Agent Complexity Theory (ACT): Predicting Agentic Architecture Trade-offs from Intrinsic Task Structure

Shreyan Basu Ray
basurayshreyan@gmail.com

July 2026

Abstract

We propose Agent Complexity Theory (ACT), a computational theory for agentic architectures inspired by classical complexity theory. This characterizes an agentic workflow by two objects: the intrinsic difficulty of the task, measured independently of any model, and the complexity signature of the architecture, measured as growth in that intrinsic difficulty. An architecture is treated as an abstract machine, a graph of roles, exactly as an algorithm is treated in classical analysis. A task is described by its minimal decision DAG, whose node count is a work size n_W and whose longest path is a span size n_S. Every architecture then has a signature (A_E, A_L, A_C, A_V) for economic, latency, communication, and verification complexity as functions of (n_W, n_S). We derive these signatures step by step for the four standard machine families and prove matching lower bounds, and we give an explicit annotation protocol that makes the intrinsic sizes reproducible on benchmark tasks. We then apply the theory to ten representative production agentic workflows spanning the dominant architectural families: four as complete mathematical case studies (Claude Research, OpenAI Deep Research, mini-SWE-Agent, Devin) and six as compact supporting analyses, in each case deriving the signature from the public architecture, computing the prediction, and reconciling it against the system's published numbers. Within each task class where numbers are comparable, the predicted ranking of architectures agrees with the observed ranking, and two of the reconciliation tests are quantitative: the debate machine's predicted cost growth exponent band in the number of agents contains all three measured exponents, and the predicted production token multiplier range contains the reported value, with one documented failure (per-round debate growth) that localizes a specific missing term. We close with a fully worked design example that carries a practitioner from a task's decision DAG to an architecture choice using only the formulas in this paper, six design studies (retrieval-augmented assistance, voice, research, technical writing, product testing, customer support) in which candidate graphs for the same task are evaluated with the full substitution arithmetic shown and one is selected by the theory, and a plain statement of what ACT does not do. We do not claim ACT finds a provably optimal architecture; we claim it provides a principled, testable way to compare candidates under common complexity metrics, before anything is built.

1. Introduction

Why complexity theory? Classical complexity theory asks how computational resources grow as problem size increases on an abstract machine, and that one question reorganized computer science: it separated properties of problems from properties of implementations, and it let engineers reason before building. Modern AI systems increasingly solve problems through orchestration rather than through a single algorithm, yet the resources they consume are evaluated empirically, per configuration, after implementation. There is no adopted framework for reasoning about the intrinsic difficulty of an agentic workflow before it is built. This paper proposes such a framework, not by analogy-mongering but by construction: agent architectures are treated as abstract machines, task difficulty is defined by an intrinsic decision structure, and complexity is the growth of resources in that structure. ACT is inspired by classical complexity theory and extends resource-growth analysis to agentic workflows; whether the abstraction earns the comparison is for the results, and for readers, to decide.

Algorithm analysis lets an engineer choose merge sort over insertion sort before running either, because complexity theory measures how cost grows in an intrinsic input size on a fixed abstract machine. Agentic systems have no such instrument. A team choosing between a single agent, an orchestrated tree, a pipeline, or a debate discovers cost and

latency only after building and benchmarking each candidate. We propose Agent Complexity Theory to supply the missing instrument.

ACT rests on two moves. The first is to treat an architecture as an abstract machine: a graph of roles with a topology and a verification policy, playing the part the algorithm plays classically. The second is to measure task difficulty intrinsically, by a quantity that does not shrink when the underlying model improves, so that the resulting complexity is a property of the task and the architecture rather than of the vendor or the model generation.

The contributions are presented in the order theory, derivation, prediction, validation, worked example, practice.

(i) **Theory.** An abstract machine for agentic computation (Section 3) and an intrinsic task size, the minimal decision DAG with work size n_W and span size n_S (Definition 2), together with an explicit annotation protocol that makes the sizes reproducible (Section 4).

(ii) **Derivation.** Complete, step-by-step derivations of the four-coordinate complexity signature for the single agent, the pipeline, the tree, and the debate machine, with matching lower bounds (Section 5).

(iii) **Prediction.** Four falsifiable predictions derived, by substitution into the theorems, before any data is consulted (Section 7).

(iv) **Validation.** Ten representative production workflows, spanning the dominant architectural families, reconciled against the predictions: four complete case studies (Section 8), six compact ones (Section 9), and a reconciliation organized as ordinal validation, quantitative validation, and failure analysis (Section 10).

(v) **Worked example.** A start-to-finish design exercise, with the constrained optimization carried out explicitly, showing how a reader applies ACT to a new task (Section 11), followed by six design studies across the workloads enterprises most commonly automate, each selecting among candidate graphs by the formulas (Section 12).

(vi) **Scope.** An explicit statement of limits (Section 14) and the research program the framework opens (Section 15).

2. Related Work: From Classical Complexity to Agentic Cost

The literature ACT draws on forms a chain, and we present it as one: complexity theory supplied the intrinsic-size discipline, parallel computing supplied the work-span laws, communication complexity supplied the message-counting coordinate, agent systems supplied the machines, and the recent cost measurements supplied the data points that a theory must organize. ACT is the last link: it connects the ends of this chain.

2.1. Classical Complexity and Parallel Computing

The mathematical spine of ACT is sixty years old, and we consider that a feature. Complexity theory's founding move was to measure growth in an intrinsic input size on a fixed abstract machine, and Kolmogorov's definition of intrinsic description length [5] supplies the epistemic template for size measures that are well defined yet uncomputable: our intrinsic sizes are of exactly this kind (Proposition 1). Parallel computing then made resource growth two-dimensional: Amdahl bounded the speedup of a partially sequential workload [1], Brent bounded the evaluation time of an expression DAG by its total work and its critical path [2], and Graham bounded list scheduling within a factor of two of optimal [3]. ACT restates these results with the task's decision DAG in the role of the expression DAG and the agent architecture in the role of the processor pool: the work-span decomposition of Section 3 is Brent's, and the intrinsic parallelism bound n_W/n_S of Corollary 2 is Brent's bound in task-intrinsic form. Yao's communication complexity [4] established that the messages a distributed computation must exchange are a resource in their own right, which is why the signature of Definition 4 carries a communication coordinate A_C alongside work and latency.

2.2. Agent Systems and Their Architectures

The machines ACT analyzes were defined by the systems literature. ReAct established the sequential reason-act loop [15]; SWE-agent added a structured agent-computer interface to it [12]; OpenHands built a planner-executor platform around an event stream [14]; AutoGen made multi-peer conversation, including debate, a first-class pattern [16]; LangGraph exposed the general role-DAG as a programming model [17]; and CrewAI popularized sequential role pipelines [18]. Production systems then composed these patterns at scale: Anthropic's research system as

an orchestrator-worker tree [6], OpenAI's Deep Research as a coordinated research hierarchy [7], and Devin as an autonomous planner over executors [13]. The SWE-bench Verified leaderboard [10] provides the one public benchmark where several of these machines are scored on the same tasks, and GAIA plays the same role for assistant-style research [8].

2.3. Measurements of Agentic Cost

A rapidly growing empirical literature measures what these machines spend. Anthropic reports a 90.2% outcome gain over a single agent at about $15\times$ the tokens of a chat, with 3 to 5 subagents in a flat tree [6]. Liu et al. measure debate cost growing by factors of 36 to 49 as agents scale from 1 to 8, and by factors of 17 to 29 as rounds scale from 1 to 4 [20], with related scaling in [21]. Kim et al. measure the token overhead of four multi-agent topologies at matched accuracy [22]. Tran and Kiela show single agents beating multi-agent systems at equal thinking-token budgets in closed-world settings [23], and Du et al. document the accuracy gains that motivate debate despite its cost [19]. We regard this entire body of measurement as *orthogonal empirical work* rather than competing theory: it measures what deployed configurations spend, and ACT supplies the coordinate system in which those measurements become comparable, transferable, and predictable. Nothing in ACT contradicts any of these results; Section 10 shows several of them falling out of the theory as instances.

2.4. The Gap ACT Fills

These measurements are points without a coordinate system. Each measures the consumption of one configuration on one workload, parameterized by design choices (agent count, rounds, branching) and by model-dependent token constants, so none of them transfers to a new task or survives a model upgrade. What is missing is what classical complexity supplies for algorithms: a fixed abstract machine, an intrinsic input size, and growth laws connecting them. ACT supplies exactly those three things and then shows that the existing measurements, from the debate scaling exponents to the SWE-bench ordering to the production token multiplier, fall out as instances. Raw token-cost models are the constants of this theory, not the theory.

3. The Framework

3.1. Architecture as Machine

Definition 1 (Architecture)**.** An *architecture* M is a finite graph of roles together with a communication topology and a verification policy. Roles are of three kinds: *workers* that resolve atomic sub-decisions, *orchestrators* that dispatch work and aggregate results, and *verifiers* that check outputs. Two architectures are the same machine when their role graphs are isomorphic and their policies agree.

Four machine families capture the dominant production patterns analyzed in this paper. They are not exhaustive: blackboard systems, market-based coordination, recursive planners, and heterogeneous mixtures are variants or compositions whose signatures are computed by the same methods (Section 9 treats one such constructor explicitly).

- The **single agent** $\mathcal{S}$: one worker resolving every sub-decision in sequence, possibly with a bounded reflective self-check loop.
- The **pipeline** $\mathcal{P}_k$: k specialized stages in a chain; every unit of work passes through every stage in order.
- The **tree** $\mathcal{T}(b, g)$: orchestrators of branching factor b over workers of *granularity* g, where g is the number of atomic sub-decisions one worker resolves.
- The **debate** $\mathcal{D}(N, R)$: N peers who each attempt the whole task and exchange messages over R rounds.

The parameters b, g, k, N, R are *internal* choices of a machine, the analog of an algorithm's loop structure or block size. They are not the input to the complexity function. This demotion is the load-bearing decision of the theory: earlier cost models, including our own, took the number of agents as the independent variable, and the number of agents is a knob a designer turns, not a property of the problem.

3.2. Cost Assumptions

Every quantitative statement in this paper traces back to the following four assumptions. Each is stated with the condition under which it breaks.

Assumption 1 (Elementary work cost)**.** Resolving one atomic sub-decision costs a model-dependent amount $\tau > 0$ of resource, and a worker of granularity g costs $g\tau + c_w$ tokens for a machine constant $c_w \geq 0$ (prompt framing, instructions). The constant τ is discarded under Θ, exactly as a constant factor is discarded in classical time complexity. *Breaks when:* a single sub-decision exceeds a context window, so it is not atomic for the machine at hand.

Assumption 2 (Coordination cost)**.** An orchestrator with b children costs $C_o + b\,s$ tokens: a fixed dispatch and framing part C_o, plus reading one condensed summary of size s from each child. A verifier costs T_v tokens per invocation. *Breaks when:* children return raw transcripts instead of summaries, in which case s grows with the child's whole output.

Assumption 3 (Additivity)**.** The machine's total resource use is the sum over all calls of their costs; no tokens are shared across calls unless explicitly cached. *Breaks when:* aggressive prefix caching makes repeated context reads nearly free.

Assumption 4 (Bounded independent retries)**.** A failing node retries independently with failure probability $p < 1$, capped at k attempts, giving the retry multiplier $\Gamma(p, k) = \frac{1-p^k}{1-p} \leq \frac{1}{1-p} = O(1)$. *Breaks when:* failures are correlated (a systematic prompt defect) or $p \to 1$, where Γ diverges and forces a redesign rather than a retry.

Assumption 1 is the invariance move. A stronger model lowers τ, and lowering a constant leaves a Θ statement unchanged, which is precisely why the resulting complexity is model-independent while a raw token count is not.

3.3. Intrinsic Task Size

We need a measure of difficulty that is a property of the task and does not fall when the model improves. Counting anything the model *produces*, tokens or reasoning steps, fails this test, because a stronger model produces fewer. The measure must describe the problem itself.

Definition 2 (Minimal decision DAG)**.** For a task t, a *decision DAG* is a directed acyclic graph whose nodes are atomic sub-decisions and whose edges are genuine data dependencies, such that any correct solver must resolve every node. The *minimal* decision DAG $D(t)$ is one of least node count. The *work size* is $n_W(t) = |V(D(t))|$ and the *span size* $n_S(t)$ is the number of nodes on a longest directed path of $D(t)$.

Both quantities are intrinsic. They state what any solver must resolve and in what dependency order, and they reference no model, no token, and no price. A stronger model resolves each node more cheaply, lowering τ, but it cannot lower the number of nodes that must be resolved or shorten the dependency chain.

Proposition 1 (Uncomputability)**.** *n_W and n_S are not computable in general.*

We argue informally, in the manner of Kolmogorov complexity [5]. The minimal decision DAG is a least-size object over all correct decompositions of a task, and deciding minimality would decide properties of arbitrary programs that generate the task. We therefore treat n_W, n_S as theoretical objects, define complexity relative to them, and supply a measurable proxy.

Remark 1 (Why an uncomputable variable is still usable). Defining a theory over an uncomputable size is not a defect of ACT; it is the standard construction for intrinsic measures, and it is worth making the lineage explicit. Kolmogorov complexity is uncomputable, yet it anchors algorithmic information theory, and practitioners estimate it every day with compressors, which are one-sided proxies: compressed length upper-bounds description length. The minimum size of a circuit computing a given function is not known to be efficiently computable, yet circuit complexity is a central theory, and explicit constructions provide one-sided upper bounds on it. Sample complexity in learning theory is defined through quantities (such as the capacity of an unknown-to-the-learner concept class) that a practitioner never computes exactly, yet generalization bounds are used daily through estimable surrogates. ACT sits in exactly this position: the minimal decision DAG is the intrinsic object, the annotated DAG of Section 4 is the proxy, and the proxy's bias is one-sided by construction, $\widehat{n}_W \geq n_W$, just as a compressor never beats Kolmogorov. What each of these theories requires of its proxy is not exactness but *consistency*: a proxy applied uniformly across a task family

preserves growth rates even when it over-counts levels, which is why Section 4 fixes one rule set per family and why the scaling laws are tested within families. The practitioner's answer to "why care about an uncomputable variable" is therefore the same as the compression practitioner's: because the computable proxy inherits the theory's predictions, and the predictions are checked against the proxy, never against the ideal.

Definition 3 (Annotated proxy)**.** For a benchmark task with a known sub-question structure, the *annotated decision DAG* $\widehat{D}(t)$ is the dependency graph of its labeled sub-questions, with $\widehat{n}_W$ and $\widehat{n}_S$ its node count and longest path. The proxy over-counts when the stated decomposition is coarser than the true minimal one, so $\widehat{n}_W \geq n_W$, and we use it only to *test* scaling laws on task families whose annotation is consistent across sizes.

3.4. The Complexity Signature

Definition 4 (Agent complexity signature)**.** The *signature* of machine M on task t is the vector

$$\mathcal{A}(t; M) \;=\; \big(A_E,\; A_L,\; A_C,\; A_V\big),$$

whose coordinates are the growth rates, in (n_W, n_S), of total resource (economic), critical-path length (latency), inter-role message count (communication), and verifier invocations (verification).

The signature plays the role that the pair (time, space) plays for algorithms. Architectures separate by signature the way algorithms separate by time and space, and the whole of Section 5 is the computation of Table 1.

4. The Annotation Protocol

Definitions 2 and 3 leave a question a careful reader must ask: who decides what is atomic? Without a fixed answer, $\widehat{n}_W$ can be inflated or deflated at will, and any scaling law tested against it is unfalsifiable. This section fixes the answer. The protocol has five rules, applied to the task statement alone, before any machine is chosen or run.

Terminology, fixed once and used consistently hereafter: a *decision DAG* is any graph satisfying Definition 2; the *minimal decision DAG* $D(t)$ is the least-node one, whose sizes are the intrinsic (n_W, n_S); and the *annotated DAG* $\widehat{D}(t)$ is this protocol's computable proxy, whose sizes are $(\widehat{n}_W, \widehat{n}_S)$. Theory is stated in the intrinsic sizes; everything measured or designed is stated in the annotated ones.

