From Code to Intent: Re-Architecting Internet Computing for the AI Era

Douglas C. Schmidt
William & Mary
Williamsburg, VA, USA

Abstract. Generative AI is redefining how we solve problems by loosening the coupling between traditional coding and computational thinking, which is often defined as telling computers *what* to do without specifying *how* to do it [1]. Rather than coding Internet applications explicitly using traditional programming languages, practitioners can now leverage large language models (LLMs) and agentic AI systems that can act autonomously to accomplish complex tasks through natural language intent. This column examines how generative AI tools and frameworks – from automated agents like OpenAI's Deep Research to agent orchestration platforms like Google's Opal – are unbinding computational thinking from code syntax and making "no-code" problem-solving a reality at scale. We highlight use cases and consider opportunities like democratized access and accelerated development.

Keywords: Generative AI; No-Code Development; Agentic Systems; Workflow Orchestration

Introduction: From Code-Centric Systems to Intent-Centric Internet Computing

For decades, turning ideas into working software required mastery of programming languages and low-level implementation details like syntax, memory management, and debugging. As a result, computational thinking was insufficient and only those with coding expertise could reliably translate ideas into executable systems. Generative AI is now eroding that barrier by allowing users to specify objectives in natural language and automatically translating them into executable solutions. This capability enables an intent-first approach that shifts development from *how* software is implemented to *what* outcome is desired [2].

This transition mirrors earlier abstractions in computing: intent-driven AI systems further extend the gains of higher-level languages and platforms. LLMs make it possible for non-specialists to analyze data, build applications, and automate tasks by describing goals and constraints rather than writing code. Computational thinking has thus become universal, opening software creation more broadly.

Why does this shift matter? In traditional Internet computing, applications were built through deterministic code, stable APIs, and predefined workflows. As we move toward intent-driven interactions, however, the Internet's infrastructure faces new demands. The emerging model is one where high-level user requests trigger AI-driven planning processes that invoke multiple services and tools dynamically to satisfy the intent, as shown in Figure 1.

Figure 1: From Intent to Outcome: AI-mediated Internet Computing.

This AI-mediated Internet computing model introduces several fundamental architectural shifts. Systems must contend with non-deterministic workflows, long-running and iterative execution, and probabilistic rather than deterministic correctness. These characteristics upend long-standing assumptions, e.g., protocols and orchestration mechanisms optimized for fast, idempotent operations must now support minutes-long, non-deterministic AI sessions. This shift also complicates caching and state.

Treating intent as a primary input demands adaptive systems that can tolerate ambiguity in natural-language requests and integrate runtime feedback. These systems must also dynamically coordinate heterogeneous tools through semantic orchestration [3] that maps high-level goals to executable services, such as selecting tools, sequencing calls, and validating outputs. The reward is greater expressiveness: users define outcomes, and AI-mediated systems determine how to achieve them.

At the same time, this shift requires engineers to architect for uncertainty explicitly. Generative AI is therefore not eliminating software engineering. Instead, it is elevating it, from writing deterministic code to orchestrating, integrating, and governing AI-driven processes. To ground the discussion, we examine two recurring architectural patterns [4] enabled by generative AI in the context of Internet computing: (1) *Agentic* R*etrieval-and-Synthesis*

Pipelines for automated research and analysis and (2) *Dynamic Intent-to-Workflow Orchestration* of "mini-apps," using representative case studies for each.

These exemplars are inspired by OpenAI's Deep Research and Google's Opal, which show how high-level intent is becoming a primary input in modern AI-mediated systems. We discuss cross-cutting challenges (from trust and security to performance) that arise and reflect on the broader implications for Internet computing practice. Generative AI is decoupling computation from coding, but realizing its promise at Internet scale requires rethinking architectures, standards, and engineering best practices.

