscientific GNW models incorporate top-down weights).

To implement GWT, one often has to add ad-hoc methods for particular tasks. NES, on the other hand, offers a built-in method for conflict resolution and decision gating. So in terms of clarity: NES is like a specific algorithm that uses global broadcasting at the end (NES explicitly notes its decision result is globally broadcast, aligning with GWT ), whereas GWT is the idea of broadcasting without a specific content selection algorithm beyond “activity”. Nonetheless, because GNW is formalized with differential equations for neural activity, one could say GWT (GNW) has a formal model at the neural level – which is precise but at a different level of description (neurons and synapses vs. cognitive rules). Both are implementable: one could implement NES on top of a GW architecture (for instance, use NES to determine which content gets broadcast).

Predictive Coding: Predictive coding is perhaps one of the most mathematically rigorous frameworks in cognitive neuroscience. It is often expressed in equations (Bayesian update equations or variational free energy formulas). This makes it highly specific algorithmically – one can write simulations of predictive coding networks that exactly follow the math. Many such simulations exist (for vision, for sensorimotor integration, etc.). The Free Energy Principle extends this to a very general mathematical conjecture: essentially providing an objective function (minimize free energy) that one could use gradient descent to optimize. In practice, implementing full brain-scale predictive coding is complex, but not due to vagueness of the theory – rather due to sheer scale. Conceptually, the steps are clear: initialize a generative model, compute prediction errors, update beliefs or predictions to reduce error, repeat continuously. Active inference adds: also choose actions that are expected to reduce error (through changing sensory input to match predictions).

So, predictive coding is computationally implementable and in fact has influenced machine learning algorithms (like variational autoencoders, which are akin to predictive coding models training a generative network). Some robotic controllers have been built on active inference principles (though not as popular as standard RL).

Comparatively, NES is specific at a more discrete decision-making level, whereas predictive coding is specific at a differential/dynamic systems level. NES could theoretically be implemented within a predictive coding scheme by having a generative model that encodes norms and predictions of outcomes if norms are broken, etc., then letting prediction errors drive a decision (the difficulty is that predictive coding doesn’t naturally do stepwise decisions – it’s continuous, but you can define a decision as made when a certain belief passes a threshold, akin to an assent gate spontaneously emerging from precision weighting). However, that is speculative. In terms of clarity, NES reads like a pseudo-code algorithm, predictive coding reads like a set of equations – both quite clear in their domain. One difference: predictive coding doesn’t explicitly code an algorithm for executive control as a separate function; it provides a general learning and inference algorithm. So if we ask “how to implement executive control in predictive coding,” one has to craft a suitable model architecture (for example, include nodes that represent goal states or norms as prior expectations). This is possible but not handed by the theory without interpretation. Meanwhile, NES gives a purpose-built algorithm for executive control.

Dual-Process Models: These models, in their basic form, are not algorithmically specified. They are conceptual distinctions supported by experimental results, but they don’t come with a unified equation or code that you can run. For example, Kahneman’s System 1 vs System 2 is described in prose and demonstrated by example (fast vs slow tasks), but there is no cognitive architecture accompanying it. Some computational cognitive models have been created to instantiate dual-process ideas in specific domains (e.g. models of reasoning that switch between fast heuristic and slower logic, or neural network models with two pathways), but there is no single “dual-process algorithm”. Thus, dual-process accounts are low in inherent algorithmic specificity. They rely on intuitive notions like “System 2 intervenes if it detects an error in System 1’s output” – but exactly how it detects error or what triggers analytic thinking is not rigorously defined generally (some specific models define it, e.g. a conflict signal or a heuristic output below confidence threshold triggers deliberation, but that’s model-specific).

NES, on the other hand, can be seen as a concrete algorithm that produces behavior consistent with dual-process phenomena (as cited, the NES architecture naturally reproduces slower reaction times when an emotional veto norm conflicts with a utilitarian goal ). So NES provides a potential implementation of a dual-process control: an impulse comes (fast process), if a norm veto is triggered, an iterative deliberation with evidence accumulation happens (slow process) . In that sense, NES is algorithmically far more concrete, whereas vanilla dual-process theory by itself isn’t something you can “run” on a computer without making additional assumptions.

Higher-Order Thought Theories: HOT theories are among the least algorithmically specified frameworks in our list because they are primarily philosophical. They state a criterion for consciousness (presence of a higher-order representation) but not a procedure. There is no algorithm for how higher-order thoughts are generated or how the mind decides to form them. Cognitive scientists have tried to model metacognition computationally (for example, computing confidence as a kind of higher-order estimate of uncertainty), but HOT theory itself doesn’t give that. It doesn’t tell you when a higher-order thought will form or not, or how it targets a specific lower state. It more or less assumes the brain has a capacity for reflexive representation and leaves it at that. So implementability is limited: one could implement a HOT theory by designing an AI with a second-level representation module that can access the first-level states and label them, but you’d be creating your own algorithm guided by the idea rather than following a specified procedure from the theory.

The Attention Schema Theory (AST) (a related idea) is a bit more concrete: it suggests the brain maintains an internal model (a schema) of its attentional focus, which is a simplified depiction that gives rise to the notion of a self. One could attempt to model that by, say, a neural network that outputs a summary of its current attention state and uses that for further control. But still, AST is not fully formalized as an algorithm, though it’s more mechanistic than HOT in describing a specific kind of information (attention parameters) being modeled.

NES compared to HOT/AST: NES has a full algorithm for decision-making; HOT/AST have at best an algorithmic sketch for generating awareness. When it comes to executive control, HOT offers almost nothing in terms of implementable detail. It is conceivable to add a HOT layer to an existing cognitive model (some have tried: e.g. a higher-order network that monitors a lower-order network’s activation and if a certain pattern occurs, declares it “conscious”), but those implementations are experimental and not part of the core theory. For instance, one might set up a system where a “thought” represented by activation in one network triggers a meta-representation in another network that encodes “I am thinking X”; but the theory doesn’t tell you how to encode “I” or how to ensure it only happens when appropriate.

In summary, HOT theories are low implementability (barring making further assumptions), whereas NES is ready-to-implement. NES even ties its components to testable signals (like conflict signals akin to ACC, neuromodulators for thresholds, etc.), which implies how one might detect or implement those in either a brain or machine.

ACT-R / Soar: These are fully realized cognitive architectures with existing software implementations (ACT-R has a publicly available code base in Lisp/Python, Soar has a C++ implementation). They are extremely specific: ACT-R defines the data structures (chunks, buffers), timing parameters, learning equations for activation and utility, etc. If anything, they might be too specific or rigid for some real-world complexity, but they are definitely implementable (thousands of models have been built in them). They come with a user manual instead of leaving things to intuition. So on algorithmic specificity, ACT-R and Soar are very high. They provide a virtual machine for cognition.

Compared to NES, ACT-R/Soar have the advantage of existing, optimized implementations and decades of refinement. NES is a newer conceptual architecture that presumably could be implemented but hasn’t been widely yet. However, NES is written in a similarly specific style. One might say NES is proposing a specialized architecture for normative executive control, whereas ACT-R is a general architecture for all cognition with extension to tasks. The difference in specificity is domain: ACT-R can model multi-step cognitive tasks (like solving algebra) by writing rules, whereas the NES blueprint is focused on the moment of impulse decision (it doesn’t, for example, describe how the agent would perform multi-step reasoning or store facts in memory – those are outside its scope). So ACT-R is broader (and has modules for memory, perception, etc.), while NES is deeper in the specific domain of impulse control and norm evaluation.

One could implement NES within ACT-R: e.g. use ACT-R’s production rules to realize the logic of NES modules. Conversely, one could extend NES to have more ACT-R-like components for memory and perception. The key point is ACT-R and Soar prove that explicit cognitive architectures are implementable and can mirror human behavior data, and NES emulates that style of rigorous specification specifically for the executive faculty. This means NES could likely be translated into an ACT-R-like production system or a specialized simulation with relative ease (at least ease conceptually – engineering it is non-trivial but straightforward given the design).

Hierarchical RL: HRL is formalized in the language of Markov decision processes and dynamic programming. For instance, the options framework specifies an “option” as a policy with a initiation set and termination condition, and one can derive equations for learning option values. Algorithms like Option-Critic exist that can learn hierarchical policies. So HRL is algorithmically explicit – it’s a subset of RL with clear mathematics. Many HRL algorithms have been implemented (in fact any RL library might include some hierarchical methods, or one can code them easily if not).

HRL’s implementability is high – it’s a staple of AI research. The only complexity is that one often has to hand-design the hierarchy or give hints, though meta-learning approaches try to learn hierarchies automatically. But the framework itself isn’t vague: it says if you have subtask policies and master policies, you update them in such-and-such way.

For example, Feudal RL (Dayan & Hinton) had a procedure where higher-level Q-values set subgoals for lower modules. Later HRL work, like MAXQ (Dietterich), gave a formal method to decompose value functions. These are all clearly implementable and have been tested in domains like navigation, games, etc.

So HRL is as concrete as mainstream RL, which is very concrete (backed by math and code). In comparison, NES is also concrete but in a different paradigm. We could argue NES is not fully formalized in mathematical equations for learning like RL, because NES largely assumes the Norm repository and weights are given or learned in specialized ways (not via end-to-end reward optimization). But NES does propose methods for updating norms via RPE signals and Bayesian inference , which is similar to RL and Bayesian learning formalisms. So in a sense, NES touches on multiple learning formalisms (reinforcement learning for norm inference, plus possibly supervised input from authority, etc.). It’s not encapsulated in one neat objective function, because it’s dealing with the interplay of hard constraints (vetoes) and soft evidence. That’s a bit trickier to cast as a single optimization problem. But it’s still an algorithmic recipe one can implement: detect big punishment errors, create a norm, etc., as described in the blueprint .

Executive Function Theories: The algorithmic specificity here varies. Some (like the Conflict Monitoring model or EVC) are fairly formal: e.g. conflict monitoring can be modeled in a neural network (a well-known connectionist model of the Stroop effect by Cohen et al. 1990 implemented conflict detection and biasing, and later models computed a quantitative expected value of control). There are equations for how much control to deploy based on expected reward rate and mental effort cost . So parts of these theories are computational. But taken as a whole, “executive function” is broad and not one algorithm. It’s an umbrella for functions. SAS is semi-formal: it outlines when the supervisory system is needed (novel situations, etc.) but doesn’t give a detailed algorithm beyond “apply attention to bias selection.” However, researchers like Norman developed computational simulations for action selection with and without SAS intervention. So, many specific models exist for pieces of executive function (task switching models, inhibitory control race models, working memory gating models using neural networks or cognitive equations).