- **R1. Nodes come from the task statement, not from any solver.** A node is a question that the task statement itself requires answering. Anything that references a tool, a model, a prompt, or an agent ("call the search API," "summarize the context") is a machine operation, not a task node, and must not appear in the DAG.
- **R2. A node is atomic if its answer is consumed whole.** Split a candidate decision into two nodes only when some other node consumes one part of its answer without the other. If the parts are always consumed together, they are one node. This rule is the stopping condition of the decomposition and the direct defense against padding $\widehat{n}_W$.
- **R3. Edges are data dependencies, not workflow habits.** Draw $u \to v$ if and only if the question at v cannot be stated, or its answer evaluated, without the answer at u. "Usually done after" is not an edge; "cannot be posed before" is.
- **R4. Merge before counting.** After applying R1 to R3, merge any set of nodes whose answers are consumed only jointly and only by the same successors. The result is the annotated DAG $\widehat{D}(t)$; its node count is $\widehat{n}_W$ and its longest path is $\widehat{n}_S$. By construction $\widehat{n}_W \geq n_W$: annotation can over-count relative to the true minimum, never under-count a correct solver's obligations.
- **R5. One rule set per task family, fixed in advance.** Scaling tests compare sizes within a family (for example, k-hop questions across k). The protocol must be applied uniformly across the family before any machine is run, and reported alongside the results. Cross-family comparisons of raw $\widehat{n}_W$ are not licensed.

Example 1 (Applying the protocol)**.** Task: "Which of the five most populous capitals in Europe has the oldest university?" R1 gives candidate decisions from the statement: identify the five most populous European capitals (one node? five?); for each capital, find its oldest university's founding date; compare. R2 splits the identification into one node,

because the list is consumed whole by the fan-out, and keeps the five date lookups separate, because the comparison consumes each date individually. R3 draws edges from the identification node to each lookup (a lookup cannot be posed without knowing its capital) and from each lookup to the comparison. R4 finds nothing further to merge. The result: $\widehat{n}_W = 1 + 5 + 1 = 7$, $\widehat{n}_S = 3$, intrinsic parallelism $7/3 \approx 2.3$. A k-hop chain question ("the author of the book that inspired the film that won…") annotates, by the same rules, to $\widehat{n}_W = \widehat{n}_S = k$.

Example 2 (A manipulation the protocol rejects)**.** Suppose an annotator, wanting a larger $\widehat{n}_W$, splits "find the founding date of the University of X" into three nodes: open a source, locate the figure, convert the format. R1 rejects all three: they reference solver operations, not task questions. Suppose instead the annotator splits the date into "find the century" and "find the year within the century." R2 rejects the split: no other node consumes the century separately from the year. The protocol does not make $\widehat{n}_W$ unique in every edge case, but it makes it auditable: two annotators applying R1 to R5 to the same family can disagree only where the task statement itself is ambiguous, and Section 15 lists inter-annotator agreement measurement as the immediate next step.

The protocol is what separates $\widehat{n}_W$ from a free parameter. Every annotation used in this paper (the research fan-out of Section 8.1, the coding chain of Section 8.3, the worked examples of Section 11) is an application of these five rules, and each is stated explicitly enough to be re-derived by a reader.

5. Complete Derivations of the Four Signatures

This section derives every entry of Table 1 step by step. We write costs in units where one atomic sub-decision costs τ; constants c_w, C_o, s, T_v are as in Assumptions 1 and 2, and every total carries the retry factor $\Gamma = O(1)$ of Assumption 4, which we suppress after its first appearance since it never changes a growth rate.

5.1. The Single Agent $\mathcal{S}$

The machine is one worker resolving all n_W sub-decisions in sequence.

Economic.

$$A_E(\mathcal{S}) = \underbrace{\left(n_W\,\tau + c_w\right)}_{\text{one worker, all nodes}} \cdot \Gamma = \Theta(n_W). \tag{1}$$

Latency. Every node is resolved by the same sequential context, so the critical path is all of the work:

$$A_L(\mathcal{S}) = n_W\,\tau = \Theta(n_W). \tag{2}$$

Communication and verification. There are no inter-role messages, $A_C(\mathcal{S}) = \Theta(1)$, and a bounded reflective self-check multiplies cost by a constant, $A_V(\mathcal{S}) = \Theta(1)$.

Equation (2) is worth pausing on: classical sequential time complexity, measured in atomic decisions, *is* $A_L(\mathcal{S})$. Big-O time is the latency projection of the trivial machine, and ACT contains it as a special case (Corollary 3).

5.2. The Pipeline $\mathcal{P}_k$

k specialized stages in a chain; each of the n_W sub-decisions is touched by the stage responsible for it, and stage boundaries add one hand-off message per unit of flow.

Economic. The stages partition the work, so summing stage costs gives

$$A_E(\mathcal{P}_k) = \sum_{i=1}^{k} \left(w_i\,\tau + c_w\right) = n_W\,\tau + k\,c_w = \Theta(n_W), \qquad \textstyle\sum_i w_i = n_W. \tag{3}$$

Latency. The stages are sequential by construction: stage $i{+}1$ consumes stage i's output. The critical path is therefore the whole task plus k hand-offs,

$$A_L(\mathcal{P}_k) = n_W\,\tau + \Theta(k) = \Theta(n_W) \quad \text{for fixed } k. \tag{4}$$

A pipeline is work-optimal but buys no latency: it reorganizes the sequence without shortening it. (With a *stream* of many tasks, pipelining raises throughput; ACT statements here are per task.)

Communication and verification. One hand-off per stage boundary per unit of work gives $A_C(\mathcal{P}_k) = \Theta(n_W)$ in the worst case and $\Theta(k)$ when stages exchange only summaries; a per-stage check gives $A_V = \Theta(k) = \Theta(1)$ for fixed k.

5.3. The Tree $\mathcal{T}(b, g)$: Structure, Then the Two Theorems

5.3.1. Structural Counts

To resolve n_W atomic decisions with workers of granularity g, the machine uses

$$L \;=\; \left\lceil \frac{n_W}{g} \right\rceil \quad \text{workers (leaves)}, \qquad h \;=\; \left\lceil \log_b L \right\rceil \quad \text{levels of orchestration.} \tag{5}$$

A complete b-ary tree over L leaves has internal node count

$$N_{\text{orch}} \;=\; \frac{L-1}{b-1}, \tag{6}$$

the standard geometric sum $\sum_{\ell=0}^{h-1} b^\ell$ evaluated at $b^h = L$. The design knobs are now determined by the task size and the machine's internal choices, not free.

5.3.2. Economic Complexity [Full Derivation]

Theorem 1 (Economic complexity of the tree)**.** *For fixed internal choices (b, g),*

$$A_E\big(\mathcal{T}(b,g)\big) \;=\; \kappa(b,g)\, n_W \;+\; O(1) \;=\; \Theta(n_W), \qquad \kappa(b,g) \;=\; \Big[\tau + \frac{c_w}{g} + \frac{C_o + bs + T_v}{g\,(b-1)}\Big]\,\Gamma,$$

and no architecture has economic complexity below $\Theta(n_W)$, so the tree is work-optimal.

Proof. We compute the three role totals and add them.

Step 1 (workers). Each of the L workers resolves g atomic decisions at cost $g\tau + c_w$ (Assumption 1):

$$\text{Workers} \;=\; L\,(g\tau + c_w) \;=\; \left\lceil \tfrac{n_W}{g} \right\rceil (g\tau + c_w) \;=\; n_W\,\tau \;+\; \frac{c_w}{g}\, n_W \;+\; O(1),$$

where the $O(1)$ absorbs the ceiling.

Step 2 (orchestrators). Each of the N_{orch} internal nodes costs $C_o + bs$ (Assumption 2); by (6),

$$\text{Orchestration} \;=\; \frac{L-1}{b-1}\,(C_o + bs) \;=\; \frac{n_W}{g\,(b-1)}\,(C_o + bs) \;+\; O(1).$$

Step 3 (verifiers). With the default policy of one verifier per orchestrator,

$$\text{Verification} \;=\; \frac{L-1}{b-1}\,T_v \;=\; \frac{n_W}{g\,(b-1)}\,T_v \;+\; O(1).$$

Step 4 (retries and total). Summing Steps 1 to 3 and applying $\Gamma = O(1)$ (Assumption 4),

$$A_E \;=\; n_W \Big[\tau + \frac{c_w}{g} + \frac{C_o + bs + T_v}{g\,(b-1)}\Big]\,\Gamma \;+\; O(1) \;=\; \kappa(b,g)\, n_W + O(1).$$

The bracket is a positive constant in n_W, so $A_E = \Theta(n_W)$.

Step 5 (lower bound). Any correct solver must resolve all n_W nodes of $D(t)$ (Definition 2), and each resolution costs at least $\tau > 0$ (Assumption 1), so every architecture obeys $A_E \geq \tau\, n_W = \Omega(n_W)$. The tree meets this bound. □

Engineering interpretation. A well-shaped tree spends what the task costs plus a bounded coordination premium per unit of work; no architecture can spend asymptotically less, so the budget conversation is about the premium κ/τ, never about the growth rate.

Corollary 1 (Overhead law: width beats depth)**.** *The coordination overhead of the tree relative to raw worker cost is*

$$\frac{\textit{Orchestration} + \textit{Verification}}{\textit{Workers}} \;=\; \frac{C_o + bs + T_v}{g\,(b-1)\,\tau} \;+\; o(1),$$

which is decreasing in b. Among tree machines, the widest tree whose orchestrator context can hold b child summaries (i.e., $b\,s \le$ context budget) minimizes κ. Depth enters only through $h = \lceil \log_b(n_W/g) \rceil$, inside the machine, where it belongs.

This is the design law practitioners rediscover empirically: Anthropic reports a flat lead-plus-subagents shape and reports that an early version which spawned dozens of subagents overran its coordinator's context [6], which is precisely the $b\,s \le$ context constraint binding.

5.3.3. Latency Complexity [Full Derivation]

Theorem 2 (Latency complexity of the tree)**.** $A_L\big(\mathcal{T}(b,g)\big) \;=\; \Theta\big(n_S + \log n_W\big)$*, while every architecture obeys $A_L \ge \Omega(n_S)$. The tree is therefore within an additive $\Theta(\log n_W)$ of the intrinsic latency bound.*

Proof. The makespan is the length of the longest chain of dependent operations. Two chains compose it.

Step 1 (the task's own chain). The longest directed path of $D(t)$ has n_S nodes, and a node cannot be resolved before its predecessors, so any schedule spends at least n_S sequential units of τ on it: contribution $\Theta(n_S)$.

Step 2 (the machine's coordination depth). A dispatch path travels down h levels and an aggregation path back up h levels, each hop costing $O(1)$ coordination units:

$$\Theta(h) \;=\; \Theta\big(\lceil \log_b(n_W/g) \rceil\big) \;=\; \Theta(\log n_W) \quad \text{for fixed } b, g.$$

Step 3 (composition). Independent nodes at the same DAG depth run concurrently on distinct workers, so no other chain exceeds these two. Summing,

$$A_L = \Theta(n_S) + \Theta(\log n_W) = \Theta(n_S + \log n_W).$$

Step 4 (lower bound). Any solver, on any architecture, must traverse the task's dependency chain of length n_S (Step 1 applies universally), so $A_L \ge \Omega(n_S)$. □

Engineering interpretation. Tree architectures reduce latency only when the task contains intrinsic parallelism; on a task that is mostly a chain, they add coordination overhead without asymptotic benefit, and the only remaining lever is a faster model.

Corollary 2 (Parallel benefit is intrinsic)**.** *The tree improves latency over the single agent from $\Theta(n_W)$ to $\Theta(n_S + \log n_W)$. The improvement factor is*

$$\frac{A_L(\mathcal{S})}{A_L(\mathcal{T})} \;=\; \Theta\Big(\frac{n_W}{n_S + \log n_W}\Big),$$

which is large exactly when $n_W \gg n_S$ (the task is intrinsically parallel) and collapses to $\Theta(1)$ when $n_S = \Theta(n_W)$ (the task is an intrinsic chain). The maximal useful degree of parallelism is $\Theta(n_W/n_S)$: this is Brent's bound [2] stated in task-intrinsic terms rather than in agent counts, and deploying more than $\sim n_W/n_S$ workers adds cost without reducing latency.

Corollary 3 (Classical time is a projection)**.** $A_L(\mathcal{S}) = \Theta(n_W)$ *is sequential time complexity measured in atomic decisions. ACT therefore contains classical Big-O time as the latency coordinate of the trivial machine, and generalizes it by admitting machines whose latency falls below total work when the task permits.*

5.4. The Debate Machine $\mathcal{D}(N, R)$

N peers each attempt the whole task, then exchange and revise over R rounds. Debate buys accuracy [19]; ACT prices what it costs.

Economic.

Step 1 (initial solves). Each peer resolves all n_W nodes: $N\,(n_W\tau + c_w)$.

Table 1: ACT complexity signatures over the same intrinsic size (n_W, n_S), as derived in Sections 5.1 to 5.4. Debate is shown with its internal choices (N, R) displayed, since the empirical literature varies them; for fixed internal choices every row is $\Theta(n_W)$ in economics, and the machines separate on latency and communication.

Machine	A_E	A_L	A_C	character
Single agent $\mathcal{S}$	$\Theta(n_W)$	$\Theta(n_W)$	$\Theta(1)$	work-optimal, no parallel benefit
Pipeline $\mathcal{P}_k$	$\Theta(n_W)$	$\Theta(n_W)$	$\Theta(k)$–$\Theta(n_W)$	sequential by construction
Tree $\mathcal{T}(b, g)$	$\Theta(n_W)$	$\Theta(n_S + \log n_W)$	$\Theta(n_W)$	work-optimal, near latency-optimal
Debate $\mathcal{D}(N, R)$	$\Theta(Nn_W + N^2R)$	$\Theta(n_S + R)$	$\Theta(N^2R)$	redundant work, quadratic messages

Step 2 (exchange). In round r, each peer reads the other $N-1$ peers' messages of size s and produces a revision of size u: per round $N\big[(N-1)s + u\big]$, and over R rounds $N(N-1)\,s\,R + NuR$.

Step 3 (total).

$$A_E(\mathcal{D}) \;=\; N\,n_W\,\tau \;+\; N(N-1)\,s\,R \;+\; O(NR) \;=\; \Theta\big(N\,n_W + N^2R\big). \tag{7}$$

For fixed internal (N, R) this is $\Theta(n_W)$ with a redundancy constant N; we display the internal parameters in Table 1 because they are what the empirical literature varies.

Latency, communication, verification. Rounds are sequential: $A_L(\mathcal{D}) = \Theta(n_S + R)$. Messages: $N(N-1)$ per round, so $A_C(\mathcal{D}) = \Theta(N^2R)$. Verification is mutual criticism, implicit in the exchange: $A_V = \Theta(NR)$.

Remark 2 (The accumulation correction, learned from a failure). Step 2 assumes each round costs the same. Measured debate systems violate this: each round re-reads the accumulated transcript of all prior rounds, so round r costs $\Theta(N^2 s\,r)$ and the exchange total becomes $N(N-1)\,s\,\frac{R(R+1)}{2} = \Theta(N^2R^2)$. We flag this here because Section 10 shows the constant-per-round model *failing* against measured round scaling, exponent 1 predicted against 2.0 to 2.4 observed, while the accumulation form predicts exponent 2. The correction was identified from the failure, not assumed in advance.

5.5. The Signature Table

Table 1 collects the four derived signatures over one intrinsic size. Read as a whole, it is the point of the theory: over the same (n_W, n_S), the tree is work-optimal and near latency-optimal, the single agent is work-optimal but latency-bound to total work, the pipeline reorganizes without accelerating, and debate pays a factor of N in work and a quadratic in communication for whatever accuracy it buys. This is a genuine separation of architectures by intrinsic complexity, in the same sense that merge sort and insertion sort separate by time.

6. A Composition Calculus for Role Graphs

The four families of Section 5 are templates, and production systems increasingly wire them together: a planner over two trees, a tree whose output feeds a pipeline, a debate used as the verifier of a hierarchy. A theory that only covers templates would be obsolete on contact with the first hybrid. This section closes that gap: signatures compose, and the composition rules are the same five that structured programming uses. Any role graph built from workers by these five operators has a computable signature, and each of the four families is itself a short composition.

6.1. The Five Operators

Let M_1, M_2 be role graphs with signatures $(A_E^{(i)}, A_L^{(i)}, A_C^{(i)}, A_V^{(i)})$ on their respective sub-DAGs.

C1. Series $(M_1\,;\,M_2)$: M_2 consumes M_1's output. Work, latency, and messages add:

$$A_E = A_E^{(1)} + A_E^{(2)}, \qquad A_L = A_L^{(1)} + A_L^{(2)} + O(1), \qquad A_C = A_C^{(1)} + A_C^{(2)} + O(1).$$

C2. Parallel $(M_1 \parallel \cdots \parallel M_b$, with or without a join): the branches run concurrently on disjoint sub-DAGs. Work and messages add; latency is the slowest branch:

$$A_E = \sum_i A_E^{(i)}\ [+\,C_o + bs \text{ if joined}], \qquad A_L = \max_i A_L^{(i)}\ [+\,O(1)], \qquad A_C = \sum_i A_C^{(i)}\ [+\,b].$$

The bracketed join terms are the orchestration cost, and they appear *only if the DAG contains a join node*. A fan-out whose branches deliver independent outputs (a sharded queue, Design Study 6) pays no join term, which is why replication beats orchestration there.

C3. Hierarchy ($M_1[M_2]$): M_2 is substituted for a worker slot of M_1. Substitute the inner signature for the worker term in the outer derivation. This rule is how the deep tree arises from the flat one: $\mathcal{T}(b, g)$ at depth h is the flat star composed with itself h times.

C4. Feedback (loop M_1 up to k times, exit on success with failure probability p): every coordinate is multiplied by the retry factor $\Gamma(p, k) = \frac{1-p^k}{1-p} = O(1)$ of Assumption 4. Bounded feedback never changes a growth rate; unbounded feedback with $p \to 1$ diverges, which is the formal reason retry budgets must be finite.

C5. Conditional (run M_1 with probability π, else M_2): expected coordinates are the π-mixture; worst-case latency is the max. The adaptive orchestrator of Section 9 is precisely a conditional over an estimated task property.

Proposition 2 (Compositionality)**.** *The signature of any role graph built from workers by C1 to C5 is computable bottom-up by one application of the matching rule per operator, in time linear in the size of the graph's syntax tree. The four families are the compositions: $\mathcal{S}$ is a worker under C4; $\mathcal{P}_k$ is C1 applied $k-1$ times; $\mathcal{T}(b, g)$ is C2-with-join under C3 to depth h; $\mathcal{D}(N, R)$ is C2-without-join under C4 for R rounds, plus the all-to-all message term counted by C2's sum.*

The proof is structural induction with nothing surprising: each rule adds, maximizes, substitutes, or scales, and Θ-arithmetic is closed under all four. What the proposition buys is future-proofing: a reviewer's blackboard system, market mechanism, or recursive planner is a composition, and its signature is a calculation, not a new theory.

Algebraic properties. The operators carry the algebra one would hope for, and one distinction worth stating. Signatures are *closed* under C1 to C5 (that is Proposition 2). Series is *associative*, $(M_1; M_2); M_3 = M_1; (M_2; M_3)$ in signature, since sums and $O(1)$ hand-offs reassociate. Parallel is associative and *commutative* in signature, since sums and maxima are. The *identity* is the empty machine (zero on every coordinate): a unit for both series and parallel. What does *not* hold is distributivity of series over parallel in general: $(M_1 \parallel M_2); M_3$ and $(M_1; M_3) \parallel (M_2; M_3)$ differ in both work (the right side runs M_3 twice) and join structure, and this failure is a feature, not a defect, because it is exactly the design question of where to place a shared downstream stage, priced rather than hidden. We do not claim a normal form; finding conditions under which every role graph reduces to a canonical composition is an open question we leave stated rather than half-answered.

6.2. A Worked Composite

Consider the hybrid a platform team actually proposed to us: a planner dispatching *two* trees in parallel (one over internal documents, one over the public web), their outputs joined and passed through a short verification pipeline. In operator form:

$$M \;=\; \Big(\mathcal{T}_1(b, g) \;\parallel\; \mathcal{T}_2(b, g)\Big)_{\text{join}} ; \mathcal{P}_2.$$

Let the two trees carry work $n_W^{(1)}$ and $n_W^{(2)}$ with spans $n_S^{(1)}, n_S^{(2)}$, and let the verification pipeline touch the joined summary, $O(1)$ nodes. Applying C2 then C1:

$$\begin{aligned}
A_E &= \Theta\big(n_W^{(1)}\big) + \Theta\big(n_W^{(2)}\big) + (C_o + 2s) + O(1) \;=\; \Theta\big(n_W^{(1)} + n_W^{(2)}\big), \\
A_L &= \max\Big(\Theta\big(n_S^{(1)} + \log n_W^{(1)}\big),\; \Theta\big(n_S^{(2)} + \log n_W^{(2)}\big)\Big) + O(1) \;=\; \Theta\big(\max_i n_S^{(i)} + \log \max_i n_W^{(i)}\big), \\
A_C &= \Theta\big(n_W^{(1)} + n_W^{(2)}\big) + O(1).
\end{aligned}$$

The composite inherits work-optimality from its parts and latency set by its slowest branch, which is exactly the design intuition ("the web tree will be the bottleneck") expressed as arithmetic. No new derivation was needed, and none will be for the next hybrid either.

7. Four Falsifiable Predictions

Before consulting any production data, we derive the predictions from the theorems by substitution, keeping the same discipline as Section 5: equation, substitution, simplification, prediction. Each names the evidence that would falsify it.

Prediction 1 (Parallel Advantage Tracks n_W/n_S). Start from the two latency laws (Theorem 2, Equation (2)) and form the gain:

$$G \;=\; \frac{A_L(\mathcal{S})}{A_L(\mathcal{T})} \;=\; \frac{\Theta(n_W)}{\Theta(n_S + \log n_W)}.$$

Substitute the two regimes:

$$n_S \ll n_W: \;\; G = \Theta\Big(\frac{n_W}{\log n_W}\Big) \gg 1, \qquad\qquad n_S = \Theta(n_W): \;\; G = \Theta\Big(\frac{n_W}{n_W}\Big) = \Theta(1).$$

Trees should therefore show large gains on breadth-first tasks and no gain on chain tasks. *Falsified by:* a tree decisively beating a comparable single agent on an intrinsically sequential task class, or failing to help on an intrinsically parallel one.

Prediction 2 (On Sequential Tasks, Simpler Is Better). Form the cost ratio from Theorem 1 and Equation (1):

$$\frac{A_E(\mathcal{T})}{A_E(\mathcal{S})} \;=\; \frac{\kappa(b,g)}{\tau} \;=\; 1 \;+\; \frac{c_w}{g\,\tau} \;+\; \frac{C_o + bs + T_v}{g\,(b-1)\,\tau} \;>\; 1,$$

while at $n_S = \Theta(n_W)$ Prediction 1 gives latency gain $G = \Theta(1)$. The hierarchy pays a strictly positive premium for a gain of zero, so a minimal sequential agent should match or beat it at comparable model quality. *Falsified by:* hierarchical coding agents systematically dominating minimal ones on SWE-bench-class tasks with comparable models.

Prediction 3 (Debate Cost Scaling). Write Equation (7) as $A_E(N) = c_1 N + c_2 N^2$ with $c_1, c_2 > 0$ and compute the growth exponent:

$$e(N) \;=\; \frac{d\ln A_E}{d\ln N} \;=\; \frac{c_1 N + 2\,c_2 N^2}{c_1 N + c_2 N^2} \;\in\; (1,2), \qquad e(N) \to 2 \text{ as } c_2 N \gg c_1.$$

In rounds, the constant-per-round form gives $A_E \propto R$, exponent 1; the accumulation form of Remark 2 gives $A_E \propto R(R{+}1)/2$, exponent $\to 2$. Measured exponents in N must fall in $(1,2)$; measured exponents in R discriminate between the two forms. *Falsified by:* measured exponents outside these bands.

Prediction 4 (Production Trees Are Wide and Flat). Differentiate the overhead of Corollary 1 in the branching factor:

$$\Omega(b) \;=\; \frac{C_o + bs + T_v}{g\,(b-1)\,\tau} \qquad\Longrightarrow\qquad \frac{\partial\Omega}{\partial b} \;=\; \frac{-\,(C_o + T_v + s)}{g\,(b-1)^2\,\tau} \;<\; 0,$$

so overhead is strictly decreasing in b and the rational design is the widest branching an orchestrator's context allows ($bs \le$ context budget), adding depth only under context pressure. *Falsified by:* production systems preferring deep narrow trees where a flat shape fits in context.

8. Case Studies I: Four Production Systems [Full]

We now confront the predictions with ten representative production agentic workflows spanning the dominant architectural families. A note on epistemic role: these case studies are *demonstrations*, showing that the machinery applies cleanly to real systems and that each system's reported behavior sits where its signature says it should; the *validation*, with its explicit tests, residuals, and failure, is Section 10. The claims that follow attach to the families, not to the particular systems that instantiate them. This section treats four flagship systems, one from each architectural family with published outcome numbers, as complete mathematical case studies. The four case studies do not carry equal evidentiary weight, and each verdict states its evidence level explicitly: *primary* where quantitative public numbers

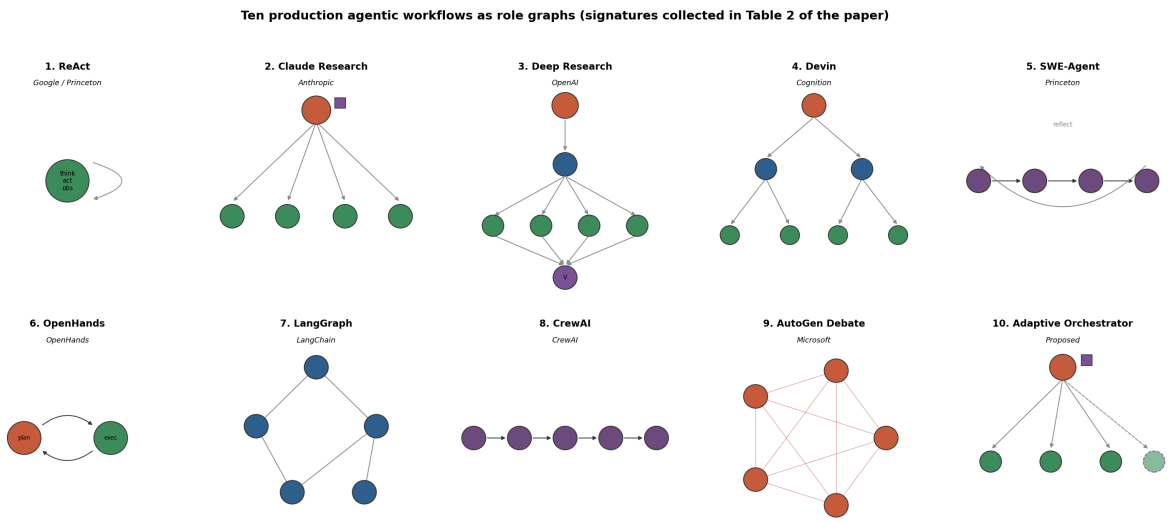

Figure 1: Ten production agentic workflows as role graphs. Orchestrator-worker and hierarchical systems are trees; coding agents are linear or reflective; debate is a complete graph. Each system's signature is collected in Table 2; the signature is a property of the architecture and holds for any task.

exist, *supporting* where only ordinal or partially public evidence does. Section 9 treats the remaining six compactly. Figure 1 draws all ten as role graphs. Every case study follows the same fixed protocol, so that nothing is fit after the fact:

S1. *Public architecture:* what the vendor or authors describe, with citation.

S2. *ACT machine:* the role graph it corresponds to in Definition 1.

S3. *Task annotation:* the intrinsic structure $(\widehat{n}_W, \widehat{n}_S)$ of the system's task class, from the task's own dependency structure. Annotations describe the task class, not any tuned quantity.

S4. *Signature:* the four coordinates, computed from the theorems of Section 5.

S5. *Prediction:* what the signature implies, stated before the data.

S6. *Reported result:* the published numbers.

S7. *Residual:* where prediction and observation agree, where they do not, and every caveat.

8.1. Claude Research: A Flat Tree on an Intrinsically Parallel Task

S1. Public architecture

Anthropic describes a lead researcher agent that plans, spawns 3 to 5 subagents that search in parallel, and synthesizes their condensed summaries, with a citation-checking pass at the end [6]. The report states that the system excels on breadth-first queries that pursue multiple independent directions simultaneously, and that an early version which spawned dozens of subagents overran the coordinator's context.

S2. ACT machine

This is the tree $\mathcal{T}(b, g)$ with $b \in [3, 5]$, one orchestration level ($h = 1$), and a verifier at the root: the shape Corollary 1 says a rational designer should pick, the widest tree whose child summaries fit the coordinator's context.

S3. Task annotation

A breadth-first research query decomposes as: 1 planning decision, q independent sub-investigations, 1 synthesis decision, with q typically 3 to 20. The dependency graph is plan $\rightarrow$ (q parallel branches) $\rightarrow$ synthesize, so

$$\widehat{n}_W = q + 2, \qquad \widehat{n}_S = 3, \qquad \frac{\widehat{n}_W}{\widehat{n}_S} = \frac{q+2}{3} \gg 1 \text{ for large } q.$$

This is the regime $n_S \ll n_W$ of Prediction 1.

S4. Signature

By Theorem 1, with $L = \lceil (q+2)/g \rceil$ workers and one orchestration level,

$$A_E = \kappa(b, g)\, \widehat{n}_W = \Theta(n_W) \quad \text{with} \quad \kappa = \Big[\tau + \frac{c_w}{g} + \frac{C_o + bs + T_v}{g(b-1)}\Big]\Gamma.$$

By Theorem 2 with $h = 1$,

$$A_L = \Theta(\widehat{n}_S + h) = \Theta(3+1) = \Theta(1) \text{ in } q, \qquad \text{versus} \qquad A_L(\mathcal{S}) = \Theta(q+2).$$

Communication is one dispatch and one summary per subagent, $A_C = \Theta(q)$; verification is one citation pass, $A_V = \Theta(1)$.

S5. Prediction

Three concrete statements, all made by the theory before looking at outcomes. (a) Latency and outcome gains over a single agent should be large and should grow with the query's breadth q (Prediction 1). (b) Token cost should exceed a single agent's by roughly the worker count plus coordination overhead: for $L \in [3, 5]$ workers each comparable to a standalone agent, the multiplier band is

$$\frac{A_E(\mathcal{T})}{A_E(\mathcal{S})} \;\in\; \Big[\, L_{\min}\,(1{+}\,\text{overhead}),\; L_{\max}\,(1{+}\,\text{overhead}) \Big] \;\approx\; [\,3.3,\; 5.6\,]$$

taking coordination overhead between 10% and 40% of worker cost from Corollary 1 at $b \in [3, 5]$. (c) The production shape should be wide and flat (Prediction 4).

S6. Reported result

Anthropic reports: the multi-agent system outperforms a single-agent baseline by 90.2% on the internal research evaluation; research time is cut by up to 90% on suitable queries; the system uses about $15\times$ the tokens of a chat while a single agent uses about $4\times$, a multi-to-single ratio of $15/4 = 3.75$; and the deployed shape is a flat lead with 3 to 5 subagents [6].