What’s lacking is a unified architecture combining all these in one system – which is exactly what something like NES or ACT-R tries to be. So while aspects are formalized, executive function theories as typically discussed in psychology are moderately specified at best. They are often expressed as box-and-arrow diagrams or qualitative principles (like “monitor conflict; if conflict high, increase control signal”), which can be turned into an algorithm but require choices (how to measure conflict, how much to increase control, etc.). EVC made that quite explicit by positing a cost function for control and saying the system chooses control level that maximizes (expected outcome – effort cost). That’s a specific algorithm (solve an optimization each time) which one could implement.

NES has actually drawn from these to make its own algorithm. For example, RAA’s decision to loop again or not is akin to computing an expected value: “our RAA chooses when to loop or not, effectively a cost-benefit of more deliberation” . This is a link to EVC – implying the model assesses if doing another cycle is worth the potential benefit (maybe implicitly if conflict remains high and time allows, it does it, but if urgency is high and not much benefit expected, it stops). The blueprint doesn’t give a numeric equation for this, but it references the concept. We might imagine implementing NES with a parameter for “urgency” or time cost that influences the recursion stopping rule.

In summary, among executive function models, some components are implementable but there isn’t a single standard code (unlike ACT-R which is a delivered code framework). NES and cognitive architectures aim to fill that gap by providing an integrated implementable model. So relative to those theories, NES is a step toward formalizing what they describe.

Conclusion on this dimension: NES and frameworks like ACT-R, HRL, predictive coding are high on algorithmic specificity – one can take these and attempt to simulate or program them. NES is novel in that it formalizes normative decision-making in a cognitive architecture style, something few others do. GWT and executive function theories, while influential, often serve as conceptual scaffolds that need additional formal assumptions to implement. Dual-process and HOT are mostly conceptual, requiring translation into a model (which NES kind of does for dual-process in the moral realm).

The computational implementability of NES seems quite feasible – it’s spelled out with enough detail that a software agent or a detailed flowchart could be made. The blueprint itself appears to have been written with the intent to eventually test it in AI or in a cognitive simulation, making it comparable to established architectures. NES stands out by tying its algorithm to normative reasoning, which means implementing it would allow experimental predictions about moral decision-making and self-control tasks. This level of detail and focus on norms is a key innovation relative to other frameworks that either ignore norms or mention them abstractly (like “internalized standards” in self-regulation theory, which is not a fully fleshed algorithm whereas NES gives one).

### **3. Treatment of Norms and Values**

NES: Norms and values are the centerpiece of the Normative Executive System. NES explicitly contains a Norm Repository – a structured store of internalized standards, rules, goals, and their associated importance weights . Every incoming impulse or action idea is evaluated against these norms via the Comparator and Norm Conflict Resolver. In NES, norms can forbid, oblige, or permit certain actions, and they come with weights indicating their priority. Crucially, NES implements a deontic veto mechanism: if a norm marked as absolute prohibition applies, it can outright veto an impulse regardless of other considerations . More generally, NES treats decision-making as normatively guided: it’s not just about maximizing reward or achieving goals, but about checking alignment with internal values at each step. The blueprint outlines scenarios like moral dilemmas where one norm (e.g. “do not kill”) conflicts with another (“save lives”), and how the system arbitrates that conflict by comparing weights or applying veto if one norm is absolute .

NES also discusses how norms are acquired and adjusted, indicating norms are not static: through social learning, reinforcement signals, and Bayesian inference, the Norm Repository can be updated (e.g. learning new rules or adjusting a norm’s weight based on experience) . This means the system’s values themselves are subject to a developmental process, akin to how humans internalize cultural and ethical rules over time.

In summary, NES explicitly represents values as first-class variables in the decision process. Norm compliance is not an afterthought; it’s baked into the core architecture. The system effectively tries to align actions with these norms, giving NES a built-in moral/self-regulatory dimension that most other models lack innately.

Global Workspace Theory: GWT itself is content-agnostic. It doesn’t provide a built-in notion of norms or values – any information can enter the workspace, including goals or norms, but the theory doesn’t assign a special mechanism to them. It’s up to whatever unconscious processes exist to enforce or consider norms. For example, one could imagine in a GWT-based mind, some unconscious rule-checking process tries to push “Warning: this action violates rule X” into the workspace when an impulse arises. If that warning wins the competition, the person becomes conscious of the conflict and might stop the action. However, GWT doesn’t specify this; it simply would accommodate that scenario. In many GWT implementations, there is a notion of “Goal context” or “context frames” which bias what gets through – this can be related to values insofar as your current goal can reflect a value (like “I value health so my goal is to avoid junk food”). But again, GWT proper doesn’t differentiate norm information from any other information.

Thus, compared to NES, GWT has no dedicated norm module or evaluation process. It would rely on other cognitive mechanisms (which could be integrated in a model using GWT) to handle norms. For instance, a GWT-based cognitive architecture might incorporate a production system or constraint solver to enforce rules behind the scenes. But that would be an addition. The global workspace can broadcast a norm (like recalling a principle “don’t lie” into consciousness), but it doesn’t automatically check every action against all norms.

In effect, values in GWT are just another kind of mental content that can influence the competition for consciousness. Emotional or value-laden content might have higher salience and thus more likely to enter the workspace, but that’s a functional detail one must design. NES, by contrast, has a systematic check: impulses are filtered through normative criteria by default. NES could thus be seen as adding a normative filtering layer to something like GWT’s architecture.

Predictive Coding: In standard predictive coding or active inference formulations, there isn’t an explicit representation of “thou shalt not do X.” However, values enter in a different way: through the notion of prior preferences or reward/punishment signals. For example, in active inference, one can encode “preferred states” that the agent tries to achieve (like avoiding hunger, maintaining integrity, etc.). These act like internal value functions. In reinforcement learning terms, predictive coding can incorporate a utility term in the free energy to reflect that some outcomes are inherently costly (normatively bad) or beneficial.

So if we consider how to embed norms in predictive coding, one way is to treat violation of a norm as a highly surprising or low-preference outcome, which the system will act to avoid. For instance, if the agent has a strong prior that “murdering is not supposed to happen,” then a scenario where it would commit murder would have high predicted free energy (because it violates the prior), and thus the system would steer away from that behavior to minimize free energy. This is conceptually workable, but it requires setting those priors explicitly. Predictive coding frameworks per se don’t tell you which priors to set – they provide the machinery to honor them once set. So the onus is on the modeler to bake in normative values.

Another angle: predictive processing often equates “value” with “expected reward” in a classic RL sense, and it does not inherently differentiate moral rules from other forms of value. All preferences are just part of the model. There is no notion of an inviolable veto; everything is trade-offs in terms of prediction errors or expected utility. You could emulate a veto by making the cost of a certain action effectively infinite in the model, but again, that’s an external imposition.

Thus, predictive coding does not natively have a norms module. It handles motivations in terms of reward expectations or prior preferences. It tends toward a utilitarian integration of values (weighing outcomes by probability and utility). NES on the other hand distinguishes norms qualitatively (especially with the veto concept for deontological rules). This is an interesting difference: NES allows some values to be non-negotiable (veto rules), whereas predictive coding’s math usually turns everything into a scalar error to minimize – which is more like weighing costs vs benefits continuously.

If we translate NES’s approach to predictive coding language, an absolute norm would be a constraint that the model will treat any policy leading to norm violation as having effectively infinite cost, so it would have prior probability zero. It’s possible to implement (hard constraints in planning), but not typical in purely Bayesian brain theories, which usually assume soft trade-offs.

Dual-Process Models: In their generic form, dual-process models are not about content (norms) but about processing style. However, when applied to domains like moral reasoning (Greene’s work), they implicitly discuss values: System 1 might carry intuitive moral rules (like deontological “do not harm”) and emotional aversions, while System 2 carries more calculative values (like utilitarian cost-benefit analysis or adherence to a considered principle). Greene et al. famously argued that deontological judgments stem from automatic emotional responses (like personal violence causing strong aversion), whereas utilitarian judgments come from effortful reasoning about maximizing welfare . In that interpretation, values are split across the systems: some values are deeply ingrained (e.g. empathy-based aversion to harm) and trigger automatically, others are explicit (e.g. valuing overall outcomes) and require deliberation.

However, dual-process theory does not itself provide a mechanism for representing a list of norms. It more or less assumes the person has some values, some of which trigger emotional responses, others which require conscious thought. It doesn’t formalize how those values are stored or checked. It also doesn’t necessarily ensure consistency – that’s more up to the individual’s reflective equilibrium.

NES vs Dual-Process on norms: NES offers a concrete method for integrating both intuitive and deliberative values: each norm has a weight and type (veto or not). An “intuitive/emotional” norm like “don’t do direct harm” could be encoded as a high-weight veto norm, causing immediate inhibition of certain impulses (System 1 style). A more deliberative value like “maximize overall good” might be encoded as a norm that contributes weight but not veto, requiring accumulation and possibly being overridden by a veto norm (System 2 calculation that might or might not prevail depending on context). This mapping aligns nicely with how dual-process sees deontology vs utilitarian thinking . In fact, NES’s norm arbitration algorithm essentially reenacts that: a strong deontic veto norm can stop an action even if a utilitarian norm with slightly less weight pushes for it, reflecting how emotional instinct can trump calculation; but if the utilitarian consideration is overwhelmingly stronger and no absolute veto is present, it can win out, reflecting rational override.

Dual-process theory itself doesn’t provide these specifics; it is more of a qualitative story. It “treats” values by dividing them into fast and slow lanes but not by representing them explicitly. So in dual-process frameworks, one often has to talk externally about the person’s goals or moral beliefs to predict which system wins (e.g. a person who strongly endorses a rational principle might engage System 2 more). NES directly encodes those endorsements in the Norm Repository.

In summary, dual-process models acknowledge values influence decisions (through either emotional instincts or conscious reasoning), but they lack an internal representation of a value set. NES offers exactly that, giving it an advantage in any scenario where content of values matters. It can model differences between agents by different Norm Repositories, something dual-process doesn’t formalize beyond saying “some people are more utilitarian” etc.

Higher-Order Thought Theories: HOT theories generally ignore norms and values. They are concerned with consciousness and awareness, not what decisions are good or bad. A higher-order thought can be about any first-order state – it could be about a desire or a perception – but HOT theory doesn’t incorporate the concept of evaluating that desire against a standard. It’s about knowing that you have the desire, not whether you approve of it. In fact, one critique could be that HOT is a rather amoral theory of mind – it doesn’t say anything about self-control or ethics, just about what it means to experience something consciously.

If anything, one might argue that being aware of a desire (via a HOT) could allow one to apply norms to it (since you can’t apply a norm to an urge you aren’t conscious of). But HOT theory itself stops at making the desire conscious; the actual checking against norms would require another system (like frontal lobe executive functioning). In a philosophical context, some have used higher-order concepts in accounts of free will or moral responsibility (e.g. Frankfurt’s idea of higher-order volitions – wanting to want something – which touches on aligning your first-order desires with higher-order values). That’s related but distinct from HOT. Frankfurt’s notion is closer to saying we have values at a higher-order level (like “I want to be a person who doesn’t smoke, even though I crave a cigarette”). If we considered that as a “higher-order desire,” it is a bit like a norm (an internal standard for oneself). But HOT theories as usually formulated by Rosenthal or others weren’t about that specifically; they were about consciousness.