S7. Residual

(a) Large gains on breadth-first queries: observed, consistent with $\widehat{n}_W/\widehat{n}_S \gg 1$. (b) The measured multiplier 3.75 lies inside the predicted band $[3.3, 5.6]$; the band was computed from the reported subagent count, not tuned to the ratio. (c) The deployed topology is the width-first shape the overhead law selects, including the documented context-overflow failure when width was pushed too far. *Caveats:* the 90.2% figure is an answer-quality metric on a private evaluation, and ACT prices resources, not quality; the agreement claimed is that the resource trade-off (more tokens, less time) landed where the signature says it must.

Verdict. Evidence level: *primary*. Three independent checks pass: gain regime (qualitative), token multiplier (quantitative, inside a pre-stated band), topology (structural). No disagreement observed.

8.2. OpenAI Deep Research: A Hierarchy on the Same Task Class [Reduced Evidence]

S1. Public architecture

OpenAI describes Deep Research as an agent that autonomously decomposes a research query, browses and analyzes many sources over tens of minutes, and synthesizes a cited report [7]. Public descriptions indicate parallel sub-investigation under a coordinating process; fine internal detail is not published.

S2. ACT machine

A tree $\mathcal{T}(b, g)$, possibly with $h \geq 2$ under context pressure. The graph is the publicly described structure, not a certified one.

S3. Task annotation

Identical task class to Section 8.1: $\widehat{n}_W = q + 2$, $\widehat{n}_S = 3$ with large q. GAIA in particular poses multi-source questions whose sub-lookups are independent, the canonical $n_S \ll n_W$ regime [8].

S4. Signature

As in Section 8.1: $A_E = \Theta(n_W)$, $A_L = \Theta(n_S + \log n_W)$, $A_C = \Theta(n_W)$, $A_V = \Theta(1)$ with a final synthesis check.

S5. Prediction

On GAIA-class parallel tasks a tree should beat sequential single-model baselines, at a token cost linear in the number of sub-investigations (Prediction 1); the comparison ACT licenses is ordinal, tree above sequential, because absolute accuracy depends on model quality, which ACT does not price.

S6. Reported result

Deep Research reported state of the art on GAIA (67.36% pass@1, 72.57% cons@64) and 26.6% on Humanity's Last Exam, in both cases far above the sequential single-model baselines available at release [7, 9].

S7. Residual

The ordinal prediction holds: the parallel machine tops the leaderboard on the parallel task class. *Caveats:* model quality and architecture are conflated in a single score, the internal graph is partially private, and no token figures are published, so only the ordinal check is available here. This case study is deliberately weaker evidence than Section 8.1 and we weight it accordingly in Section 10.

Verdict. Evidence level: *supporting*. This case study is intentionally weaker than Section 8.1 because the architecture is only partially public and no token figures are published; the ordinal check passes, and no quantitative check is possible from public data. We include it because the task class matches and the direction of the result is independent of the missing detail.

8.3. mini-SWE-Agent: The Trivial Machine on an Intrinsically Sequential Task

S1. Public architecture

mini-SWE-Agent is a deliberately minimal coding agent, about a hundred lines of control logic: one model, a linear unmodified history, one bash tool, no orchestration, no subagents [11]. It is the purest production instance of the single-agent machine. Its parent, SWE-agent, adds a structured agent-computer interface but keeps the linear loop [12].

S2. ACT machine

Exactly $\mathcal{S}$: one worker, sequential context, bounded self-checking.

S3. Task annotation

A SWE-bench issue resolves as a chain: reproduce the bug $\to$ localize $\to$ edit $\to$ run tests $\to$ interpret failure $\to$ edit again $\to \cdots \to$ tests pass. Each step consumes the previous step's output, so the decision DAG is nearly a path:

$$\widehat{n}_S \approx \widehat{n}_W, \qquad \frac{\widehat{n}_W}{\widehat{n}_S} \approx 1.$$

This is the regime of Prediction 2.

S4. Signature

From Section 5.1: $A_E = \Theta(n_W)$ at the work-optimal constant τ per node (no κ overhead), $A_L = \Theta(n_W)$, $A_C = \Theta(1)$, $A_V = \Theta(1)$.

S5. Prediction

By Theorem 2, when $n_S \approx n_W$ *every* machine has latency $\Theta(n_W)$, so parallel structure buys nothing; by Theorem 1 any coordination pays $\kappa > \tau$ per node for that nothing. The minimal machine should therefore be competitive with or better than any hierarchy at comparable model quality.

S6. Reported result

mini-SWE-Agent scores above 74% on SWE-bench Verified (75.4% in its best published configuration with Claude-family models), placing it at or near the top of the public comparisons [11, 10].

S7. Residual

The trivial machine is the strongest public performer on the sequential task class, exactly the direction Prediction 2 points. *Caveats:* SWE-bench scores conflate scaffold with model, and mini-SWE-Agent's scores use frontier models; the clean claim is comparative and within comparable model families, and it is made precise in the ranking test of Section 10.

Verdict. Evidence level: *primary*. Agreement. The result that a hundred-line agent tops far heavier architectures stops being a surprise and becomes the predicted consequence of $n_S \approx n_W$.

8.4. Devin: A Hierarchy Paying κ on a Task Where Trees Cannot Win [Confounded Evidence]

S1. Public architecture

Devin is described publicly as an autonomous software engineer with long-horizon planning, its own shell, editor, and browser, and internal task decomposition and re-planning [13]. Internals are proprietary; public descriptions and third-party analyses consistently describe a planner-over-executors hierarchy.

S2. ACT machine

A tree with $h \geq 1$ over the same coding task class, taken from the public description; the graph is not certified.

S3. Task annotation

The same class as Section 8.3: $\widehat{n}_S \approx \widehat{n}_W$.

S4. Signature

By Theorems 1 and 2 evaluated on a chain-like DAG:

$$A_L(\mathcal{T}) = \Theta(n_S + \log n_W) = \Theta(n_W) \quad \text{(no better than } \mathcal{S}\text{)}, \qquad A_E(\mathcal{T}) = \kappa\, n_W > \tau\, n_W = A_E(\mathcal{S}).$$

The hierarchy pays the coordination constant and cannot collect the latency dividend, because the task has no parallel slack to exploit.

S5. Prediction

On SWE-bench-class tasks the hierarchy should not outperform minimal machines, and each orchestration layer adds a coordination-error surface (mis-scoped subtasks, lossy summaries) with no compensating latency gain: the complex graph is not justified on this task class.

S6. Reported result

Devin reports 45.8% on SWE-bench Verified in standard unassisted evaluation [10, 13], against mini-SWE-Agent's 74+% and OpenHands' 72%.

S7. Residual

The observed ordering, minimal linear above planner-executor above hierarchy, is the ordering the signature predicts for a sequential task class. *Caveats, stated fully:* Devin's models are proprietary and possibly weaker than the frontier models under the minimal scaffolds; its score is older than its competitors' latest; and its internals are private. Any

one of these could account for part of the gap. What survives the caveats is the direction: nothing in the public record shows hierarchical coding agents dominating minimal ones, which is what falsification of Prediction 2 would require.

Verdict. Evidence level: *supporting, confounded*. Directional agreement with explicit confounds (private models, private internals, older evaluation date). Counted as one ranking check, not as an isolated accuracy claim.

9. Case Studies II: Six Supporting Systems [Compact]

The remaining six systems apply the same protocol with the derivations referenced rather than repeated. Table 2 collects all ten.

9.1. ReAct

The original reason-act loop [15] is $\mathcal{S}$ with tool calls: signature $(\Theta(n_W), \Theta(n_W), \Theta(1), \Theta(1))$ by Section 5.1. ACT adds the design statement: ReAct is work-optimal and is the right machine whenever $n_W/n_S \approx 1$ or n_W is small enough that coordination overhead cannot amortize (Corollary 1).

9.2. OpenHands

OpenHands runs an event-stream agent with a planner and an executor over a sandbox [14]: a two-role, nearly linear machine, latency $\Theta(n_W)$, with a small constant coordination overhead over $\mathcal{S}$. Its reported 72% on SWE-bench Verified [10] sits, as the signature suggests it should, with the minimal machines and above the deep hierarchy on the sequential class.

9.3. CrewAI

CrewAI's default process is a role pipeline, agents executing in sequence with hand-offs [18]: the machine $\mathcal{P}_k$ of Section 5.2, signature $(\Theta(n_W), \Theta(n_W), \Theta(k), \Theta(k))$. ACT's statement: role specialization can raise quality, but the pipeline shape cannot reduce latency; teams choosing it for speed on a single task are buying the wrong coordinate.

9.4. LangGraph

LangGraph is not one machine but a machine *constructor*: it lets a developer wire an arbitrary role DAG [17]. ACT is, precisely, the type system for that wiring: given the drawn graph, its signature is computed by the methods of Section 5 (workers cost by Assumption 1, fan-in nodes by Assumption 2, longest path for A_L). A LangGraph app that encodes a chain inherits the pipeline row of Table 1; one that encodes a star inherits the tree row.

9.5. AutoGen Debate

AutoGen's group-chat configuration [16] instantiates $\mathcal{D}(N, R)$, whose signature is the debate row of Table 1 as derived in Section 5.4. This is the one family with published per-configuration token measurements along both internal axes [20], which makes it the quantitative anchor of Section 10: the measured growth exponents in N (1.72, 1.82, 1.87 across three benchmarks) fall inside the predicted $(1, 2)$ band, and the measured growth in R falsifies the constant-per-round form exactly as Remark 2 records.

9.6. Adaptive Orchestrator [Proposed]

The tenth graph in Figure 1 is not deployed; it is one architecture naturally suggested by ACT, and it deserves a fuller treatment than the other nine because it is the only one that uses the theory *at runtime*. The machine runs a five-step loop:

A1. *Estimate.* Sketch the task's decision DAG from the request (a cheap, single-call application of the protocol of Section 4) and read off $\widetilde{P} = \widetilde{n}_W/\widetilde{n}_S$, an estimate of the intrinsic parallelism.

A2. *Choose.* Apply the selection rule of Table 5: $\mathcal{S}$ if $\widetilde{P} \approx 1$, replicas if the sketch has no join, $\mathcal{T}$ at width $\min(\widetilde{P}, b_{\max})$ otherwise.

A3. *Run* the chosen machine on the task's frontier: the sub-decisions currently resolvable.

Table 2: All ten systems: machine, signature, task class, prediction, and reported outcome. Full derivations for the first four are in Section 8; the notation $(\cdot, \cdot, \cdot, \cdot)$ is (A_E, A_L, A_C, A_V). Reported numbers are cited in the corresponding case study.

System	Machine	Signature	Task class	Reported	Agrees
Claude Research	$\mathcal{T}(3..5, g), h{=}1$	tree row of Table 1, $A_V{=}\Theta(1)$	research, $n_S \ll n_W$	$+90.2\%$ vs single, time -90%, $3.75\times$ tokens	yes
OpenAI Deep Research	$\mathcal{T}(b, g)$	tree row	research, $n_S \ll n_W$	GAIA SOTA, HLE 26.6%	yes (ordinal)
mini-SWE-Agent	$\mathcal{S}$	single-agent row	coding, $n_S \approx n_W$	74+% SWE-bench Verified	yes
Devin	$\mathcal{T}, h \geq 1$	tree row at $n_S \approx n_W$	coding	45.8% SWE-bench Verified	yes (directional)
ReAct	$\mathcal{S}$	single-agent row	general	baseline loop	yes
SWE-agent	$\mathcal{S}$ + ACI	single-agent row	coding	precursor of mini	yes
OpenHands	planner + executor	near-single, const. overhead	coding	72% SWE-bench Verified	yes
CrewAI	$\mathcal{P}_k$	pipeline row	workflows	sequential by design	consistent
LangGraph	constructor	per drawn graph	any	framework	n/a
AutoGen debate	$\mathcal{D}(N, R)$	debate row	reasoning	exponents 1.72–1.87 in N; super-linear in R	yes / fail-then-fix

A4. *Re-estimate.* As results return, the true DAG reveals itself; recompute $\widetilde{P}$ on the remaining frontier.

A5. *Switch* when the estimate crosses a threshold: collapse the tree to a single agent when the remaining work is a chain (the synthesis tail of a research task), or fan out when a chain step uncovers independent branches (a bug that turns out to be five bugs).

Its signature is the pointwise minimum of the rows it switches between, plus one estimation step per switch, which is $O(1)$ per phase and never changes a growth rate. The interesting quantity is the cost of *misestimation*, and ACT bounds it.

Proposition 3 (Misestimation cost of the adaptive machine)**.** *Suppose the estimator misclassifies the task's regime for a phase covering $m \leq n_W$ atomic decisions. If it chooses $\mathcal{T}$ when the phase is a chain, the excess cost is at most the coordination premium $(\kappa{-}\tau)\, m$ with no latency loss (Theorem 1). If it chooses $\mathcal{S}$ when the phase is parallel, the excess latency is at most $m - (m_S + \log m)$ units of τ, where m_S is the phase's span, with no economic loss (Theorem 2). Neither error affects a growth rate, both are bounded by the phase they occur in, and both are observable in the running totals, which is what step A4 corrects.*

The proof is by reading the two theorems at the misclassified phase's size and subtracting; the content is the design consequence, that adaptive selection is *safe*: its worst case is one phase of the premium it was trying to avoid, not a blowup. The estimator itself is a conditional composition (rule C5), so the whole machine has a computable signature by Proposition 2. Building this machine and testing whether it matches the best fixed architecture per task class without being told the class is one of the concrete programs of Section 15.

10. Reconciliation by Task Class

A single global ranking across all ten systems would mix incomparable numbers, because the systems run different benchmarks with different models. The comparisons ACT licenses are *within* a task class, where the intrinsic structure is shared, plus the fit-free quantitative tests that individual sources make possible. The reconciliation is organized by the kind of evidence each test provides: ordinal validation (Section 10.1), quantitative validation (Section 10.2), and failure analysis (Section 10.3). No test involves a fitted parameter; every prediction is computed before the comparison; and the failure is reported with the same prominence as the agreements, because a theory is validated by where it breaks as much as by where it holds. Figure 2 summarizes.

10.1. Ordinal Validation

10.1.1. Test O1: The Coding Class Ranking

Prediction (from Theorems 1 and 2 at $n_S \approx n_W$): on SWE-bench Verified, machines rank by inverse coordination: minimal linear $\succeq$ planner-executor $\succ$ deep hierarchy.

Observed: mini-SWE-Agent 74+%, OpenHands 72%, Devin 45.8% [10, 11].