In comparison to NES: HOT doesn’t provide a mechanism to incorporate norms in decision-making. NES does. If one extended a HOT framework with normative content, one would basically be building something like NES anyway (like a second-order desire to adhere to certain principles controlling first-order desires – that’s conceptually similar to NES’s normative layer controlling impulse layer). But HOT by itself doesn’t say how that control happens; it only ensures one knows what one is doing (which is necessary but not sufficient for normative control).

ACT-R / Soar: These architectures have no inherent moral or value system, but they do have goals and utilities. In ACT-R, one could treat a goal chunk as representing the current objective or context (e.g. “I want to keep my promise”). Additionally, ACT-R’s production rules can carry utility values that indicate how desirable that rule’s action is (these utilities can be learned from reinforcement). If modeling a normative behavior, a modeler can encode norms as either hard constraints (rules that fire to inhibit certain actions when conditions are met) or as additional costs in the utilities (making norm-violating actions carry huge negative utility, effectively preventing them). So the architecture is capable of representing norms, but it doesn’t come with a pre-populated normative knowledge base – you have to program it in.

There have been ACT-R models of moral reasoning or ethical decision-making, but they are special-case. The architecture per se is neutral on values: it’s a general problem-solving engine. The difference is that ACT-R could equally well model an agent with very immoral goals if given such rules. Nothing in it ensures alignment with human norms, unless you explicitly add those as knowledge (there’s no built-in “conscience module”).

Soar similarly doesn’t include specific value content – it will pursue whatever goal it is given. It has a mechanism for preferences in decisions (Soar can represent that one operator is preferred over another, akin to a rule weight). So you could encode a norm like “if operation causes harm, prefer the other one.” But again, it must be encoded by the model designer or learned via feedback.

NES vs ACT-R on norms: NES offers a conceptual advance in that it treats a repository of norms as a distinct data structure that is always consulted. In ACT-R, one would have to design the model such that it checks a “moral memory” or includes conditions in every production to check if a rule violates a norm. That’s doable, but burdensome and not a native feature. NES basically says: always check Norm Repository before finalizing an action – a principle built-in. ACT-R/Soar have no always-on moral check unless you program it.

One could combine them: implement a Norm Repository as part of ACT-R’s declarative memory, and have a set of general production rules in ACT-R that whenever an action is about to be executed, query memory for any norms against it. That would emulate NES. But that’s adding to the base architecture.

Hierarchical RL: Standard RL encodes “values” as numeric reward signals. Hierarchical RL similarly propagates a reward (or cost) up and down the hierarchy. There is no notion of a deontological rule or a distinction between types of value – all is one scalar reward to maximize. For example, if we want an RL agent to follow a rule like “never cross a red light,” we typically implement that by assigning a very large negative reward if it crosses a red light. The agent, if it learns optimally, will avoid that because it’s not worth the pain. This works but is essentially treating norms as hard-coded penalties in the reward function. There’s no separate module that “vetoes” the action of crossing the red light; it’s just that the optimal policy under the reward will not do it.

This means RL is inherently consequentialist (outcomes weighted by reward). Norms in human sense often have a deontic character (like you simply do not do certain things). But RL can mimic that if the penalty is effectively infinite or if the training examples never show any violation because it was constrained.

Some newer approaches in AI consider safety constraints or shielding, which is akin to having a module that intercepts actions that would violate a formal safety specification. That is more similar to a normative check. But baseline RL doesn’t have it – it’s a bolt-on from formal methods.

NES vs RL on values: NES distinguishes between normative value and utility. A norm might tell the system to forego an action even if that action would have yielded a higher utility (like save more lives at cost of breaking a rule, NES could refuse because of a veto). RL would just pick whichever yields higher expected cumulative reward (unless shaped as above). This is a fundamental difference: RL tends to integrate everything into one value function, whereas NES allows lexicographic or rule-based prioritization. In practice, a hierarchical RL might approximate rule-based behavior if one part of the hierarchy encodes constraints – for instance, a higher-level policy could override a lower-level one if it predicts huge negative reward (like a self-preservation instinct kicking in to stop a dangerous behavior). But that still reduces to weighing reward.

So out-of-the-box, hierarchical RL does not treat “norms” distinctly. One could program an RL agent with multiple reward signals (multi-objective RL) to reflect different values, and then have a mechanism to resolve them (like weight them or treat some as constraints). That essentially starts turning RL into something like NES’s weighted + veto scheme. In fact, NES’s conflict resolver could be seen as a multi-objective decision rule (some objectives are lexicographic vetoes, others summed by weight). This is not typical in RL, where a single scalar reward is simpler.

However, RL could incorporate that: e.g. constrained MDPs or using Lagrange multipliers for constraints. But again, those are explicit designs. The normative aspect in RL is usually extrinsic (given by the reward design from a human); RL itself doesn’t have an internal notion of “right or wrong,” just “what gets me more reward.” If we anthropomorphize, an RL agent’s “values” are only as good as its reward function.

Executive Function Theories: In psychology, notions like “values,” “goals,” or “normative standards” do appear in self-regulation theory. For example, Carver and Scheier’s control model describes behavior as a negative feedback loop where you compare your current state to a reference value (which could be a goal or ideal). This reference is essentially a value or standard. If there’s a discrepancy, you try to reduce it. That is a very similar concept to NES’s comparator (which compares impulse outcomes to internalized standards). The difference is Carver and Scheier’s is a general idea not tied to a particular implementational mechanism, whereas NES spells out how the comparison and correction happens in the context of action impulses.

The concept of conscience or internalized social norms has been studied in developmental psychology (e.g. how children internalize parental rules). Those accounts, however, are often verbal or at best computational in broad strokes (like reinforcement learning of rule-following behavior or psychoanalytic models which aren’t computational at all). There isn’t a widely accepted computational model of “conscience” in neuroscience, though theories implicate certain brain areas (ventromedial PFC storing social rules, or the guilt feeling generated by limbic interactions).

Norman & Shallice’s model did not explicitly include “norms,” but the SAS by nature enforces the person’s intentions or will over habits. If the person’s intention is guided by a value (e.g. I intend to be honest), SAS will help override an automatic impulse to lie. So indirectly, the executive system enforces values by implementing the top-down goals which presumably reflect those values. But again, the theory doesn’t enumerate those values, it assumes the person has them set as goals when relevant.

EVC and conflict-monitoring models focus more on cognitive efficiency than content of norms. They pick control allocation to optimize performance on tasks given certain rewards for success or errors. If one equates those rewards to values, then yes, they incorporate values in the cost function. But typically that’s about accuracy vs effort, not moral values.

Overall, psychology/neuroscience executive models usually require that the values be embedded as goals or task parameters by the experiment or the person, and then the exec system just tries to optimize behavior to meet those goals. They don’t intrinsically question the goals. NES goes a step beyond by having an explicit knowledge base of norms that can sometimes override even a goal if it’s forbidden. For instance, an impulse might align with a short-term goal (steal money to get rich quick), but violate a moral norm (“don’t steal”), and NES would forbid it due to the norm. A standard exec function model might just see a conflict between a temptation and a goal of being moral if that goal is active; if the person hadn’t actively set a goal “be moral,” the model might not intervene. NES’s Norm Repository is always active in evaluation, effectively assuming the agent carries its values into every situation without needing to consciously activate them each time.

Comparison summary for norms: NES uniquely prioritizes norms/values as core inputs to decision-making. Most other frameworks either:

* Do not explicitly represent norms at all (GWT, HOT, basic dual-process, ACT-R base, RL base).
* Or they treat values in a simplified way (as scalar rewards or goals).
* Some psychological theories acknowledge internal standards but don’t formalize how they are applied in real-time decision.

NES introduces a voting and veto paradigm for norms , which none of the others have in their basic form. This is a strong divergence. It means NES can straightforwardly model ethical reasoning and self-control dilemmas. For example, NES can simulate the classic “angel vs devil on shoulder” scenario: a norm (angel) vetoing a tempting impulse (devil) with a mechanistic outcome. Other models require us to conceptualize that in their terms (maybe as conflict between two goal representations in a global workspace, or as a negative reward for the bad action).

Finally, from an AI alignment perspective, NES providing an internal norm module is interesting – it’s akin to programming an AI with explicit ethical rules and a way to balance them, which goes beyond plain RL or planning. It’s somewhat reminiscent of Asimov’s laws of robotics conceptually (where certain rules are hard constraints). The blueprint’s Canonical Alignment Guide presumably deals with how to align those norms with desirable principles. Traditional frameworks don’t directly address alignment or normative guidance (they assume the user sets the goal or reward). NES proactively includes it.

### **4. Assent and Withholding Mechanism (Executive Gating of Action)**

NES: The concept of assent – giving or withholding permission for an impulse to turn into action – is explicitly implemented by NES’s Assent Gate. This module serves as a dynamic gatekeeper: as the Comparator accumulates evidence for an impulse, the Assent Gate decides whether the threshold for action is reached or not . If the threshold is not reached (or a veto is signaled), the impulse is withheld (no “go” signal), effectively inhibiting the action. The Stoic notion that the mind can withhold assent to a tempting impression is directly mirrored here. The Assent Gate is not a simple static filter; it is adjustable – for instance, its threshold can be raised or lowered by various factors like emotional state or specific instructions from the RAA . This means the degree of impulse control is context-sensitive: in high-risk situations the system might require more evidence (thus withholding more easily), whereas in low-stakes or urgent situations it might lower the bar to act quickly.

Importantly, the Assent mechanism in NES is binary in outcome – either an action impulse is allowed to pass or it is blocked (at least temporarily). NES design thereby distinguishes between the generation of impulses (which might be automatic or distributed) and the final commitment to action, which is controlled. If a norm vetoes an impulse, the Assent Gate can enforce a “No-Go” regardless of how strong the impulse evidence is . Conversely, if an impulse strongly aligns with norms and evidence (and no vetoes), the gate opens (assent given).

The presence of an explicit gate with thresholds is analogous to models of response inhibition in neuroscience (e.g. the idea of a threshold in basal ganglia that needs to be reached for a motor action to initiate, with dopamine modulating it). In fact, NES draws that parallel by analogizing the threshold modulation to serotonin’s role in inhibition – higher serotonin means a higher threshold (harder to trigger action, more withholding) consistent with behavioral inhibition .

Global Workspace Theory: GWT itself doesn’t have a dedicated “gate” for action. However, it implies something similar: only the information that wins the competition and is broadcast will drive conscious deliberation and widespread recruitment for action. We can interpret that if an impulse is not broadcast, it might remain unconscious and not fully executed. But it’s not as clear-cut as an on/off gate for action; unconscious processes could still trigger habitual actions without entering the workspace. In Baars’ framework, though, voluntary (non-habitual) actions typically involve conscious broadcasting of the goal or decision.

In practice, cognitive models inspired by GWT often include a notion of a gate for working memory or action selection (e.g. many models have a “contention scheduling” for routine actions and an attention/supervisory gating for non-routine, similar to SAS). But GWT itself doesn’t delineate that. The homunculus is avoided by having the competition, but that competition must have some criteria.

So we can say GWT’s gating is implicit: attention and competition mechanisms effectively gate what enters consciousness, and thus what becomes a deliberate action. Yet, reflexive or very strong impulses might cause action without a conscious gate (like a startle response – though one could argue there is still a threshold to overcome, just one set low and in subcortical circuits).

Compared to NES, GWT doesn’t provide a lever for raising/lowering thresholds on the fly by a central system. It does acknowledge that top-down attention can bias what content wins the workspace (which is analogous to raising threshold for irrelevant stuff, lowering for relevant). But that is more about selection of perceptual content than a direct yes/no on a single action.

Predictive Coding: In predictive coding, there’s not a discrete gating mechanism but a continuous modulation of signals by precision weighting. “Precision” in predictive coding is akin to the inverse of a variance – how much trust to put in a signal. Some have equated precision modulation to attention (high precision on certain prediction errors means you pay attention to those errors and adjust predictions accordingly). If we draw an analogy, an impulse to act could be a prediction at a motor level, and whether it gets executed might depend on the precision weighting of the sensory consequences or the confidence in that prediction.

Active inference suggests that actions are selected to minimize surprisal with respect to a goal state. If an action would cause a highly surprising outcome (like violating a strong prior), then the system either won’t select that action or will have a strong error signal if it starts to do it, possibly leading to an abort. That’s somewhat analogous to a “withhold if it violates prior.” But this is all within a continuous optimization framework, not a binary gate.

In simpler terms, predictive coding doesn’t have a labeled component that says “this is the stop-signal mechanism.” However, one could build a stop-signal task model with predictive coding – some have done similar things with hierarchical models where a higher-level can cancel a lower-level plan.

If we consider brain implementation, one might map the idea of withholding to the function of the subthalamic nucleus in the basal ganglia, which is known in neuroscience to sometimes act as a “brake” on actions (the hyperdirect pathway can send a global no-go signal). Predictive coding on its own doesn’t mention that, but an integrated neuroscientific account might add it.

NES’s assent gate is like a functional analog of inhibitory control circuits (like basal ganglia stops or the prefrontal top-down inhibition). Many frameworks acknowledge such inhibition but often not in detail. NES formalizes it clearly.

Dual-Process Models: These imply a mechanism for override of impulses, but often don’t clarify how. The typical description is: System 1 generates an intuitive response; if System 2 is engaged and finds it inappropriate, it can override or modify the response. This “override” implies withholding the initial response and substituting another. For instance, in a Go/No-Go context, an automatic “Go” might be inhibited by a controlled “No” if one recognizes the context requires it.

However, dual-process doesn’t give a timing or threshold model for this. It’s more of a flow: if conflict or if motivation to be accurate is present, then engage control and stop the automatic response.

Some psychological models that deal specifically with inhibition, like the “Inhibitory control” models (e.g. the horse-race model of stop-signal tasks by Logan), are quite specific: they have a race between a go process and a stop process, if the stop process finishes first, the action is inhibited. That’s a kind of gating mechanism model. It’s not exactly dual-process in the broad sense, but a similar idea (two processes racing, one acts as a gate).

NES’s gate is effectively a threshold that an accumulation must reach, which is mathematically equivalent to a race between a GO process and a STOP process to threshold (drift-diffusion model perspective). So NES aligns well with those cognitive models (drift-diffusion and race models of decision). Many dual-process phenomena (like impulse inhibition in a Stroop or a temptation scenario) can be mapped to such models.

So, dual-process theory by itself just says “the executive can intervene,” but NES provides a clear mechanism for that intervention – the Assent Gate (potentially commanded by RAA or norm signals to raise threshold, i.e. basically adding a STOP bias).

Higher-Order Thought Theories: HOT doesn’t incorporate an action gating mechanism. A higher-order thought might inform one’s conscious knowledge (“I have an urge to eat cake and I know it”), but whether one then withholds or assents to that urge depends on other processes (like willpower, executive function). HOT theory doesn’t claim that the presence of a HO thought automatically vetoes the first-order state; it just makes one aware. Arguably, being aware of an impulse is a prerequisite to voluntarily inhibiting it, but not a guarantee. So HOT is neutral on gating – one could incorporate a mechanism that once you have a HOT about a forbidden impulse, you feel conflict and then another system stops the action. But again, that other system is something like NES.

So HOT alone: no gating, just introspection. The heavy lifting of assent/withholding is outside its scope.

ACT-R / Soar: These architectures do have mechanisms for conflict resolution among actions. In ACT-R, only one production can fire at a time, so if two possible actions are represented by two rules, only one will get selected (the other is, in effect, “withheld”). The selection is done by utility or by specificity, etc., and importantly, ACT-R can model inhibitory rules – rules that specifically stop other rules from firing. For example, you could have a production like “if about to perform action A and condition X is true (a norm or constraint), then cancel action A (or set a flag that blocks it).” This is one way to implement a veto.

Another way ACT-R handles not acting is by doing nothing if no production meets criteria or if a production that acts is withheld by a condition. That said, ACT-R doesn’t inherently simulate the continuous time competition where an impulse gradually builds and a gate stops it. It’s more discrete: either a rule fires this cycle or not. But the pattern matching for rules can be seen as a gating: if conditions (including normative conditions) aren’t satisfied, the action rule doesn’t match and thus doesn’t fire.

So yes, ACT-R can implement gating by requiring an “assent” condition in production rules for actions. There’s no built-in default that all actions must get approval though; you’d have to craft the model that way. Similarly, Soar would allow implementing a structure where an operator (action) is proposed and then a separate decision step must confirm it, or a higher context rule can reject it.

So basically, ACT-R/Soar can model assent/withholding but do not have a specialized module named that. It’s just part of the rule logic or meta-rules.

One more relevant thing: cognitive architectures often incorporate the concept of a goal stack. If an action doesn’t align with the current goal, typically the architecture won’t choose it (because the conditions won’t be met). If we treat norms as persistent goals (e.g. a goal like “never break rule X”), then any action that would break rule X could be encoded as conflicting with that goal and thus not be selected. That requires some conflict resolution strategy which often is utility-based or priority-based. So it can be done.

NES formalizes it more straightforwardly: a separate gating function rather than burying it in production conditions.

Hierarchical RL: In HRL or RL in general, an agent could learn a policy to “hesitate” or “stop” when uncertain, effectively learning a gating behavior. Also, in some HRL approaches, a high-level policy can terminate a low-level option early if things go wrong. However, RL typically doesn’t feature a distinct explicit stop controller. One notable algorithmic concept in RL is safe interruptibility – the idea of being able to interrupt an AI’s action. But that’s usually considered from an external perspective (a human can press a stop button) rather than the agent itself having an interrupt mechanism. That said, an agent could have an “abort” action in its repertoire, which it might execute if it predicts continuing current action yields bad reward.

Hierarchically, a meta-controller could decide to abort a subtask if it’s not fulfilling expected reward. That’s akin to withholding further commitment. But again, this is just part of the decision policy. There’s no special notion of “assent” separate from normal decision steps.

NES’s assent gating could be seen as a kind of internally learned action selection threshold – RL could approximate that by learning to be more cautious in some states (like requiring higher Q values to take an action rather than explore). There are RL algorithms that adjust exploration vs exploitation (like if uncertainty high, maybe wait), but that’s not quite normative gating, it’s more risk-based.

Executive Function Theories: They explicitly talk about inhibitory control. In cognitive psychology, inhibitory control is one of the core executive functions (Miyake’s taxonomy). It is the ability to suppress prepotent responses. The Stop-Signal Task (SST) is a common paradigm: one is told to perform an action but occasionally a stop signal sounds and they must inhibit the initiated action. Models of SST often use a race model between the go process and a stop process (logically similar to gating threshold).

Norman & Shallice’s SAS has the ability to “veto” an automatic schema. They envision that when a schema that is inappropriate gets activated (like an urge to grab someone’s food), the supervisory system intervenes, biasing it down or activating an alternative. But the details were not fully fleshed out. They just asserted the need for a mechanism to prevent strong but wrong responses.

Neuroscientifically, the concept of a veto has been famously discussed by Benjamin Libet (the “free won’t” idea – that conscious will might not initiate actions but can veto an action at the last moment). The idea is that after unconscious processes start preparing an action, consciousness can still intervene to stop it before execution. This notion maps onto an executive gating function that operates in a short time window. Some have speculated about a specific brain basis (like the fronto-basal-ganglia network providing a last-moment stop).

NES’s Assent Gate is essentially providing that “free won’t” functionality in a formal way. It literally embodies a conscious controller veto over impulses, consistent with the Libet-style veto (though NES is not necessarily limited to conscious veto; it could also operate semi-automatically for deeply internalized norms).

So within executive function theories, NES is most aligned with those that highlight inhibitory control. For example, Verbruggen & Logan’s work on stopping (which NES even cites referencing how older theories used a vague “inhibition homunculus”) is mechanistically addressed by NES through a drift-diffusion threshold model. NES basically says, we provide a concrete mechanism (accumulator and adjustable threshold) instead of just a conceptual “inhibition module.”

Thus, NES’s gating operation is consistent with executive function literature but more fully specified. Executive function theories would wholeheartedly agree that a critical part of executive control is deciding which impulses to allow and which to suppress; they have studied it empirically (in tasks, in brain imaging like seeing the right inferior frontal cortex active during stopping tasks, etc.). But those theories typically don’t combine that with normative reasoning explicitly. They see it as, say, suppressing a response that conflicts with task goals. NES expands it to include suppressing responses that conflict with internal norms.

Summary on gating: NES explicitly provides an assent/withhold gate, something most other frameworks either lack or implement implicitly:

* GWT: implicit gating via attention/competition.
* Predictive coding: implicit via precision and surprisal (no binary gate).
* Dual-process: conceptual gating (System 2 override) but no single mechanism defined; NES can implement that concept.
* HOT: no gating mechanism, focuses on awareness.
* ACT-R/Soar: can do gating through rules, but not a dedicated module – it’s part of the logic.
* RL: no inherent gating aside from learned policies (which can include “do nothing” actions if beneficial).
* Exec function models: acknowledge gating (inhibition/veto) as a key function; NES essentially provides a module for it that aligns with their observations.

NES’s uniqueness is partly in explicitly naming and structuring this function (Assent Gate) and linking it to normative evaluation. It doesn’t just randomly inhibit, it inhibits specifically when norms or insufficient evidence say “don’t act.” This normative gating is novel: in many cognitive models, gating is based on cognitive factors like conflict or uncertainty, whereas NES gating is explicitly tied to norm compliance as well.