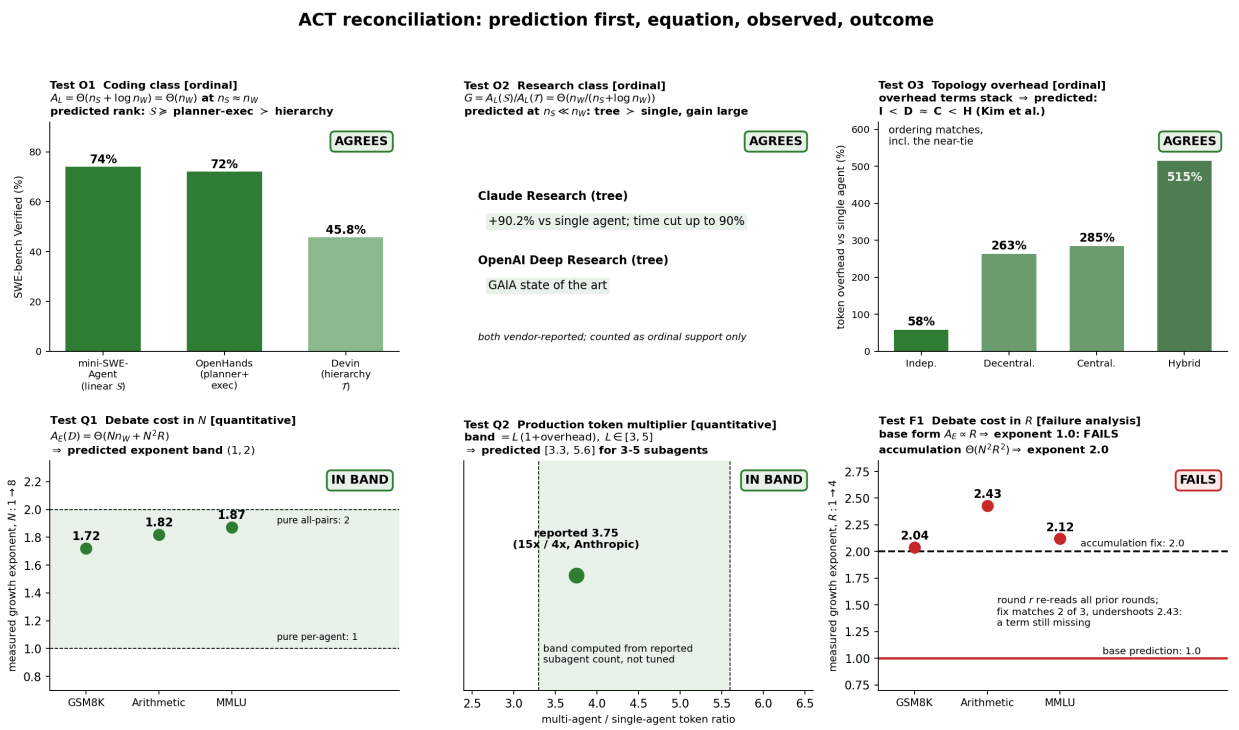

Figure 2: The reconciliation at a glance. Each panel is one test: prediction (from the theorems, stated first), the equation it comes from, the observed value, and the outcome. The round-scaling failure is displayed with the same prominence as the passes; its documented cause, transcript accumulation (Remark 2), corrects the exponent from 1 to 2.

Residual: all three pairwise orderings agree. The first margin (74 vs 72) is small, as the signature suggests it should be, since planner-executor overhead is a constant, not a growth rate; the second margin is large, and part of it is plausibly model quality (the Devin caveats of Section 8.4).

10.1.2. Test O2: The Research Class Ranking

Prediction (Corollary 2 at $n_S \ll n_W$): trees beat single agents, with gains growing in breadth.

Observed: Claude Research $+90.2\%$ over its single-agent baseline with time cut up to 90% [6]; Deep Research tops GAIA, where prior sequential baselines trailed by wide margins [7, 9].

Independent corroboration: the ranking does not rest on vendor reports alone. Hugging Face's open-source reproduction, a manager-plus-search-agent hierarchy built on smolagents, reports 55.15% pass@1 on the GAIA validation set against roughly 46% for the prior open agentic system and far below-agentic scores for plain sequential model calls [24]. An independent team, using open code and a different model stack, lands the same ordering: coordinated parallel machines above sequential ones on the parallel task class.

Residual: all checks agree. The two vendor reports (one on a private evaluation) contribute ordinal support; the open-source replication is independent and public, which is why we list it separately rather than folding it into the vendor evidence.

10.1.3. Test O3: The Topology Overhead Ordering

Prediction (from which overhead terms each topology activates, Section 5): at matched accuracy, token overhead orders as Independent $<$ {Decentralized $\approx$ Centralized} $<$ Hybrid, since Hybrid stacks its parts' overheads.

Observed: Kim et al. measure 58%, 263%, 285%, 515% [22].

Residual: the ordering matches, including the near-tie in the middle. Exact percentages are not claimed; the sources' internal token counts are not public.

10.2. Quantitative Validation

10.2.1. Test Q1: Debate Cost Scaling in N

Prediction (Prediction 3): the growth exponent $e(N)$ over $N \in [1, 8]$ must lie in the open band $(1, 2)$, approaching 2 as peer-reading dominates.

Observed: cost growth factors of $36\times$, $44\times$, $49\times$ on GSM8K, Arithmetic, and MMLU as N goes 1 to 8 [20], giving implied exponents

$$\frac{\ln 36}{\ln 8} = 1.72, \qquad \frac{\ln 44}{\ln 8} = 1.82, \qquad \frac{\ln 49}{\ln 8} = 1.87.$$

Residual: all three exponents fall inside the predicted band, without tuning. The band is not vacuous: pure per-agent accounting predicts exponent 1, pure all-pairs accounting predicts 2, and the data land where a mixture with a dominant quadratic term must.

10.2.2. Test Q2: The Production Token Multiplier

Prediction (Section 8.1, computed from the reported subagent count before the comparison): multi-to-single token multiplier in $[3.3, 5.6]$ for 3 to 5 subagents.

Observed: $15 \times / 4\times = 3.75$ [6].

Residual: inside the band, with the band derived from the tree derivation of Theorem 1 and the source's own subagent count, not tuned to the ratio.

10.3. Failure Analysis

10.3.1. Test F1: Debate Cost Scaling in R

Prediction (constant-per-round form of Section 5.4): exponent 1.0 in rounds.

Observed: growth factors $17\times$, $29\times$, $19\times$ over $R \in [1, 4]$, implied exponents 2.04, 2.43, 2.12 [20].

Analysis: the base model is wrong by a factor of about two in the exponent, and this is not noise. The documented mechanism is transcript accumulation: round r re-reads all prior rounds, turning the exchange total into $\Theta(N^2R^2)$ (Remark 2), predicted exponent 2.0, which matches two of the three measurements and still sits below the third (2.43). We record both facts: the correction was identified from the failure, and one benchmark grows faster than even the corrected form, so a term is still missing (output length growing with rounds is the leading candidate). This failure is the paper's strongest evidence that nothing here is fitted: a framework free to tune itself would not have produced it.

10.4. Summary of the Reconciliation

Table 3 collects the six tests. Read down the Kind column: the ordinal tests establish that the predicted rankings hold within each task class; the quantitative tests establish that where measured exponents and ratios exist, they land inside bands computed in advance; and the failure analysis establishes where the current form of the theory stops and what term is missing there.

Table 3: The reconciliation tests, grouped by the kind of evidence each provides. Each row states the prediction, computed before the comparison, and what was observed. No test involves a fitted parameter. The round-scaling failure and its correction are described in Section 10.3.

Test	Kind	Predicted	Observed
O1 coding ranking	ordinal	mini $\succeq$ OH $\succ$ Devin	$74 > 72 > 45.8$, agrees
O2 research ranking	ordinal	tree $\succ$ single	$+90.2\%$; GAIA SOTA, agrees
independent replication (open source)	ordinal	tree $\succ$ sequential	GAIA val. 55.15% vs prior, agrees
O3 topology overhead	ordinal	I $<$ D $\approx$ C $<$ H	$58 < 263 \approx 285 < 515$, agrees
Q1 debate cost in N	quantitative	exponent $\in (1, 2)$	1.72, 1.82, 1.87, in band
Q2 production multiplier	quantitative	$[3.3, 5.6]$	3.75, in band
F1 debate cost in R	failure analysis	1.0 (base form)	2.04, 2.43, 2.12, **fails**
after accumulation correction		2.0	matches 2 of 3, undershoots 2.43

10.5. What the Reconciliation Does and Does Not Establish

It establishes that, within each task class where public numbers permit comparison, the ranking of architectures predicted from intrinsic task structure agrees with the observed ranking, and that where per-configuration token measurements exist, the measured growth exponents fall in the predicted bands except for the documented round-scaling failure, which localizes a specific missing term. The contribution is not a pass rate; it is that the same small set of equations, with nothing fitted, predicts architectural behavior across independent sources and exposes its own breaking point. The reconciliation does not establish absolute accuracy prediction (ACT does not price answer quality), it does not certify private architectures, and it does not substitute for the controlled study of Section 14, which remains the decisive test.

11. How to Use ACT: Procedure and Worked Examples

11.1. The Procedure

ACT runs at design time, before anything is built (Figure 3).

Step 1. Write the task's decision DAG: list the atomic sub-decisions, draw only genuine data dependencies.

Step 2. Read off $\widehat{n}_W$ (node count) and $\widehat{n}_S$ (longest path); compute the intrinsic parallelism $\widehat{n}_W/\widehat{n}_S$.

Step 3. List the candidate machines: $\mathcal{S}$, $\mathcal{P}_k$, $\mathcal{T}(b, g)$, $\mathcal{D}(N, R)$.

Step 4. Evaluate each machine's signature at $(\widehat{n}_W, \widehat{n}_S)$ using Table 1.

Step 5. Rank: minimize A_E subject to the latency budget on A_L and the message budget on A_C.

Step 6. Size the winner's internal knobs: b as wide as context allows (Corollary 1), worker count near $\widehat{n}_W/\widehat{n}_S$ (Corollary 2), retry cap k small (Assumption 4).

The same procedure, stated as an algorithm so that it is reproducible and implementable, with the guarantee it does and does not carry made explicit:

Proposition 4 (Selection guarantee)**.** *Under Assumptions 1 to 4 and a correct annotation* $(\widehat{n}_W, \widehat{n}_S)$*,* ACT-SELECT *returns an architecture that satisfies both budgets and has minimal economic complexity among the enumerated candidate set* $\mathcal{C}$*, or correctly reports that no member of* $\mathcal{C}$ *is feasible. If the task's DAG has no join node, the returned replication is optimal over* all *architectures on all three active coordinates simultaneously, since it meets the lower bounds of Theorems 1 and 2 with zero messages.*

The proof is immediate from the construction: lines 5 to 8 evaluate closed forms proved in Section 5, the filter enforces the budgets, and the argmin is taken over the filtered set; the no-join case meets the universal lower bounds, so nothing can improve on it. The scope is equally explicit: the guarantee is relative to $\mathcal{C}$, and a graph outside $\mathcal{C}$ whose signature beats the returned machine is possible in principle, although by work-optimality (Theorem 1) it can win by at most a constant factor in economics and by Theorem 2 at most an additive $\log$ in latency.

11.2. Worked Example A: A Competitive-Intelligence Research Agent

The brief. An analytics team wants an agent that, given an industry sector, collects 80 public filings, extracts a fixed set of indicators from each, cross-compares them, and writes a ranked brief. Constraints, fixed by the team before any design work: latency budget $A_L \leq 15$ sequential τ-units, communication budget $A_C \leq 50$ messages, and no accuracy-critical requirement that would motivate redundant solving. All numbers below are properties of the task statement, written down before any architecture is considered, by the protocol of Section 4.

Step 1 (decision DAG). Two planning decisions (scope the sector, fix the indicator set); 80 independent extract-and-summarize decisions, one per filing; a five-step synthesis chain (normalize, cross-compare, rank, draft, cite-check). Dependencies: planning precedes extraction; every extraction precedes synthesis; synthesis steps are sequential.

Step 2 (intrinsic size).

$$\widehat{n}_W = 2 + 80 + 5 = 87, \qquad \widehat{n}_S = 2 + 1 + 5 = 8, \qquad \frac{\widehat{n}_W}{\widehat{n}_S} = \frac{87}{8} \approx 10.9.$$

Algorithm 1: ACT-SELECT *architecture selection at design time*

Input: annotated DAG $\widehat{D}$ (protocol of Section 4); latency budget Λ; message budget X; context width $b_{\max}$; role constants (τ, c_w, C_o, s, T_v); retry cap k, failure rate p.

Output: an architecture $M^\star$ with sized internal knobs, or INFEASIBLE.

1. $\widehat{n}_W \leftarrow |V(\widehat{D})|$; $\widehat{n}_S \leftarrow$ longest path of $\widehat{D}$; $P \leftarrow \widehat{n}_W/\widehat{n}_S$ $O(|V|+|E|)$
2. **if** $\widehat{D}$ has no join nodes **then return** N independent replicas of $\mathcal{S}$ (rule C2, no join term)
3. $g \leftarrow \lceil \widehat{n}_W / \min(P, b_{\max}^2) \rceil$; $L \leftarrow \lceil \widehat{n}_W/g \rceil$; $h \leftarrow \lceil \log_{b_{\max}} L \rceil$ (Cor. 2, 1)
4. $\mathcal{C} \leftarrow \{\, \mathcal{S},\ \mathcal{P}_k,\ \mathcal{T}(\min(L, b_{\max}), g)$ at depth h, $\mathcal{D}(N, R)$ if accuracy-critical $\}$ $\cup$ composites required by the DAG's shape (Section 6)
5. **for each** $M \in \mathcal{C}$: evaluate $\mathcal{A}(t; M) = (A_E, A_L, A_C, A_V)$ at $(\widehat{n}_W, \widehat{n}_S)$ by Table 1 for the families and rules C1 to C5 for composites $O(|\mathcal{C}|)$
6. $\mathcal{F} \leftarrow \{\, M \in \mathcal{C} : A_L(M) \le \Lambda \ \wedge\ A_C(M) \le X \,\}$
7. **if** $\mathcal{F} = \emptyset$ **then return** INFEASIBLE with the binding constraint (renegotiate Λ or X)
8. $M^\star \leftarrow \arg\min_{M \in \mathcal{F}} A_E(M)$; size knobs: width $\min(L, b_{\max})$, workers $\approx P$, retries $\le k$
9. **return** $M^\star$

Cost of the algorithm itself: one graph traversal plus a constant number of closed-form evaluations, $O(|V|+|E|+|\mathcal{C}|)$; no agent is invoked and no token is spent. *Correctness scope:* $M^\star$ minimizes A_E among the enumerated candidates under the stated assumptions; it is not claimed optimal over all conceivable graphs (Section 14).

The task is intrinsically parallel with useful parallelism about 11 (Corollary 2).

Steps 3 and 4 (signatures at $(87, 8)$, in units of τ and messages).

Machine	A_E	A_L	A_C	note
$\mathcal{S}$	87τ	87τ	0	work-optimal, 87 sequential units
$\mathcal{P}_5$	$87\tau + 5c_w$	$87\tau + \Theta(5)$	$\Theta(5)$	no latency gain
$\mathcal{T}(11, 8)$	$\kappa \cdot 87$, $\kappa \approx 1.2\tau$–1.4τ	$(8+2)\,\tau$-units	$\approx 2 \times 11$	see sizing below
$\mathcal{D}(4, 3)$	$4 \cdot 87\,\tau + 48s$	$(8+3)\tau$-units	$N^2R = 48$	$4\times$ redundant work

Tree sizing, from the corollaries rather than by guesswork: worker count should not exceed the useful parallelism ≈ 11, so choose $L = 11$ workers of granularity $g = 8$ (each handles 8 filings, $11 \times 8 = 88 \ge 87$); a single orchestrator holds $b = 11$ child summaries if $11s$ fits its context, giving $h = 1$ and $N_{\text{orch}} = 1$. Then by Theorem 1, $\kappa = \tau + c_w/8 + (C_o + 11s + T_v)/(8 \cdot 10)$, a 20 to 40% overhead for typical role constants, and by Theorem 2 the latency is the synthesis chain plus one dispatch-aggregate level: about 10 sequential units against the single agent's 87.