### **5. Conflict Resolution and Recursive Processing**

NES: Conflict resolution in NES happens on two levels:

1. Within a single decision cycle: If multiple influences on a decision conflict (e.g. an impulse has pros and cons, or two impulses at once, or norms for and against), the system initially tries to resolve it through the Norm Conflict Resolver (if it’s norm-vs-norm conflict) or by weighing evidence in the Comparator. The Norm Conflict Resolver uses a clear algorithm: weighted voting plus any absolute vetoes to produce a recommended stance on the impulse . So at the first pass, NES might determine, for example, “On balance, Norms say NO to this impulse.” If that is decisive (e.g. a veto norm triggers), then the conflict is resolved by stopping the action.
2. Recursion across cycles (RAA): If the conflict is not easily resolved – say the weights between yes and no are nearly equal, or a veto norm conflicts with an equally absolute obligation norm (imagine an extremely rare tie of absolute duties) – the NES can enter a recursive deliberation. In the next cycle, it might do things like seek more information, re-evaluate the context, or adjust parameters (maybe increase the urgency if action is needed, or strengthen a norm weight if one is contextually more relevant) . This recursive process is essentially thinking twice: the system recognizes a stalemate or persistent conflict and loops again. During recursion, it can also call the conflict resolver in a meta way (maybe to decide which norm to relax if they are deadlocked) , or apply a meta-norm (like “in stalemate, choose the safer option”). It also might arbitrate multiple impulses by focusing on them one at a time if they were competing .

NES’s recursion has stopping criteria to avoid infinite loops: for example, it might allow only N cycles, or it stops if urgency is too high, or if additional cycles are yielding diminishing returns. If it ends the recursion without a clear “yes”, it defaults to withholding action (better to do nothing than a norm-violating something) .

Thus, NES handles conflicts in a robust way: explicit conflict detection (comparator oscillation or flagged norm conflict triggers RAA) , structured resolution (norm voting or re-weighting), and iteration if needed. It doesn’t assume one-pass decision-making; it acknowledges some decisions require reflection.

Global Workspace Theory: GWT deals with conflict in terms of competition for the global workspace. If multiple processes want attention or multiple action tendencies are activated, they compete (via things like lateral inhibition or attention biases) and typically one wins and gets broadcast. The ones that lose are suppressed, at least temporarily. This is a form of conflict resolution by competition and dominance – the strongest signal wins. However, if two contenders are close in strength, GWT might predict instability or oscillation: perhaps the system might flip between them (like in a Necker cube perceptual rivalry, two interpretations alternate because neither fully suppresses the other permanently).

In terms of action, that could mean indecision or vacillation. A GWT-based system might experience a “moment of hesitation” as different options battle for supremacy in the workspace until one crosses threshold. There isn’t an explicit loop that says “if conflict then do X” in classical GWT, other than that unconscious processes might continue to process and tip the balance or conscious deliberation might occur (the person might explicitly think through the pros and cons, effectively injecting additional info into the workspace until one option gains enough weight).

So conflict resolution in GWT is emergent: whomever shouts loudest (or is boosted by attention or reward signals) wins. In the brain, that might correlate to something like the basal ganglia decision threshold – similar to NES’s threshold concept, but not guided by normative logic, just by overall activation.

No formal recursion beyond maybe sequential thought: one content could enter workspace (“Impulse to do A”), then another content could come (“But maybe I shouldn’t because…”), which is akin to a recursive consideration but orchestrated by dynamic arrival of competing content. In effect, it might achieve a similar outcome (multiple cycles of consideration) but GWT doesn’t codify it as a separate mechanism; it’s just how conscious thought flows.

Predictive Coding: In predictive coding, conflict appears as prediction errors or competing hypotheses. The system resolves these by iterative updating. If two interpretations of data are possible, the system can oscillate or eventually pick the one that globally minimizes prediction error. This is done via gradient descent or message passing – effectively recursion until convergence (though not a discrete loop one can count, but continuous relaxation).

For action, if there’s a conflict between predicted outcomes of different candidate actions, an active inference agent might form a mixed policy until clarity is achieved or do a little exploration to reduce uncertainty. It’s a bit abstract, but one could say predictive coding handles conflict in a distributed negotiation manner: each hypothesis tries to reduce its error, and the equilibrium reached is the resolution (which might correspond to committing to one action plan).

If uncertainty remains high, predictive coding naturally keeps the system in an unstable state - which might correspond to dithering or random exploration. It’s not going to “decide by rule” like NES does; it will just follow the gradients.

So predictive coding has intrinsic iterative resolution (since it’s inherently iterative until errors reduce) but not a high-level conflict arbiter. There’s no meta-cognitive check “hey I’m conflicted, let me deliberately gather more info”. It just happens as part of the cycle: if error stays high, it means conflict persists, and the process continues automatically until something changes (external info or eventually noise pushes it one way).

NES’s approach is more strategic: it can detect conflict and possibly deliberately seek additional input (like consult norms more deeply, maybe query memory for an exception, or escalate to external advice if designed to). That’s something an engineered predictive coding agent wouldn’t spontaneously do unless that action reduces predicted error (which it might if the model knows asking someone reduces uncertainty).

Dual-Process Models: Conflict resolution is essentially their raison d’être in many contexts. They often describe conflict detection (System 2 kicks in when System 1 outputs conflict either with another impulse or with a goal). The classic example is cognitive conflict (Stroop word vs color, etc.). The dual-process resolution is: when conflict is detected, engage more deliberative processing to adjudicate.

However, dual-process doesn’t typically mention recursion beyond that initial override. It’s more like a one-shot: if automatic response seems fine, go with it; if it conflicts (with either another impulse or with a normative goal), then apply controlled reasoning to figure out what to do. After that, you act. It doesn’t propose you then conflict again and do another layer. In principle, one could imagine multiple rounds (you have an intuition, then a second intuition against it, then you deliberate, then you even question your deliberation with a meta-thought, etc.), but dual-process lumps most of that into “the controlled process.” It’s not hierarchical beyond two levels.

NES explicitly allows multi-step (which might be considered multi-level in effect, although practically it’s iterative on the same level until decision emerges).

Dual-process accounts of moral conflict (like Greene’s work) essentially say emotional intuition and rational analysis conflict, but they don’t formalize how that conflict is solved aside from which one is stronger or gets more weight by the person’s decision. Greene’s neuroscience findings suggest in personal moral dilemmas, emotional response often delays the decision as rational consideration battles it out – eventually one side wins. That’s consistent with what NES does: the Norm Conflict Resolver might need more time or recursion when a strong emotional veto norm conflicts with a utilitarian norm . Possibly requiring multiple cycles to come to a result (maybe the emotional side holds the gate closed until evidence or reasoning accumulates enough to override, or the system just times out and defaults to not acting harmfully). NES basically models that scenario.

So dual-process, while not algorithmically detailed, would align with NES in that conflicts between fast and slow considerations cause slower, more effortful resolution – which NES quantitatively captures (e.g. longer decision time when norms conflict strongly ).

Higher-Order Theories: These aren’t designed to handle conflicting impulses or thoughts – they address the existence of thoughts. If one has conflicting first-order desires, one could have higher-order thoughts about each (“I want X” and “I want Y” and I am aware of both). But how to resolve wanting X vs Y is not answered by HOT. Some philosophers (like Frankfurt, as mentioned) talk about identifying with one desire over another as a will formation process, but that goes beyond HOT into volitional theory. HOT itself is silent on resolving conflict except maybe one could say: whichever desire becomes conscious might have more weight because you acknowledge it – but actually sometimes making both conscious just highlights your ambivalence.

So HOT doesn’t provide a mechanism for conflict resolution or any recursion except the notion of a thought about a thought (which is a two-level structure, but not iterative beyond that typically).

ACT-R / Soar: These systems have explicit conflict resolution strategies for their production rules. In ACT-R, if two rules match simultaneously (meaning two possible actions can be taken), the architecture uses conflict resolution by comparing their utility values (learned goodness) or by specified priorities. One with higher utility will be chosen stochastically (there’s a noise parameter) in ACT-R. So that is a clear rule: pick the best action with some noise (so if they’re almost equal, it’s almost random who wins, reflecting indecision or trial-and-error).

If there’s an impasse (like in Soar, if no rule applies or multiple equally good apply and it can’t break tie), Soar triggers a subgoal (a new problem: “decide between these options”). That subgoal is essentially recursion: the system then uses knowledge (maybe look-up more info or ask for user input or simulate outcomes) to resolve the tie. Once resolved, it returns the result. This is directly analogous to NES’s recursive adjudication – except Soar’s criteria to go into subgoal is an impasse like tie or no knowledge, whereas NES’s criteria is conflict signals or norm stalemate.

Soar’s subgoaling is a form of algorithmic recursion and learning (once solved, it remembers the decision for next time). NES’s RAA is recursion but not necessarily learning each time (though presumably the experience might update norm weights gradually as some learning mechanism, or at least it solves the immediate conflict).

ACT-R doesn’t do explicit subgoals automatically (the modeler must program how to handle ties if needed, or just let the utility noise decide), whereas Soar automates it. So in terms of recursion:

* ACT-R: low-level (one-step selection, conflict resolved by a numeric comparison).
* Soar: hierarchical recursion for unresolved decisions.

NES is closer to Soar in having a purposeful recursion for unresolved conflict.

Additionally, ACT-R can do iterative re-evaluation through time if something changes – e.g. rules can be triggered in sequence to refine a decision. But it doesn’t have a loop for “no decision yet” because time in ACT-R is sliced by production firings, and typically an action is chosen each cycle.

NES explicitly models the case of no decision in the first cycle – something ACT-R would rarely allow (ACT-R would pick something even if random, unless you model indecision with a loop that cycles doing nothing until a condition changes).

So NES addresses a scenario many architectures ignore: the agent can pause in indecision and actively deliberate further. Many AI agents, especially older ones, either pick an action or require external input; NES says the agent can internally choose to delay and think more. That’s cognitively plausible (people often hesitate when conflicted), but not all models incorporate that (some assume a decision threshold might adjust but they don’t explicitly model the deliberation process beyond a slower reaction time).

Hierarchical RL: Hierarchical RL resolves conflict by a structure: a higher level policy chooses which lower-level policy runs. If two sub-policies could apply, presumably the high-level picks one based on expected return. There’s no simultaneous conflict execution – one option is chosen for execution. If that option fails or finishes, then it chooses again.

In standard flat RL, conflict between actions at a given state is resolved by the policy (e.g. pick the action with highest Q-value, possibly stochastic if a softmax is used). It’s akin to ACT-R’s utility comparison. If values are close, the choice may be essentially random or influenced by exploration. If values diverge due to learning or reward signals, the better is chosen. There’s no capacity for regret or mid-action reevaluation unless the environment provides new info that triggers a reevaluation at next time step.

However, some advanced RL includes planning or lookahead – like Monte Carlo Tree Search (not RL per se, but planning) which does recursion simulation to decide. If we consider that, an AI could simulate consequences of conflicting choices in a tree search and thus effectively do recursion in an internal simulation. That’s beyond basic RL though, more in the realm of hybrid planning.

NES’s conflict resolution with explicit evaluation of norms can be thought of as a special case of a multi-objective planning: it’s evaluating an action against a set of constraints (norms). RL doesn’t do that on the fly; it bakes it into the reward. But if we had RL with constraints, often you convert constraints into either penalties or treat it as an optimization under constraints solved by Lagrange multipliers (which might need iterative solving to find the right trade-off weight). This is analogous to what Norm Conflict Resolver does by summing weighted norms. The difference is RL would not normally have an if absolute then veto logic – that’s a non-differentiable decision (though one can enforce hard constraints by removing those actions from policy space entirely).

Executive Function Theories: Conflict resolution is central. The conflict-monitoring model (Botvinick) explicitly hypothesizes a monitor (ACC) that detects when simultaneous incompatible responses are active (like in Stroop both the word and color suggest different responses). When conflict is high, it signals to recruit more cognitive control (dorsal PFC increases attention to task). That is a form of meta-loop: conflict triggers an adjustment that then resolves the conflict by biasing one response more strongly. Over a few trials, this shows as reduced conflict (like the Gratton effect: after a conflict trial, next conflict trial is easier because control was upregulated).

This is somewhat like a mild recursion (carry-over adjustments rather than immediate loops). In a single instance, if conflict is detected, there’s evidence that within that trial the brain can adjust (e.g. mid-task adjustments) – but it’s limited by time. Usually conflict monitoring is about next-trial adaptation.

The Expected Value of Control (EVC) theory formalizes that the brain might compute, for a given situation, what level of control (additional focus, inhibition, etc.) yields best expected outcomes net of mental effort cost . It would then set that level. If one interprets “level of control” as something like how high to set the threshold or how much to bias toward goal-consistent information, that’s analogous to NES’s ability to adjust thresholds or weights in recursion . Indeed, NES’s blueprint acknowledges its RAA corresponds to doing a cost-benefit on continuing deliberation .

So executive theories provide pieces:

* Conflict detection triggers control (like RAA triggers recursion).
* A decision of how much control or whether to deliberate can be seen as computing expected value (NES says it stops after N cycles or if urgency is high – essentially implementing a cost of time).
* If conflict persists, strategies like focusing on the most critical aspect or defaulting to a safe no-action are similar to heuristics a human might use when torn.

No mainstream executive theory except Soar’s architecture explicitly mentions recursion or re-entrant processing. Most talk of one-step adjustment or continuous control allocation. NES is somewhat unique in framing it as a recursive algorithm (though that is a natural computer science approach to conflict resolution).

Summary on conflict/recursion: NES provides a structured conflict resolution algorithm with potential multiple iterations, whereas:

* Many frameworks rely on one-pass competition (GWT, RL, base ACT-R).
* Some allow hierarchical problem-solving (Soar’s subgoals, planning algorithms).
* Executive function theories emphasize adjusting control intensity rather than looping, but conceptually similar to doing more thinking for tough conflicts.

NES’s approach ensures that if a decision is not reached because of internal disagreement, the system doesn’t just randomly pick or freeze indefinitely; it actively engages in a resolution process, possibly repeatedly. This is reminiscent of how we sometimes stop and think harder when something isn’t clear, or reconsider our values if two values clash (maybe coming up with a compromise or prioritization). It’s bridging a gap: mechanical systems often either thrash or choose arbitrarily under conflict, whereas humans often deliberate or seek more info. NES tries to model that deliberative step.

From a philosophical view, this also prevents infinite regress by capping recursion (no endless homunculus because it stops or defaults after limited cycles). It becomes a finite, well-defined process.

Thus, relative to other models:

* NES and Soar share the idea of explicit conflict resolution and iterative subdecision.
* NES and conflict-monitoring theory both have triggers for deploying extra control on conflict detection.
* NES goes beyond by integrating norm-based conflict resolution (not present in general exec theories) and by clearly specifying the loop rather than leaving it informal.

### **6. Integration of Cognition, Emotion, and Value**

NES: The Normative Executive System is explicitly designed to integrate cognitive evaluation, emotional modulation, and value-based (normative) criteria into one decision process. Several elements illustrate this integration:

* The Comparator accumulates evidence for an impulse. This evidence can come from cognitive appraisal (e.g. assessing outcomes or benefits) and also from emotional salience signals (the blueprint mentions affect influencing drift rates) . For example, an impulse that is very appealing (pleasure) might accumulate evidence faster, whereas one that triggers fear might accumulate evidence slower or negative evidence, unless norms counteract.
* The Assent Gate is modulated by an emotion-like parameter (5HT\_level analogous to serotonin) and by emotional states . “Hot” emotions like anger or excitement can lower the threshold (making the system more impulsive), while “cold” moods like caution or anxiety might raise the threshold (making it harder to act) . This is a direct integration of affect into the executive gate. It means the state of emotion can tilt the balance between impulse and restraint – consistent with how humans may act rashly when angry (low threshold) or freeze when afraid (high threshold).
* The Norm Repository provides a value context: these internalized values guide the cognitive evaluation (via Norm Conflict Resolver). So values are not separate from cognition here; they are part of the processing of the impulse’s acceptability.
* The RAA (recursive process) monitors conflicts which could be between rational calculations and emotional responses. Indeed, the blueprint explicitly ties its model to dual-process: an emotional veto norm vs a calculated utilitarian norm interplay . It also references integrative theories like EVC which inherently combine emotion (expected outcomes, which include value valences) and cognitive control .
* NES’s design ensures that what “feels right” (emotion) and what “is right” (norm) and what “makes sense” (practical evidence) all influence the final decision. For instance, if an action is strongly prohibited by a norm but the impulse is driven by a strong desire (emotion), the system registers both: the desire yields a lot of evidence in Comparator, but the norm triggers a veto in Conflict Resolver, and the result might be no action but the conflict is felt as tension (which in a human might be an emotional experience of conflict, possibly even conscious as cognitive dissonance).
* Because NES aims at implementation in either a brain or AI, it attempts to map these components to known subsystems: e.g. it mentions analogies like ACC for conflict (often considered part of affective pain of conflict), serotonin for inhibitory control (neuromodulator linked to mood and impulse), attention schema (which relates to conscious awareness of where focus is, bridging cognition and awareness) .

So NES is a holistic model: cognition (logical weighing of options), emotion (via threshold and salience), and value (norms) converge. This means it can account for scenarios like:

* Emotional bias: If one is angry, threshold lowers and one might assent to an impulse that normally norms would block, unless norm veto is absolute (in which case, conflict intensifies but threshold is low, could lead to a quick but potentially regrettable action unless norm veto absolutely prevents it).
* Value alignment: If one has a strong value against something, even a high emotional desire might not trigger action because the norm veto stands firm.
* Emotional learning: If repeatedly an impulse cause guilt (norm violation emotion), the Norm weight might increase or impulse salience might decrease next time (some feedback, though specifics depend on learning which NES says can happen via RPE).
* Cognitive reframing: The recursion might allow cognitive re-appraisal (like thinking again in a calmer way if first reaction was emotional).

Global Workspace Theory: GWT doesn’t explicitly unify emotion and cognition; it’s more an architectural concept. But in practice, GWT proponents (like Dehaene) have noted that emotional stimuli often have advantage in accessing the workspace because they are salient. Emotional circuits (amygdala, etc.) can tag information with value and boost its chances of winning attention. Also, once something is broadcast in the workspace, both analytical processes and emotional associations can act on it simultaneously (it becomes available to the whole brain, including emotional appraisal systems in limbic areas and cognitive reasoning in PFC). So GWT inherently allows integration by broadcasting to all. It is often said that consciousness (the workspace) is where disparate information (sensory, memory, emotion, etc.) meet and are integrated.

However, GWT doesn’t prescribe how emotion influences the competition exactly; it just acknowledges any process can contribute. Researchers have extended GWT in models with emotional modules that bias the global workspace entries via salience or reward signals (the “attention/emotion” network providing context to the workspace).

So in GWT, integration happens through global sharing: if an emotional reaction enters the workspace, it becomes conscious and can influence decisions; if a cognitive analysis enters, it can incorporate emotional context if that was also present or known. But there’s no structured algorithm specifically balancing them – it’s all in the emergent mix.

Predictive Coding: In predictive coding, emotion and cognition are theoretically unified under the same predictive framework. Emotions could be treated as predictions about interoceptive states (bodily signals). For example, Lisa Feldman Barrett and others have argued emotions are the brain making sense of body states with predictive models. In active inference, all perception and action, including physiological regulation (like heart rate, hormone levels), are under one imperative (minimize surprise). So an emotional state like anxiety might be the result of predicted danger, which also influences how one perceives options (biased towards pessimistic predictions). Conversely, a confident mood might reduce perceived risk.

Because it’s one big system, predictive coding doesn’t separate cognitive and emotional processing – it’s all just processing at different hierarchical levels with different kinds of predictions (some about factual external variables, some about body state and reward). Value comes in as prior preferences (i.e. an intrinsic bias of the model to certain outcomes). In that sense, value, emotion, and cognition are integrated mathematically: the same equations that update beliefs about the world can include a term for “utility” or “preferred states” that biases them.

However, in practical modelling, often perception (cognition) and emotion (affect) might be handled in separate sub-networks that interact through some shared variables (like an expected reward prediction influences motor outputs). It’s unified theoretically but implemented sometimes distinctly (some models have separate interoceptive stream, etc.).

What predictive coding is strong at is showing how expectations (which embody value if they encode anticipated reward) shape perception and how observations shape feelings if they violate expectations. But it doesn’t provide an easy handle to say “emotion raises threshold” or such; rather, emotion is threshold in some interpretations (precision can be modulated by neuromodulators which have emotional correlates).

Active inference literature does discuss something like “expected precision” might correlate with confidence or anxiety. If you’re anxious, you might assume high volatility (low precision on predictions, so you keep being in surprise mode, which can freeze you). That loosely maps to how NES says fear raises threshold (i.e. you need more evidence because you trust the world less, so you hesitate). So the ideas align, but predictive coding wraps it into the statistical param tuning.

So yes, predictive coding provides a common currency (prediction error) and all factors (cognitive evidence, reward, emotion) ultimately manifest as modifications of that currency. It’s elegant but a bit opaque to disentangle for a specific scenario without running the model.

Dual-Process Models: By definition, they often separate emotion and cognition into two different “systems.” System 1 includes emotional, instinctive reactions; System 2 is deliberative and often emotion-free or even works to override emotional bias. Many dual-process discussions revolve around the tension or cooperation between these two: e.g., in decision making, an emotional framing might lead System 1 to a different choice than rational calculation would – if conflict is detected, System 2 might override, but if not, the emotional influence might just carry the decision.