Step 5 (rank under constraints: the optimization, carried out). The selection rule of the procedure is: minimize A_E subject to $A_L \le 15$ and $A_C \le 50$. Evaluating each candidate against the constraints:

Machine	A_E (τ-units)	A_L (τ-units)	$A_L \le 15$?	$A_C \le 50$?	decision
$\mathcal{S}$	87	87	no ($5.8\times$ over)	yes (0)	**rejected: latency**
$\mathcal{P}_5$	$87 + 5c_w/\tau$	≥ 87	no ($5.8\times$ over)	yes (5)	**rejected: latency**
$\mathcal{T}(11, 8)$	$\approx$ 104–122	≈ 10	yes	yes (22)	feasible
$\mathcal{D}(4, 3)$	≥ 348	≈ 11	yes	yes (48)	feasible

Two machines are feasible, so the objective decides: $A_E(\mathcal{T}) \approx$ 104–122 against $A_E(\mathcal{D}) \ge 348$, a factor of about 3, and the debate's extra spend buys an accuracy mechanism the brief did not request. Note the pipeline's rejection is structural, not marginal: it has nearly the lowest cost in the table and still loses, because it optimizes the coordinate the task does not constrain (A_E) while missing the one it does (A_L). *Select $\mathcal{T}(11, 8)$ with a top verifier: the unique feasible minimizer.*

Step 6 (what was decided without building anything). The team now knows, before writing a line of code: the architecture family (flat tree), the worker count (11, from $\widehat{n}_W/\widehat{n}_S$, not from enthusiasm), the branching (as wide as context allows, from the overhead law), the expected cost premium over a single agent (20 to 40%), the expected speedup (about $8.7\times$), and the two numbers to monitor in production (if measured cost per filing grows faster than linearly, Assumption 3 is breaking; if latency exceeds the synthesis chain plus a constant, the orchestrator is a bottleneck).

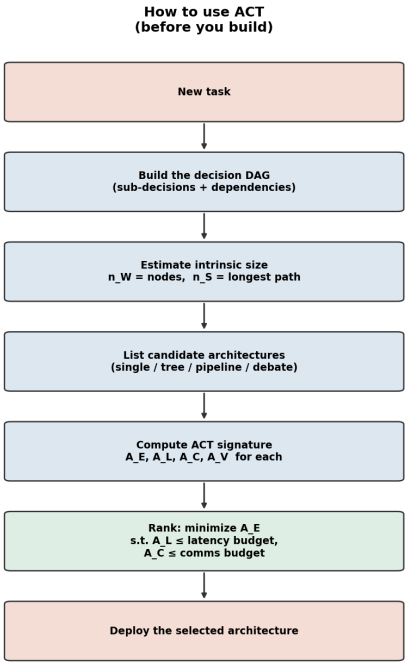

Figure 3: The ACT design-time procedure. The output is an architecture chosen from task structure and constraints, not from trial and error.

11.3. Worked Example B: The Same Procedure Saying No

The brief. A platform team wants to speed up an agent that fixes failing CI builds: reproduce, localize, patch, re-run, iterate.

Steps 1 and 2. The decision DAG is a chain of about 40 test-and-edit decisions, each consuming the previous result: $\widehat{n}_W \approx \widehat{n}_S \approx 40$, intrinsic parallelism ≈ 1.

Steps 3 to 5. By Theorem 2, every machine's latency is $\Theta(40)$; no architecture can beat the chain. By Theorem 1, any tree pays $\kappa > \tau$ per node for that nothing. *Choose $\mathcal{S}$ with a bounded reflective loop, and spend the freed budget on a stronger model (lower τ), the only lever that moves a chain.*

The two examples bracket the theory's use: it recommends coordination exactly when the task's structure can pay for it, and it refuses coordination when the structure cannot, which is the discipline missing from architecture choice today.

12. Design Studies: Choosing the Graph for Six Industry Workloads

The case studies of Sections 8 and 9 showed ACT explaining systems that already exist. This section shows ACT doing the job it was built for: choosing an architecture *before* implementation. We take six workloads enterprises commonly automate with agents, and for each we run the same design-study discipline: state the task, annotate its decision DAG under the protocol of Section 4, draw the plausible candidate graphs, evaluate every candidate's signature at the annotated size, and select under the stated constraints. Each study ends with the quantity that *decides* it, because the most useful output of a design study is knowing which property of the task to measure before committing. The annotations are illustrative applications of the protocol to representative briefs, labeled as such; they are not measurements, and a team applying ACT substitutes its own task's numbers. Table 4 collects the six decisions, Table 5 compresses them into a selection rule, and Figure 10 draws the whole section as one map.

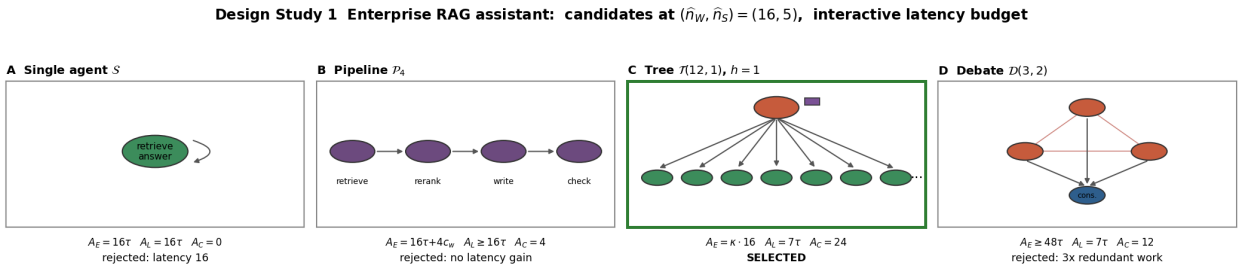

Figure 4: Design Study 1 (enterprise RAG): four candidate graphs for the same task, each with its signature at $(\widehat{n}_W, \widehat{n}_S) = (16, 5)$. The one-level tree is selected: it alone meets interactive latency without redundant work.

12.1. Design Study 1: Enterprise RAG Assistant

The brief. A knowledge assistant over an internal corpus (wikis, tickets, PDFs) answers employee questions with citations, in interactive time. Retrieval-augmented generation is the standard mechanism [25]; the architectural question is how to organize it when a question needs evidence from many places at once. Constraints fixed in advance: latency budget $\Lambda = 8$ sequential τ-units (interactive), message budget $X = 40$.

Decision DAG (protocol applied). Interpret the question (1 node); retrieve and extract evidence from d independent corpus regions, one node each (R2: each extract is consumed individually by the reconciliation); reconcile and rank the evidence (1); compose the cited answer (1); check citations (1). Edges: interpretation precedes every retrieval; every retrieval precedes reconciliation; the tail is a chain. For a representative multi-source question, $d = 12$:

$$\widehat{n}_W = 1 + 12 + 3 = 16, \qquad \widehat{n}_S = 5, \qquad \widehat{n}_W / \widehat{n}_S = 3.2.$$

The calculations, candidate by candidate (Figure 4). Each candidate is evaluated by substituting $(\widehat{n}_W, \widehat{n}_S) = (16, 5)$ into its signature; nothing else is needed.

Graph A, single agent $\mathcal{S}$. One context resolves all nodes in sequence:

$$A_E = \widehat{n}_W\, \tau = 16\tau, \qquad A_L = \widehat{n}_W\, \tau = 16\tau > \Lambda = 8\tau \quad \Rightarrow \quad \textbf{rejected: latency,}\ 2\times \text{ over budget.}$$

Graph B, pipeline $\mathcal{P}_4$ (retrieve $\to$ rerank $\to$ write $\to$ check). Stages partition the same work and remain sequential:

$$A_E = 16\tau + 4c_w, \qquad A_L = 16\tau + \Theta(4) > \Lambda \quad \Rightarrow \quad \textbf{rejected: latency}, \text{ reorganized but not shortened.}$$

Graph C, tree $\mathcal{T}(12, 1)$, one planner over 12 parallel retrievers ($g = 1$, $L = 12 \leq b_{\max}$, so $h = 1$):

$$A_L = \underbrace{\widehat{n}_S}_{5} + \underbrace{2h}_{2} = 7\tau \leq \Lambda, \qquad A_E = 16\tau + \underbrace{(C_o + 12s + T_v)}_{\text{one join}} = \kappa \cdot 16\tau, \qquad A_C = 2 \times 12 = 24 \leq X.$$

Feasible, at the usual coordination premium κ/τ.

Graph D, debate $\mathcal{D}(3, 2)$, three independent answerers plus consensus:

$$A_E = 3 \cdot 16\tau + \Theta(N^2 Rs) \geq 48\tau, \qquad A_L = 5 + 2 = 7\tau \leq \Lambda, \qquad A_C = N(N-1)R = 12 \leq X.$$

Feasible, at three times the tree's work.

Summary of the four evaluations:

Graph	Machine	A_E	A_L	A_C	outcome
A: iterative loop	$\mathcal{S}$	16τ	16τ	0	rejected: latency
B: retrieve-rerank-write	$\mathcal{P}_4$	$16\tau + 4c_w$	$> 16\tau$	4	rejected: latency
C: planner + 12 retrievers	$\mathcal{T}(12, 1)$	$\kappa \cdot 16\tau$	7τ	24	feasible
D: 3 answerers + consensus	$\mathcal{D}(3, 2)$	$\geq 48\tau$	7τ	12	feasible, $3\times$ cost

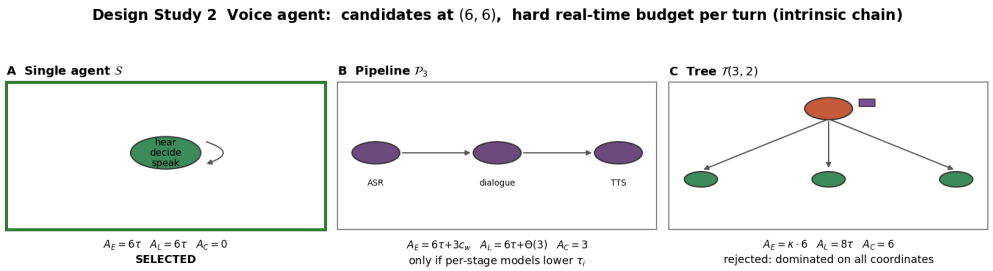

Figure 5: Design Study 2 (voice agent): three candidate graphs at $(6, 6)$. The task is an intrinsic chain, so the single agent is selected and the tree is dominated on every coordinate; the pipeline survives only as a constant-factor specialization argument.

Selection. Two candidates are feasible, so the objective decides: $A_E(\mathcal{T}) = \kappa \cdot 16\tau$ against $A_E(\mathcal{D}) \geq 48\tau$, and since κ/τ is a 20 to 40% premium while debate's is a 200%+ premium for an accuracy mechanism the brief did not request, the tree is the argmin. *Select the one-level tree, as wide as the question's evidence spread.*

What decides it. The evidence spread d. At $d \leq 3$ the coordination overhead κ/τ cannot amortize and the single agent wins; the crossover is where the latency saved, $(d-1)\tau$, exceeds the coordination paid, which for typical role constants is at d of about 4 to 6. Measure the typical d of your query mix before building.

12.2. Design Study 2: Voice Agent

The brief. A real-time voice agent (support, booking, triage) handles one conversational turn: hear, understand, decide, act, respond, under a hard sub-second latency budget per turn.

Decision DAG (protocol applied). Transcribe and understand the utterance (1); resolve intent against dialogue state (1); perform the required lookups, which in a support flow are typically sequential because each depends on the last (account, then entitlement, say 2); decide the action (1); compose and vocalize the reply (1). Every node consumes its predecessor's output:

$$\widehat{n}_W = 6, \qquad \widehat{n}_S = 6, \qquad \widehat{n}_W/\widehat{n}_S = 1.$$

The calculations, candidate by candidate (Figure 5). Substituting $(6, 6)$:

Graph A, single agent $\mathcal{S}$:

$$A_E = 6\tau, \qquad A_L = 6\tau, \qquad A_C = 0. \qquad \text{Minimal on every coordinate.}$$

Graph B, pipeline $\mathcal{P}_3$ (ASR $\rightarrow$ dialogue $\rightarrow$ TTS):

$$A_E = 6\tau + 3c_w, \qquad A_L = 6\tau + \Theta(3), \qquad A_C = 3.$$

Strictly worse than A in the growth model; survives only on the constant-factor argument that a fast specialized model per stage lowers the summed τ_i below the single model's.

Graph C, tree $\mathcal{T}(3, 2)$ ($L = 3$, $h = 1$). The chain is the longest branch, so the max over branches is the whole task:

$$A_L = \underbrace{6}_{\text{chain}} + \underbrace{2}_{\text{dispatch/aggregate}} = 8\tau > 6\tau, \qquad A_E = \kappa \cdot 6\tau > 6\tau, \qquad A_C = 6 > 0.$$

Worse than A on all three active coordinates: **dominated**, before any budget is even consulted.

Selection. By Theorem 2 at $n_S = n_W$, no graph can beat the chain, so every coordination hop is pure loss against a real-time budget: the tree is dominated on all four coordinates. The pipeline is justified only by heterogeneity, running a fast specialized model per stage (speech, dialogue, speech) so that each stage's τ_i is small, which is a constant-factor argument outside the growth rates. *Select the single agent; adopt the three-stage pipeline only if per-stage model specialization measurably lowers the summed τ_i.*

What decides it. Nothing about scale; only the per-node constant τ. A conversation turn is an intrinsic chain, and the only lever on a chain is a faster model. This is the design study that shows parallelism is not a default good.

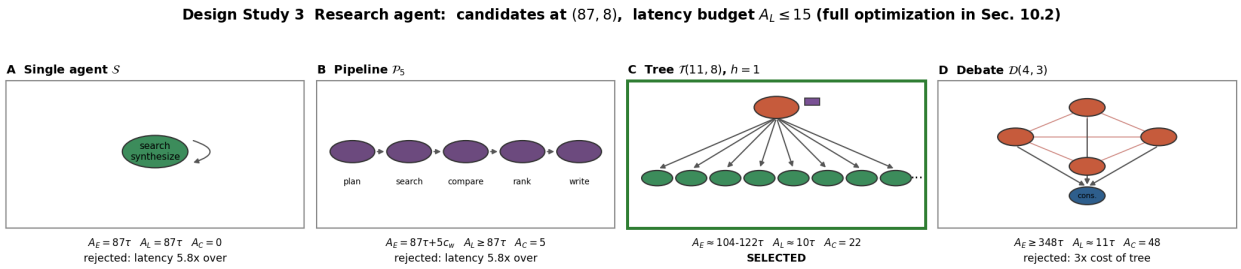

Figure 6: Design Study 3 (research agent): four candidate graphs at $(87, 8)$. The flat tree at width 11 is the unique feasible minimizer of the explicit optimization in Section 11.2.

12.3. Design Study 3: Research Agent

The brief. Breadth-first research (competitive intelligence, literature review, due diligence): fan out over many sources, then synthesize. This is the workload of Sections 8.1 and 8.2, and the one design study with production evidence on both the vendor and the independent side (Test O2).

Decision DAG. Worked Example A (Section 11.2) annotates the representative brief in full: $\widehat{n}_W = 87$, $\widehat{n}_S = 8$, intrinsic parallelism ≈ 11.

The calculations, candidate by candidate (Figure 6). The full constrained optimization is carried out in Section 11.2; the four evaluations, at $(87, 8)$ under $\Lambda = 15\tau$:

$$A_L(\mathcal{S}) = 87\tau, \qquad A_L(\mathcal{P}_5) \ge 87\tau, \qquad A_L(\mathcal{T}(11,8)) = 8 + 2 = 10\tau, \qquad A_L(\mathcal{D}(4,3)) = 8 + 3 = 11\tau,$$

so the first two are rejected at $5.8\times$ over budget, and among the feasible pair

$$A_E(\mathcal{T}) \approx 104\text{–}122\,\tau \;\text{ against }\; A_E(\mathcal{D}) \ge 4 \cdot 87\,\tau = 348\tau,$$

making the one-level tree at width 11 the unique feasible minimizer.

Selection. Select the flat tree, width set by $\widehat{n}_W/\widehat{n}_S$, granularity set so worker count matches that width. This is also the shape the production systems of the class converged to independently.

What decides it. The fan-out q (equivalently $\widehat{n}_W/\widehat{n}_S$), and the orchestrator's context width $b_{\max}$: at q beyond $b_{\max}$ the tree gains a second level (see Design Study 5).

12.4. Design Study 4: Technical Writing Agent

The brief. Produce a large technical document, say API documentation of a dozen chapters with worked examples, from a codebase and a style guide. This is the design study whose answer is conditional, which makes it the most instructive of the six.