So integration in dual-process is often not simultaneous blending, but sequential or competitive: either the emotional heuristic leads, or the analytic reasoning leads. Some models describe them working in parallel and the result is a combination (e.g. judgment might be a weighted sum of an intuitive answer and a corrected answer if time permits). But typically one is dominant.

Dual-process frameworks do allow that in many everyday decisions, the two systems agree or one influences the other – e.g., one might consciously reason using feelings as data, or emotions might set the options and then reason picks among them. But since it’s not formal, integration is scenario-dependent.

NES by integrating them in one architecture (with interplay between norm/emotion and evidence) arguably moves beyond a simple two-system view to a more unified system with multi-faceted inputs. It shows how an “emotional norm” (like an internalized empathetic rule) might operate like a System 1 (fast veto), whereas a utilitarian count of outcomes is System 2, but they feed into one decision mechanism. That’s actually a form of integration: both influence the gate’s decision rather than two separate outputs requiring a second conflict resolution stage. In NES, the conflict is resolved within one system, whereas dual-process often posits two systems then requiring a vague resolution process.

Higher-Order Theories: They don’t specifically handle emotion vs cognition, but one could say you can have higher-order thoughts about emotional states or about cognitive states. If an emotion becomes conscious (via a HOT), now cognitive appraisal can happen on it. But that’s just making emotion part of conscious content. HOT doesn’t integrate them into a decision per se; it’s more about making them explicit.

Philosophically, one might argue our sense of self or conscious will involves integrating our feelings and our reasons (like “I feel like doing X but I think I should do Y, and I as a self consider both”). HOT doesn’t give that process, it only assures you know what you feel and know what you think if those become higher-order.

So not much to say for integration – HOT is neutral, it can encapsulate both feeling and thought under “mental states” that can become conscious, but doesn’t say how they interact to cause behavior.

ACT-R / Soar: Historically, ACT-R did not include emotion in its core; it was oriented towards rational task performance (like solving math, remembering lists). However, extensions to ACT-R have been developed to include an emotional module (e.g. ACT-R has had modules proposed for fatigue, stress, etc., which modify parameters like utility or activation). One concept: utility learning in ACT-R can be seen as integrating value (if something yields positive feedback, its rule’s utility increases). That’s a bit like reinforcement learning.

So values can be encoded as goal priorities or utility values. Emotions can be simulated by altering parameters: e.g., high stress might reduce retrieval accuracy (to simulate choking under pressure), or fear might be modeled by injecting certain goal or utility biases.

So integration in ACT-R is not inherent, but possible:

* Cognition: represented by symbolic processing, logic.
* Emotion: must be represented either as special symbols (like a chunk “FEAR=HIGH”) that can affect processing or as numeric modulations of base-level activations or utilities.
* Value: represented either as part of the goal structure or as the reward signals shaping utilities.

For instance, ACT-R could model a person’s internal conflict: one production is triggered by a desire (with some utility), another production represents a moral rule (with another utility). The one with higher utility wins. If the modeler sets those utilities appropriately (maybe moral rule usually has higher unless temptation has extremely high context-specific utility), then the system effectively integrates value and desire by whichever dominates the utility. That’s a simple integration (like a weighted sum in utility computation).

However, ACT-R doesn’t spontaneously generate an emotional state like anger that then globally lowers thresholds. Unless the modeler explicitly codes: “if angry chunk is active, add bias to these rules / lower utility of cautious rules,” etc.

So integration in such architectures is manual and not central to their design. They excel at cognitive, but emotion had to be patched in by research efforts (some frameworks like CLARION tried multi-tier with motivational subsystem separate from cognitive).

Soar similarly can incorporate emotional factors as preferences or as meta tags on states (though less literature on Soar and emotion, to my knowledge). But nothing in base Soar addresses affect.

Hierarchical RL: RL by default integrates everything into the reward function. If by “cognition” we mean planning and by “emotion/value” we mean reward, RL merges them by policy optimizing expected reward. Emotions as a separate concept usually aren’t present, but one could interpret certain algorithmic features as analogous to emotional states (like an exploration rate might correspond to curiosity or frustration if high).

In hierarchical RL, a higher-level might encode something akin to a goal that reflects a value (like “safety-first” vs “achievement”). But normally, the hierarchy is for subtask efficiency, not for separating rational vs emotional processes.

One could design a hierarchical agent with one component being a fast heuristic (like an instinctual controller) and another being a slower planner, and have a meta-controller decide when to use which – this would mirror dual-process in RL form. That has been explored in some meta-RL contexts. But mainstream HRL doesn’t specifically do that (it’s more like stepwise abstraction, not qualitatively different systems).

So integration in RL is often trivial in concept (all influences just become numbers in reward), but that can be a weakness too: it struggles with cases where a hard constraint (norm) should not be violated even for large reward, because if you represent it as a finite penalty, extreme cases might still break it. RL doesn’t natively have “sacred values” concept, whereas humans do treat some values as not tradable (NES explicitly models that via veto).

Executive Function Theories: They often treat emotion as something that can hijack or burden executive control. For instance, there are distinctions like hot executive function (in emotional contexts, like delaying gratification with something desirable present) vs cool executive function (abstract contexts, like solving a puzzle in a calm setting). It’s known that emotional arousal can impair certain executive operations or change strategy. But these theories often consider them separately then compare. They don’t necessarily have an integrated model except saying PFC interacts with limbic system (e.g., top-down regulation of emotion, or bottom-up capture of attention by emotional stimuli).

An integrative view in neuroscience is that the ventromedial PFC integrates affective value with decision, whereas dorsolateral PFC handles abstract logic and working memory. These regions then together (with ACC) weigh decisions. That’s somewhat integrated in the brain architecture sense. So a model might have separate pathways for emotional vs rational evaluation that converge in a final decision node. This is similar to dual-process again but in neuro terms.

The EVC model (Shenhav) deals a bit with cost of effort vs value of goal, which is an integration of a kind (valuing cognitive effort is adding an emotional/effort cost into a cognitive control decision). But it doesn’t incorporate, say, anger or joy as variables – it’s more task-value oriented.

NES’s integration vs others: NES does something quite ambitious: unify rational deliberation (evidence accumulation), emotional state (through threshold modulation and likely through initial impulse strength as well), and normative values (explicit rules) in one loop. Many frameworks only handle one or two of these at a time:

* Traditional AI: rational (ACT-R: rational, RL: rational in reward sense).
* Emotional models: like pure emotional decision (some models like Loewenstein’s visceral factors, but those are descriptive).
* Normative reasoning: e.g. deontic logic systems in AI that handle rules but often ignore emotional factors (they just check constraints logically).  
    
   NES tries to unify all three influences – something quite needed to model real human decision-making but rarely achieved in a single model.

We see in the blueprint references to how it matches empirical phenomena that are inherently about integration:

* Personal moral dilemmas: emotional aversion vs utilitarian reasoning (NES reproduces longer RT and outcome patterns ).
* Effects of serotonin: linking neurochemical affects (often tied to mood/impulse control) to decision thresholds .
* ACC conflict signals: ACC is considered part of the limbic system (monitors conflict and also registers it as a form of distress), bridging cognitive conflict and affect (it’s sometimes called the “oh no!” area when conflict occurs, indicating an emotional response to conflict). NES monitors conflict signals akin to ACC – implying the model includes an internal recognition of conflict which has both cognitive (need for control) and affective (discomfort) aspects.
* “Thermostat regulating temperature – a clear feedback system, not an intelligent ghost” – thermostat analogy: it integrates a measured value (temperature) with a reference (desired temp) and yields action (heat on/off). Here conflict or error is measured and yields control adjustments. In a home thermostat, there’s no emotion; in a brain, conflict often comes with negative affect. But mechanistically it’s integrated.

So, overall NES stands out in explicitly merging emotion and values into the cognitive control algorithm. This contrasts with older models that might treat emotion as noise or disturbance to rational processing, whereas NES sees it as a parameter in the decision threshold. It also differs from purely emotional or value-driven models by keeping a rational evidence accumulator in the loop.

In simpler terms: NES’s agent would feel something about its choices (through threshold changes and conflict signals) and think about them (accumulate evidence, recall norms) and the outcome is a product of both feelings and thoughts within a normative frame. That’s very human-like in aspiration.

Other frameworks typically either:

* Focus on thought and minimize feeling (classical cognitive models).
* Or focus on feeling and ignore complex thought (some affective neuroscience or behaviorism).
* Or treat them as separate processes (dual-process).

NES tries to computationally marry them.

This integrated approach can generate richer predictions: e.g. if someone is in a manipulated emotional state (fearful or calm), NES would predict different thresholds and hence different rates of impulsive vs normative decisions. That’s testable (fear tends to make people more risk-averse and stick to default rules; anger might make them more risk-seeking and break some rules – indeed psychological studies show emotion-specific effects on moral judgment: anger can increase punitive judgments, disgust can increase strictness on purity norms, etc.). NES could in principle model those by modulating which norms get emphasized or threshold adjustments with different emotions.

Conclusion: NES demonstrates a tightly integrated approach to cognition, emotion, and value where:

* Cognitive evaluation provides information about outcomes.
* Emotions adjust the urgency/restraint parameters of decision-making.
* Values provide criteria to judge the impulses.  
    
   All simultaneously influence whether an action is taken or not.

Other frameworks either do not incorporate all these categories or do so in a less unified way. NES’s integration is one of its unique strengths, aiming for a comprehensive model of human-like decision processes where reason, feeling, and moral rules jointly determine behavior.