Decision DAG, two legitimate annotations. Research the API surface: 10 module analyses, mutually independent (parallel layer). Outline (1). Draft $c = 12$ chapters. Consistency pass (1), review (1). The drafting edges are the crux, and the protocol forces the question into the open. *Regime (a), reference-style chapters:* each chapter depends only on the outline, and cross-references are resolved in the consistency pass; the drafting layer is parallel:

$$\widehat{n}_W = 10 + 1 + 12 + 2 = 25, \qquad \widehat{n}_S = 1 + 1 + 1 + 2 = 5, \qquad \widehat{n}_W/\widehat{n}_S = 5.$$

Regime (b), narrative chapters: each chapter builds on the previous (a tutorial), so drafting is a chain:

$$\widehat{n}_W = 25, \qquad \widehat{n}_S = 1 + 1 + 12 + 2 = 16, \qquad \widehat{n}_W/\widehat{n}_S \approx 1.6.$$

The calculations, regime by regime (Figure 7). The same three candidates are evaluated at both annotations.

Regime (a), at $(25, 5)$. Chapter tree $\mathcal{T}(12, 1)$ with $L = 12 \le b_{\max}$, $h = 1$:

$$A_L(\mathcal{T}) = 5 + 2 = 7\tau \qquad \text{against} \qquad A_L(\mathcal{S}) = A_L(\mathcal{P}_4) = 25\tau,$$

a $25/7 \approx 3.6\times$ latency gain purchased at the coordination premium $A_E(\mathcal{T})/A_E(\mathcal{S}) = \kappa/\tau \approx 1.2\text{–}1.4$. The tree wins whenever the document is wanted sooner than $3.6\times$ matters, which for documentation deadlines it typically is.

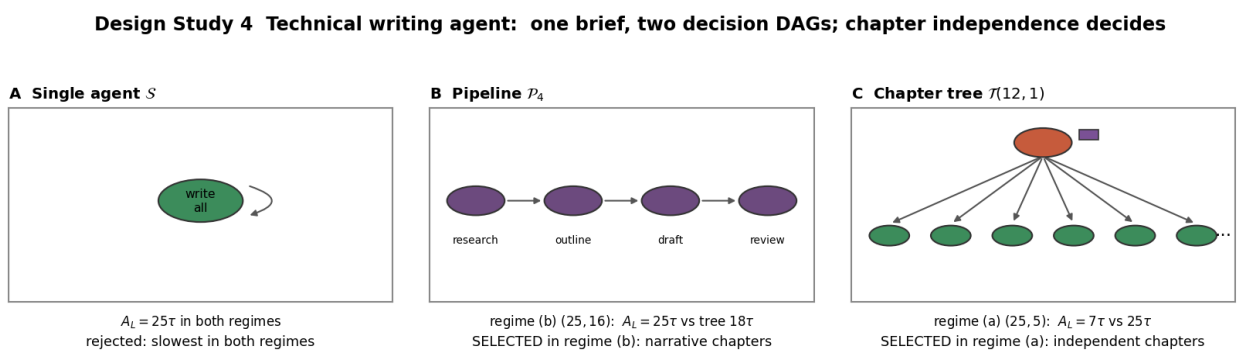

Figure 7: Design Study 4 (technical writing): the same brief admits two decision DAGs. With independent chapters, $(25, 5)$, the chapter tree is selected; with narrative chapters, $(25, 16)$, the stage pipeline is. The choice is made by an inspectable property of the outline, not by preference.

Regime (b), at $(25, 16)$. The drafting chain enters the span, and the same tree now yields

$$A_L(\mathcal{T}) = 16 + 2 = 18\tau \qquad \text{against} \qquad A_L(\mathcal{P}_4) = 25\tau,$$

a $25/18 \approx 1.4\times$ gain, purchased at the same κ premium *plus* the merge-quality risk of parallel drafts of a narrative. The pipeline, which matches the document's natural stages (research $\to$ outline $\to$ draft $\to$ review), wins on the totality.

Selection. Regime (a): tree over chapters. Regime (b): pipeline. The same brief, two defensible architectures, and the choice is made by an inspectable property of the DAG rather than by taste.

What decides it. Chapter independence, which is measurable before building: count the cross-chapter dependencies in the outline. If a chapter cannot be outlined without another chapter's *content* (not just its existence), it is regime (b).

12.5. Design Study 5: Product Testing Agent

The brief. Full product test-and-triage: execute a large regression suite, cluster the failures, root-cause each cluster, and produce a report. Overnight batch window, so the latency budget is loose but not unlimited; cost matters at this scale.

Decision DAG (protocol applied). Plan and select tests (1); execute $m = 1000$ test cases, mutually independent (R2: each verdict is consumed individually by clustering); cluster failures (1); root-cause $f = 20$ failure clusters, mutually independent (parallel); write the report (1):

$$\widehat{n}_W = 1 + 1000 + 1 + 20 + 1 = 1023, \qquad \widehat{n}_S = 5, \qquad \widehat{n}_W/\widehat{n}_S \approx 205.$$

The calculations, candidate by candidate (Figure 8). The interesting comparison is inside the tree family, because this is the scale at which Corollary 1's context constraint binds.

Graphs A and B, sequential runner and stage pipeline:

$$A_L(\mathcal{S}) = 1023\tau, \qquad A_L(\mathcal{P}_4) \geq 1023\tau :$$

at any realistic per-decision time, days of wall-clock. Both rejected even against an overnight window.

Graph C, flat tree $\mathcal{T}(128, 8)$. Sizing from the structure equations:

$$L = \left\lceil \frac{1023}{8} \right\rceil = 128 \quad \Rightarrow \quad b = 128 \text{ at } h = 1, \qquad \text{but} \qquad 128\, s \gg \text{any coordinator context},$$

so the width-first shape violates the $b\, s \leq$ context constraint: **infeasible**, not merely suboptimal.

Graph D, two-level tree $\mathcal{T}(12, 8)$. Depth is forced by exactly the quantity that killed C:

$$h = \lceil \log_{12} 128 \rceil = 2, \qquad A_L = \underbrace{5}_{\widehat{n}_S} + \underbrace{2h}_{4} = 9\tau, \qquad A_E = \kappa \cdot 1023\, \tau, \qquad A_C \approx 2\,(128 + 12).$$

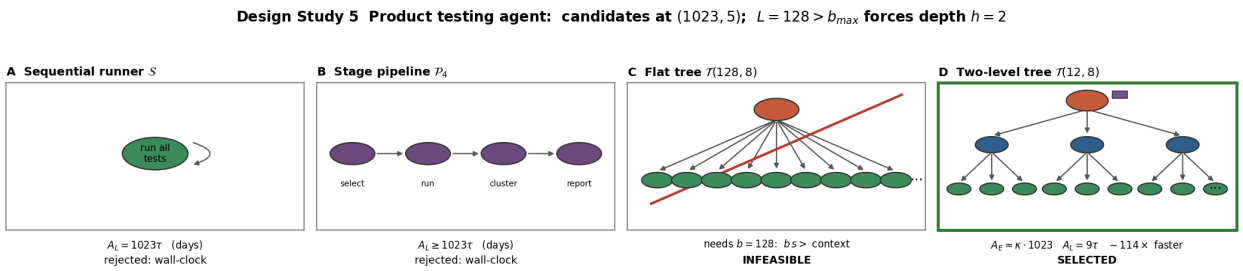

Figure 8: Design Study 5 (product testing): at $(1023, 5)$ the flat tree is infeasible because no coordinator context holds 128 child summaries, so depth appears for the one reason Corollary 1 permits: $L > b_{\max}$. The two-level tree is selected.

Against A this is a $1023/9 \approx 114\times$ wall-clock reduction at the usual κ premium, with the worker count 128 comfortably under the useful-parallelism ceiling $\widehat{n}_W/\widehat{n}_S \approx 205$ of Corollary 2, so none of the width is wasted.

Selection. Select the two-level tree: the only feasible candidate that converts the task's parallelism into wall-clock, and the overnight budget absorbs its two coordination levels easily.

What decides it. This is the one study where *depth* is correct, and it is correct for exactly the reason Corollary 1 names: L exceeds the widest branching a coordinator's context allows, and depth appears as $\lceil \log_b L \rceil$, inside the machine, only when forced.

12.6. Design Study 6: Multi-Agent Customer Support

The brief. A support operation resolves a queue of $T = 200$ tickets per hour. Each ticket is a short chain: classify, look up the account, check entitlement, take the action, reply. Constraints: per-ticket latency at most 6 sequential units; the queue must clear within the hour. This study has a different winner than the other five, and is decided at line 2 of Algorithm 1 before any signature is evaluated.

Decision DAG (protocol applied). Each ticket annotates to a 5-node chain, and tickets are mutually independent (R3: no ticket's question requires another ticket's answer). Crucially, the queue's DAG has *no join node*: each reply is delivered to its own customer, and nothing downstream consumes the set of replies together.

$$\widehat{n}_W = 200 \times 5 = 1000, \qquad \widehat{n}_S = 5, \qquad \widehat{n}_W/\widehat{n}_S = 200, \qquad \text{join nodes: } 0.$$

The calculations, candidate by candidate (Figure 9).

Graph A, one agent working the queue: $A_L = 1000\tau$; the queue does not clear. Rejected.

Graph B, orchestrator tree over tickets, $\mathcal{T}(12, 5)$**.** The signature evaluates as usual, $A_L = 5 + 2h$ per ticket and $A_E = \kappa \cdot 1000\tau$, but look at what the join term is buying: the orchestrator aggregates 200 ticket summaries *that no node of the DAG consumes*. By composition rule C2, the join terms $(C_o + bs)$ per internal node are attached to a join that does not exist in the task:

$$A_E(\mathcal{T}) - A_E(\text{replicas}) \;=\; \frac{L-1}{b-1}\,(C_o + bs + T_v) \;>\; 0 \quad \text{purchased for nothing.}$$

Graph C, $N = 200$ **independent replicas of** $\mathcal{S}$ (shard the queue, one agent per ticket):

$$A_E = 1000\,\tau \;\; (\text{exactly work-optimal, no } \kappa), \qquad A_L = 5\,\tau \text{ per ticket}, \qquad A_C = 0.$$

Every coordinate is simultaneously optimal: the lower bound of Theorem 1, the intrinsic floor of Theorem 2, and zero messages.

Selection. Select replication, not orchestration: N independent single agents behind a queue. Algorithm 1 returns this at line 2, from the single structural fact that the DAG has no join node.

What decides it. The existence of a join node. Fan-out with a join is a tree's territory; fan-out without one is a sharding problem, and adding an orchestrator to a sharding problem is the most common over-engineering ACT diagnoses in practice.

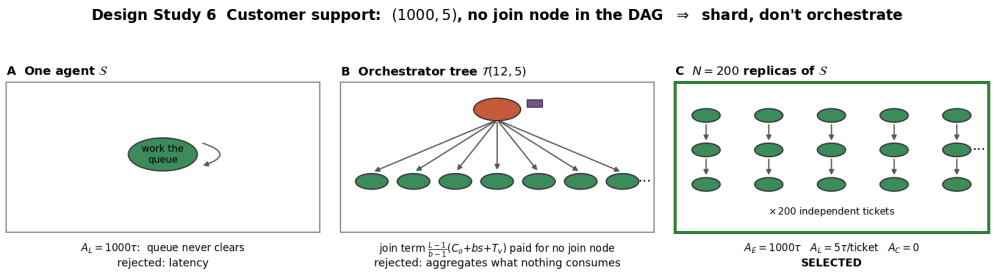

Figure 9: Design Study 6 (customer support): the queue's DAG has no join node, so the orchestrator of graph B aggregates summaries nothing consumes. Replication (graph C) is simultaneously optimal on all three active coordinates, and Algorithm 1 selects it structurally, before any signature is evaluated.

Table 4: The six design studies. Candidates are evaluated at the annotated intrinsic size under the stated constraints; the deciding quantity is the property of the task to measure before building.

Workload	$(\widehat{n}_W, \widehat{n}_S)$	Candidates	ACT selection	Deciding quantity
Enterprise RAG	$(16, 5)$	single, pipeline, tree, debate	flat tree	evidence spread d per query
Voice agent	$(6, 6)$	single, pipeline, tree	single agent	none; chain $\Rightarrow$ lower τ
Research agent	$(87, 8)$	single, pipeline, tree, debate	flat tree, width $\approx n_W/n_S$	fan-out q vs context $b_{\max}$
Technical writing	$(25, 5)$ or $(25, 16)$	single, pipeline, tree	tree (a) / pipeline (b)	chapter independence
Product testing	$(1023, 5)$	single, pipeline, trees	two-level tree	$L > b_{\max}$ forces depth
Customer support	$(1000, 5)$, no join	single, tree, replicas	N replicas of $\mathcal{S}$	existence of a join node

12.7. The Decision Matrix and the Selection Rule

The matrix is the section's argument in miniature: six workloads, five different answers, and in every case the answer is read off a measurable property of the task rather than argued from fashion. Three of the six selections are not trees, one flips between two architectures on a single inspectable bit, and one rejects orchestration entirely, which is what distinguishes a selection framework from an advocacy exercise. Figure 10 compresses the whole section into one picture.

13. Threats to Validity

We separate the threats by kind, in the discipline software engineering research uses, because naming a threat precisely is the first step to discharging it.

Internal validity (did we measure what we think we measured?). The reconciliation compares predictions against numbers produced by others, under conditions we did not control: model choice, prompt engineering, and evaluation harness all vary across the compared systems. The coding-class ranking in particular conflates scaffold with model quality, and we flag the Devin comparison as confounded for exactly this reason (Section 8.4). The debate exponents are the strongest internal evidence because they come from one source varying one factor at a time. The discharge is the controlled study of Section 15: same model, same prompts, same tools, same tasks, different architectures, which no public dataset currently provides.

External validity (does it generalize?). Six of the ten case-study systems are vendor-built, and vendor reports may select favorable metrics. We mitigated this with the open-source replication in Test O2 and with the academic sources in Tests O3, Q1, and F1, but the research-class evidence remains vendor-weighted. The four machine families capture the dominant production patterns analyzed here, not all conceivable coordination schemes; Section 6 extends coverage by composition, but composed signatures have not themselves been validated against measured hybrids.

Table 5: The ACT architecture selection rule, compressed. Conditions are evaluated on the annotated decision DAG of Section 4; every row follows from the theorems and corollaries of Section 5.

If the task satisfies…	then select…	by
$\widehat{n}_S \approx \widehat{n}_W$ (intrinsic chain)	single agent; spend on lower τ	Thm. 2
$\widehat{n}_W \gg \widehat{n}_S$ and $L \leq b_{\max}$	flat tree, width $\approx \widehat{n}_W/\widehat{n}_S$	Cor. 2, 1
$\widehat{n}_W \gg \widehat{n}_S$ and $L > b_{\max}$	tree of depth $\lceil \log_{b_{\max}} L \rceil$	Cor. 1
fan-out with no join node in the DAG	replicate single agents (shard)	rule C2, Sec. 6
sequential specialization with distinct stage models	pipeline (constant-factor case)	Sec. 5.2
accuracy worth a quadratic communication cost, small $\widehat{n}_W$	debate, small N and R	Eq. (7)
task class varies at runtime	adaptive orchestrator	Sec. 9

Construct validity (do our variables mean what we claim?). The intrinsic sizes are uncomputable and the annotated proxies are one-sided over-counts, so every empirical statement in this paper is about $(\widehat{n}_W, \widehat{n}_S)$, not (n_W, n_S). The protocol of Section 4 makes annotation auditable, but inter-annotator agreement has not yet been measured; until it is, the possibility that different annotators produce materially different sizes on the same family remains open, and we list that measurement as the immediate next step rather than assuming it away. ACT's cost constructs also deliberately exclude answer quality, so no claim here should be read as a claim about accuracy.