### **7. Learning, Adaptability, and Trainability**

NES: The blueprint explicitly outlines mechanisms for learning norms and adjusting control parameters. Key points:

* Norm Acquisition: NES can acquire new norms through multiple pathways . It can take explicit instruction from authority (like a parent telling a child a rule) and directly add that to the Norm Repository with a high weight . This is a kind of supervised or knowledge-based learning – effectively someone programs a new norm. NES architecture supports that by allowing dynamic updates to the repository (functions like add\_norm or update\_norm\_weight) .
* Reinforcement & Internalization: NES uses Reward Prediction Error (RPE) signals to infer norms from experience . If an action leads to unexpected punishment or negative feedback (social disapproval, personal bad outcome), the system generates a large negative RPE. The blueprint says the agent can interpret that as evidence that this action is disallowed, prompting it to hypothesize a new norm “don’t do that” . Over time, repeated experiences will adjust that norm’s weight (if consistently negative outcomes, strengthen it). Conversely, unexpected positive outcomes can lead to inferring a positive norm (“this is good to do”).  
  + This is essentially a model of social norm learning akin to how humans learn via reinforcement which behaviors society rewards or punishes . It’s RL-like internalization rather than just adjusting policy for immediate reward. The agent actually forms an explicit rule that it then follows even if immediate reward might tempt otherwise (because it internalized the pattern of consequences).
* Bayesian learning and imitation: The blueprint also mentions observing others and updating beliefs about norms in effect . The agent can use Bayesian inference – treat witnessing punishments or rewards given to others as evidence for certain norms existing in that context (like seeing someone scolded for loud noise indicates a norm of quiet in library) . This allows learning without direct experience, via vicarious or cultural transmission.
* Updating Norm Weights: The architecture supports adjusting the strength (weight) of norms as experience accumulates . For example, if a norm is often violated without much negative consequence, the system might downgrade its weight; if a norm violation consistently leads to big punishment, weight increases.
* Emotional calibration: NES’s threshold modulation by serotonin analog suggests an ability to adapt to chronic mood changes or pharmacological intervention. If an agent’s baseline 5HT level changes (maybe due to learning or external factors), it permanently alters impulsivity. That might not be “learning” in a normative sense, but it’s an adaptability of control style. Perhaps through experience, the agent might adjust its baseline threshold for certain contexts (like learning to be more cautious at night, effectively raising threshold in that context).
* Recursive improvement and meta-learning: The RAA itself is a kind of meta-cognitive loop that could be modulated by learning. For instance, the agent might learn after repeated deliberations that certain conflicts aren’t worth too many cycles (costly and rarely change outcome), so it might tune the stopping criterion or urgency parameter. While the blueprint doesn’t explicitly detail meta-learning of RAA rules, it suggests the recursion is rule-governed but presumably those rules could be optimized (like expected value of control learning could refine how many cycles to attempt for what kind of conflict).
* Trainability by external intervention: Because NES’s norms are explicit, an external user (like a developer or a teacher) can modify them to shape the agent’s behavior. For instance, if an AI with NES is not following a needed safety rule, one can insert that rule with high weight and maybe simulation punishments to ensure it’s internalized. This is easier than trying to indirectly shape behavior by reward tuning (like in pure RL). It’s akin to programming in constraints or doing value alignment by providing the values directly.
* The Alignment Guide likely elaborates how to align the NES’s norm set with human values (through training, audits, etc.). While we haven’t read it fully, it implies a methodology to ensure the Norm Repository is loaded with the “canonical” values (maybe through iterative training with human feedback).

So overall, NES is highly trainable:

* It learns from experience (like an RL agent but forming norms not just habits).
* It learns from instruction (like a symbolic system).
* It can adapt its internal parameters (norm weights, thresholds) based on context and feedback.
* It also can incorporate new information quickly through authority input, which is something typical RL struggles with (you usually have to shape rewards and let it iterate many times; NES can one-shot learn a rule from one telling).

Global Workspace Theory: GWT itself is an architecture and doesn’t specify a learning mechanism. However, one argument is that making information conscious in the workspace facilitates learning because it then is globally available to any learning mechanism. Bernard Baars suggested that the global broadcast can lead to widespread synaptic updates (the “teaching machine” idea: conscious focus leads to forming long-term memories or learning new skills by connecting different modules). In Dehaene’s GNW, he proposes that consciousness is needed for certain types of learning like chaining operations (as in complex tasks).

* So one might say learning in a GWT system happens as a result of iterative practice where relevant info is attended and broadcast, allowing memory formation and adjustments in various specialized processors. For example, when you explicitly focus on a mistake and correct it, that conscious act may strengthen the correct action path for next time (because you broadcast the error and the correction widely, so associative learning can encode it).
* There have been computational models (like the IDA/LIDA cognitive architectures) that incorporate learning mechanisms (like forming new memory chunks or reinforcement learning) triggered by conscious episodes.
* GWT doesn’t restrict learning, but it’s not built-in: one must add whatever learning algorithms (Hebbian learning, backprop in neural implementations, etc.) in the modules around it. So adaptability in a GWT system depends on those added components. A typical scenario: the system tries a solution, if it fails (feedback is received), the conflict or error is broadcast, and a learning rule updates some connections (like basal ganglia might adjust the probability of that action next time).
* GWT also doesn’t have any direct representation of knowledge like norms that can be easily updated. It’s more low-level in that sense. Knowledge is distributed in the system (in synapses or in symbolic memory that’s accessed).
* For training, an external trainer can influence a GWT system by controlling what information enters the workspace (like instructing it, showing salient cues) or providing reward/punishment signals that the learning modules can use. But it’s not as straightforward as inserting a rule, unless the architecture has a symbolic component.
* If the GWT is implemented as a production system (like Global Workspace Architectures with rules), then training can include adding new rules or adjusting existing ones. But as far as GWT theory goes, it’s neutral on that.

Predictive Coding: Predictive coding is inherently a learning system – it continuously updates its model parameters (synaptic weights) to better minimize prediction error. That is typically done via some gradient descent or Bayesian update. Over time, such a system adapts to the environment: it learns statistical regularities, cause-effect relationships, etc. So it is very trainable in the unsupervised sense: expose it to data, it will adjust beliefs to fit data.

Active inference also learns policies by updating expected outcomes (some approaches do policy learning, though the core principle is state estimation – but there’s an extended formalism for learning the model structure or policy).

* It can also learn reward structure if formulated in (like learning what behaviors yield less surprise).
* If something changes in the environment, predictive coding will adapt until prediction errors for new patterns reduce (so it’s inherently plastic).
* However, pure predictive coding might have trouble learning abstract rules that are rarely violated (if an agent never violates a norm, it might not “notice”(The answer continues in the next message due to length.)

### **Learning, Adaptability, and Trainability (continued)**

Predictive-coding engines **learn continuously**, retuning synaptic weights so that future input generates smaller prediction error. When an unexpected social sanction follows an otherwise self-rewarding act, however, a vanilla error-minimisation loop will merely raise the expected penalty for that act and, if the long-term reward outweighs the new cost, still perform it. Without an explicit *categorical* representation of “forbidden”, the system forever treats every rule as a **soft constraint**—just another term in a scalar cost function. NES instead stores newly inferred prohibitions in its **Norm Repository**, flags them as *veto-eligible*, and thereafter **blocks** the action even when short-horizon gains remain attractive. Norm acquisition thus becomes a one-shot structural update rather than a slow tweak of Q-values.

Dual-process accounts say little about learning beyond noting that System 1 habits form through repetition and System 2 deliberation is trainable by practice. NES makes that claim operational: RPE-driven weight change stiffens or relaxes each norm; Bayesian observation updates raise or lower a prior that a rule applies in a context; recursive self-audit re-prioritises norms that collide too often. The overall effect is a slowly moving but **internally coherent** value hierarchy rather than a mere accumulation of habits.

Higher-order theories are agnostic on acquisition. ACT-R and Soar already offer reinforcement and chunking mechanisms; NES could piggy-back on them by storing norms as declarative chunks whose *utility* parameters are updated exactly as the Blueprint prescribes. Hierarchical RL also learns, yet everything is folded into the same reward gradient—hard constraints have to be smuggled in as prohibitively negative rewards. NES’s veto flag supplies a cleaner engineering affordance: drop a single line in the repository, and the impulse is henceforth impossible, no retraining required.

Overall, NES yields a **dual-track plasticity**: slow, value-driven consolidation of the norm set, and faster, serotonin-/urgency-mediated tuning of thresholds that govern moment-to-moment impulse control. That bifurcation mirrors empirical evidence that moral development unfolds over years while inhibitory parameters can shift within minutes under pharmacological or emotional load.

### **Homunculus Problem and Self-Model**

Predictive coding dissolves homunculi by reducing agency to network dynamics—yet at the cost of explanatory traction on the felt **moment of choice**. Classical architectures (Baddeley’s Central Executive, early dual-process “System 2”) avoid the regress only by fiat: they declare a top box and leave its internals blank. HOT moves the ghost one level up: a higher-order state “observes” the first-order state, but how that observer chooses anything remains mysterious.

NES answers the regress in a **layered-but-finite** style. Evidence integration, threshold gating, and norm arbitration are each algorithmically elementary; none contains further miniature governors. The **Recursive Adjudication Agent** is not a captain giving orders but a *loop condition*—a rule that either reiterates the cycle or terminates it under explicit cut-offs (time, unresolved veto clash). Because every loop consumes real latency and metabolic cost, infinite iteration is impossible; agency bottoms out in a concrete halt instruction rather than in an unexplained fiat. The architecture therefore secures both Stoic unity (“one faculty rules”) and modern mechanistic humility (“many idiots, no inner king”).

The self-model emerges implicitly: the system logs which impulses were tabled, which norms overruled them, and which conflicts warranted recursion. Those traces constitute the **narrative the agent can later recall as ‘my reasons’**, supplying the phenomenology that higher-order theorists attribute to meta-representation without requiring a separate observer module.

## **NES in the Comparative Landscape**

NES’s distinctiveness crystallises at the intersections the other frameworks leave under-specified. It retains the empirical plausibility of conflict-monitoring loops, the mathematical rigour of drift-diffusion decision theory, and the flexibility of rule-based cognitive architectures, but **splices in an explicit normative backbone** that can host absolute prohibitions alongside weighted preferences. Where predictive coding offers seamless but value-neutral inference, NES bolts in a value grammar. Where dual-process stories describe veto without mechanism, NES supplies threshold math. Where ACT-R and Soar deliver code without conscience, NES provides the conscience.

By granting *veto primacy* to certain rules, NES also rectifies a limitation of scalar-reward approaches in AI safety: it forbids prohibited behaviour even when the expected utility of transgression is astronomical—a property ordinary RL can simulate only by ill-conditioned penalties. That makes NES a promising scaffold for **alignment-critical agents** whose designers cannot afford soft constraints.

Empirically, the blueprint predicts:

* longer reaction times exactly when veto rules and utilitarian weights nearly balance (mirroring Greene’s fMRI findings on personal moral dilemmas),
* serotonin-linked threshold shifts matching the behavioural impulsivity literature, and
* ACC-indexed conflict bursts at recursion onsets—all testable with existing tasks.

No competing model presently combines those three quantitative signatures in a single simulation.

## **Conclusion**

The **Normative Executive System** operationalises the Stoic ruling-faculty intuition in modern computational terms, filling conceptual lacunae across cognitive science and AI. Against **Global Workspace** it contributes a determinative *pre-broadcast filter*; against pure **Predictive Coding** it introduces categorical norms; against **Dual-Process** it replaces declarative folk-psychology with algorithmic anatomy; against **HOT** it shows how self-awareness can be a by-product of audit logs; against **ACT-R/Soar** it supplies moral semantics; against **Hierarchical RL** it secures deontic guarantees instead of scalar expediencies; and it concretises the **executive-function** vocabulary into code-ready modules.

In sum, NES is not merely another cognitive architecture but a **synthetic hinge** between rational computation, affective modulation, and ethical constraint—precisely the triad any robust theory of agency must reconcile.