Conclusion validity (are the inferences statistically sound?). The quantitative tests compare point predictions against point measurements without confidence intervals, because the sources publish single runs; the exponent band test is robust to this (the band is wide and the three benchmarks agree independently), but the production multiplier test rests on one reported ratio. Where variance data does not exist we have not invented it, and we treat every single-source number as one observation, not a distribution.

14. Limitations

The scope of the claims is as follows. ACT prices resources, not answer quality; whether an architecture returns a correct answer is a separate axis, and the right architecture maximizes quality per unit of the cost ACT measures. The intrinsic sizes are uncomputable in general (Proposition 1) and annotated in practice, so a task's true (n_W, n_S) may differ from its labeled decomposition. The scaling laws assume a roughly uniform machine, summary-sized messages, and independent bounded retries (Assumptions 1 to 4), and Section 10 exhibits a measured regime, debate rounds, where one of these assumptions fails and a corrected term is required. The validity threats are catalogued separately in Section 13. Most importantly, the decisive test is a controlled study the authors have not yet run: annotate a benchmark family by decision-DAG size, run fixed architectures across sizes, fit the constant κ on a training range, and predict economic and latency cost on held-out sizes, testing Theorem 1's linear growth in $\widehat{n}_W$ and Theorem 2's $\widehat{n}_S + \log \widehat{n}_W$ latency in the one variable a model upgrade cannot change. We do not claim ACT selects a provably optimal architecture. We claim it provides a principled, testable way to compare candidate architectures under common complexity metrics, and that across the evaluated production systems its predicted rankings align with observed trade-offs.

15. A Research Agenda

ACT is built to be a starting point, and what follows is an agenda, not a to-do list: each item is a research program with its own questions, stated with the result that would constitute progress. If the abstraction of this paper is sound, each program inherits it; if any program falsifies it, that too is progress.

The controlled scaling study. The decisive test named in Section 14: annotate a benchmark family by decision-DAG size under the protocol of Section 4, run fixed architectures across sizes, fit κ on a training range, and predict held-out sizes. Progress is a held-out validation, or falsification, of Theorems 1 and 2 in the one variable a model upgrade cannot change.

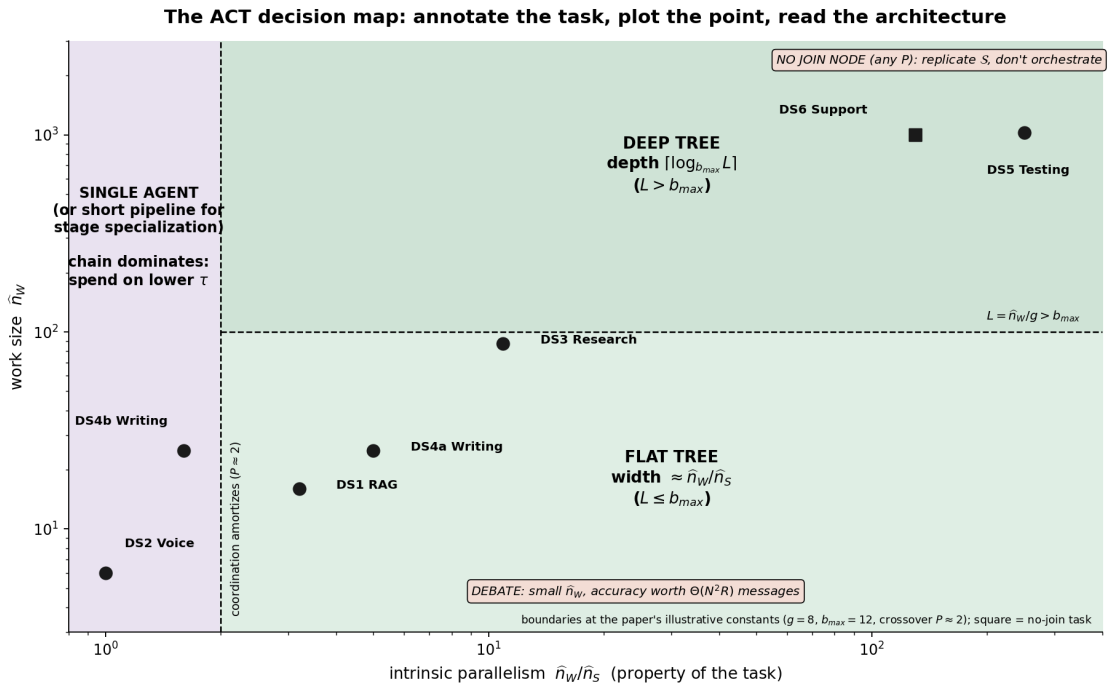

Figure 10: The ACT decision map. The horizontal axis is intrinsic parallelism $\widehat{n}_W/\widehat{n}_S$, the vertical axis is work size $\widehat{n}_W$; both are properties of the annotated decision DAG, so a task is a point and the region it lands in is the recommendation. The six design studies of this section are plotted at their annotated coordinates.

Annotation at scale. Measure inter-annotator agreement of the protocol on existing multi-hop and planning benchmarks, then learn to extract decision DAGs from task statements automatically. Progress is an extractor whose $(\widehat{n}_W, \widehat{n}_S)$ agree with human annotation within a stated tolerance, which would turn intrinsic size from an annotation into a measurement.

ACT-guided orchestration. The adaptive machine of Section 9 run in earnest: an orchestrator that estimates $\widehat{n}_W/\widehat{n}_S$ at dispatch time and selects its own topology. Progress is a system that matches the best fixed architecture per task class without being told the class.

Dynamic architectures. Real tasks reveal their DAGs incrementally; the machine that re-plans as the frontier of the DAG unfolds has no signature in Table 1 yet. Progress is extending the derivations of Section 5 to machines whose topology is a function of the partially observed task.

Accuracy complexity. ACT prices resources; the missing axis is how answer quality grows with resources spent on a task of intrinsic size (n_W, n_S). Progress is a quality analog of the signature, at which point architecture choice becomes a genuine two-objective optimization rather than cost minimization under a quality assumption.

Benchmark generation. Invert the protocol: generate task families with prescribed (n_W, n_S), giving the community benchmarks whose difficulty is controlled by construction rather than estimated after the fact. Progress is a public family spanning an order of magnitude in both coordinates.

The missing debate term. Section 10.3 localizes it: per-round output growth beyond transcript accumulation. Progress is a measured model of round cost that closes the gap to the 2.43 exponent without breaking the N-scaling band.

Energy and memory complexity. The signature's four coordinates are not privileged; energy per task and peak memory (context residency across concurrent agents) are resources with the same structure, and each should have its own coordinate with its own lower bound. Progress is a derivation of either coordinate for the tree machine with a matching intrinsic bound, extending the signature to $(A_E, A_L, A_C, A_V, A_{\text{mem}}, A_{\text{energy}})$.

An ACT compiler and runtime. Algorithm 1 is a selection procedure a human runs; the natural end state is a compiler that takes an annotated task description and emits the selected role graph as executable orchestration code,

and a runtime that meters τ, κ, Γ live and re-plans when the measured constants drift from the assumed ones. Progress is a system in which the architecture is an output of the toolchain rather than a hand design, with the adaptive machine of Section 9.6 as its scheduler.

16. Conclusion

ACT separates two things the cost literature fused: how much a configuration spends, and how a machine's resource use grows in the intrinsic difficulty of the task. By fixing the architecture as machine and the minimal decision DAG as intrinsic size, we obtained complexity signatures with matching lower bounds; four falsifiable predictions derived by substitution before the data; ten representative production workflows, spanning the dominant architectural families, reconciled against those predictions, with quantitative bands that contain the measured exponents and one documented failure that localizes a missing term; and a worked procedure that carries a practitioner from a task description through an explicit constrained optimization to an architecture choice using nothing but the formulas in this paper. The mathematics is deliberately simple, in the tradition of the parallel-computing laws it generalizes, and its value is that it holds in a variable no model upgrade can erase. Complexity theory guides algorithm choice before a program is written; ACT is built so that it can do the same for agent architectures, and Sections 14 and 15 name the experiment that would establish it and the research program it opens.

A. Interpreting the Constants

ACT discards constants under Θ, and Assumption 1 explains why that is the right theoretical move. A practitioner sizing a deployment still has to know what each constant *is* and where to read it off. This appendix gives the interpretation and the estimation route for every constant in the paper; deliberately, no numbers, since the numbers are what change with every model generation while the interpretations do not.

Constant	What it is	How to estimate it from your own logs
τ	marginal resource to resolve one atomic subdecision on the current model	regress total tokens on $\widehat{n}_W$ across tasks of one annotated family; the slope is τ, and a model upgrade shows up as the slope dropping
c_w	worker framing: system prompt, instructions, tool schemas, per call	fixed input-token count of any single worker call; the intercept of the same regression
C_o	orchestrator fixed cost: planning prompt plus dispatch framing	input plus output tokens of a lead-agent call with its child summaries subtracted
s	one child's condensed summary, as read by its parent	directly visible in any parent call; the design lever, since $b_{\max} \approx$ context budget / s
T_v	one verifier invocation	any verifier call's total tokens
p, k, Γ	per-node failure rate, retry cap, and the resulting multiplier $\Gamma = \frac{1-p^k}{1-p}$	p is the observed node-level failure rate; k is configured; correlated failures (one bad prompt failing everywhere) violate Assumption 4 and show up as Γ underpredicting
$b_{\max}$	widest branching one coordinator sustains	context budget divided by s, minus planning headroom
$\kappa(b, g)$	the tree's all-in per-node cost, $\tau + \frac{c_w}{g} + \frac{C_o + bs + T_v}{g(b-1)}$	compute from the rows above; the worked examples' 1.2–1.4$\times$ range is a statement about typical ratios of these constants, not a universal value

Two practical notes. First, the regression route for τ doubles as a health check on the annotation: if tokens do not grow linearly in $\widehat{n}_W$ within a family, either the annotation is inconsistent (protocol rule R5 violated) or an assumption is breaking, and the residual pattern says which. Second, every constant here is per model and per role; a heterogeneous deployment (cheap workers, strong orchestrator) simply carries one column of constants per role, which changes κ and nothing else in the theory.

A.1. Sensitivity: What Moves When a Constant Doubles

Because every constant enters through κ, Γ, or $b_{\max}$ and never through an exponent, sensitivity analysis is short and worth stating exactly. Doubling c_w, C_o, or T_v raises κ by at most a factor of two and changes no growth rate and no ranking driven by growth rates: a task in the flat-tree region of the decision map stays there. Doubling τ (a weaker model) scales every machine's cost equally and cancels out of all comparisons, which is the invariance the theory is built on. The two constants that *can* flip a selection are the ones that sit inside a boundary condition. Doubling s halves $b_{\max} \approx \text{context}/s$, which can push L past $b_{\max}$ and convert a flat-tree selection into a deep-tree one (the Design Study 5 boundary); it also raises the debate exchange term quadratically faster than the tree's join term, widening the tree's margin. And $p \to 1$ diverges Γ for every machine at once (Assumption 4), which is not an architecture change but a stop signal: no selection survives a per-node failure rate near one, and the correct response is redesigning the node, not the graph. In summary: cost constants move costs, boundary constants move selections, and only Γ can move the enterprise.

B. Notation

Symbol	First defined	Meaning
$D(t)$, $\widehat{D}(t)$	Def. 2, Def. 3	minimal decision DAG of task t; annotated (proxy) DAG
n_W, n_S	Def. 2	intrinsic work size (node count) and span size (longest path)
$\widehat{n}_W$, $\widehat{n}_S$	Def. 3	annotated work and span sizes; $\widehat{n}_W \geq n_W$
P	Sec. 11	intrinsic parallelism $\widehat{n}_W/\widehat{n}_S$
$\mathcal{S}$, $\mathcal{P}_k$, $\mathcal{T}(b,g)$, $\mathcal{D}(N,R)$	Sec. 3	single agent; k-stage pipeline; tree of branching b, granularity g; debate of N peers, R rounds
$\mathcal{A}(t;M)$	Def. 4	complexity signature (A_E, A_L, A_C, A_V) of machine M on task t
A_E, A_L, A_C, A_V	Def. 4	economic, latency, communication, verification complexity
τ, c_w	Asm. 1	cost of one atomic decision; worker framing constant
C_o, s, T_v	Asm. 2	orchestrator fixed cost; child summary size; verifier cost
$\Gamma(p,k)$	Asm. 4	retry multiplier at failure rate p, cap k
L, h, N_{orch}	Eq. (5), (6)	worker count $\lceil n_W/g \rceil$; depth $\lceil \log_b L \rceil$; internal nodes $\frac{L-1}{b-1}$
$\kappa(b,g)$	Thm. 1	the tree's all-in per-node cost constant
$b_{\max}$	Cor. 1	widest branching one coordinator's context sustains
Λ, X	Sec. 11	latency budget; message budget
C1–C5	Sec. 6	composition operators: series, parallel, hierarchy, feedback, conditional

References

[1] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proc. AFIPS Spring Joint Computer Conference*, pages 483–485, 1967.

[2] R. P. Brent. The parallel evaluation of general arithmetic expressions. *Journal of the ACM*, 21(2):201–206, 1974.

[3] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.

[4] A. C.-C. Yao. Some complexity questions related to distributive computing. In *Proc. 11th ACM Symposium on Theory of Computing (STOC)*, pages 209–213, 1979.

[5] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

[6] Anthropic. How we built our multi-agent research system. Engineering blog, June 2025. `https://www.anthropic.com/engineering/multi-agent-research-system`

[7] OpenAI. Introducing deep research. February 2025. `https://openai.com/index/introducing-deep-research/`

[8] G. Mialon, C. Fourrier, C. Swift, T. Wolf, Y. LeCun, and T. Scialom. GAIA: a benchmark for general AI assistants. *arXiv:2311.12983*, 2023.

[9] OpenAI. Deep research system card and benchmark results (GAIA 67.36% pass@1, 72.57% cons@64; HLE 26.6%). February 2025. https://openai.com/index/introducing-deep-research/

[10] SWE-bench Verified leaderboard. Accessed July 2026. https://www.swebench.com/

[11] SWE-agent team (Princeton and Stanford). mini-SWE-agent: the 100-line AI agent that scores above 74% on SWE-bench Verified. https://github.com/SWE-agent/mini-swe-agent

[12] J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan, and O. Press. SWE-agent: agent-computer interfaces enable automated software engineering. *arXiv:2405.15793*, 2024.

[13] Cognition. Introducing Devin, the first AI software engineer. March 2024. https://cognition.ai/blog/introducing-devin

[14] X. Wang et al. OpenHands: an open platform for AI software developers as generalist agents. *arXiv:2407.16741*, 2024.

[15] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. ReAct: synergizing reasoning and acting in language models. *arXiv:2210.03629*, 2022.

[16] Q. Wu et al. AutoGen: enabling next-gen LLM applications via multi-agent conversation. *arXiv:2308.08155*, 2023.

[17] LangChain. LangGraph: build stateful, multi-actor applications with LLMs. https://github.com/langchain-ai/langgraph

[18] CrewAI. Framework for orchestrating role-playing autonomous AI agents. https://github.com/crewAIInc/crewAI

[19] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv:2305.14325*, 2023.

[20] T. Liu et al. GroupDebate: enhancing the efficiency of multi-agent debate using group discussion. *arXiv:2409.14051*, 2024.

[21] The Ringelmann effect in multi-agent LLM systems: debate token scaling. *arXiv:2606.02646*, 2026.

[22] Y. Kim et al. Towards a science of scaling agent systems. *arXiv:2512.08296*, 2025.

[23] D. Tran and D. Kiela. Single-agent LLMs outperform multi-agent systems under equal thinking token budgets. *arXiv:2604.02460*, 2026.

[24] Hugging Face. Open-source DeepResearch: freeing our search agents. Blog and open-source release, February 2025. https://huggingface.co/blog/open-deep-research

[25] P. Lewis et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2005.11401.