Pattern 1: Agentic Retrieval-and-Synthesis Pipelines

One emerging architectural pattern is the *Agentic Retrieval-and-Synthesis Pipeline*, in which an AI agent autonomously retrieves information from multiple sources, analyzes it, and produces a synthesized result with explicit citations. Unlike traditional web search—which returns links—this pattern performs deep, cross-source analysis on the user's behalf. The pipeline includes multi-step retrieval, analysis, synthesis, and provenance tracking. The goal is to deliver actionable knowledge through a natural-language query interface, rather than raw search results.

Exemplar – OpenAI's Deep Research for automated retrieval and synthesis. An example of this pattern is OpenAI's Deep Research agent, introduced in early 2025 [5], with similar capabilities now appearing across leading LLM platforms. Deep Research conducts multi-step Internet research for complex queries, often completing in minutes tasks that would take humans hours to do manually [6]. Users provide a high-level prompt, such as "Identify U.S. first-year Computer Science major intent trends from 1971–2025" (see [7] for a video walkthrough of this example). Deep Research then autonomously searches the web, analyzes relevant articles, papers, and databases, and synthesizes a coherent report with valid cited sources. No scraping scripts or custom code are required; users simply clarify intent when needed and review the resulting analysis.

How it works. Deep Research orchestrates a workflow familiar to human researchers, but automated end-to-end. After receiving a query, the agent may request clarification from the user and dispatch browser tools to retrieve relevant sources. It then extracts and cross-verifies key information, progressively building an understanding of the topic. Finally, it compiles a structured report with citations linking each claim to its source.

For example, Figure 2 shows how Deep Research can identify authoritative U.S.-wide datasets on Computer Science major intent over the past five decades via natural-language intent alone, without manual coding. It can then compile annual statistics and generate a time-series visualization with inline references. Multi-step retrieval, cross-source synthesis, and explicit provenance are defining characteristics of this pattern.

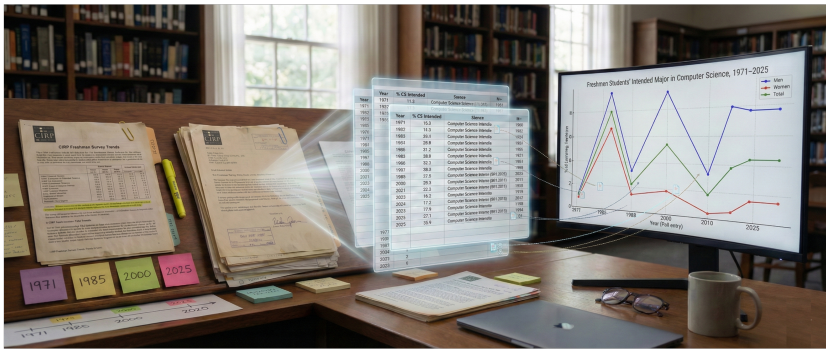

Figure 2: Visualizing CS Majors from 1971 to 2025.

No-code, high-depth inquiry. Deep Research delivers in one AI-driven pipeline what traditionally required substantial coding and manual effort by collapsing tasks into a single service accessed via natural-language prompts and intent. All intermediate computation is handled internally, and the output is delivered with provenance. This pattern offers analysis-as-a-service, where users pose questions and an AI agent returns actionable, source-backed answers.

Early applications span public policy, science, medicine, education, and finance [8]. Analysts use such agents to generate rapid trend reports, researchers survey literature on niche topics, and students gather background material efficiently. In these workflows, humans shift from searching and collating information to reviewing synthesized knowledge—evaluating sources, checking conclusions, and deciding how to apply the results.

Failure modes and trust. Despite its power, the *Agentic Retrieval-and-Synthesis Pipeline* pattern has limitations. Agents may miss subtle details, overgeneralize when information is sparse, or hallucinate facts, which are well-known failure modes of LLMs [9]. Bias in source selection can also skew results, and coverage may be incomplete when relevant data is unavailable or behind paywalls. Human oversight remains essential, including spot-checking key claims. In practice, the agent performs the labor-intensive retrieval and synthesis, while users provide judgment,

validation, and contextual understanding. “No-code” computational thinking does not eliminate the need for human insight; it reallocates effort from execution to oversight.

Implications for Internet computing. This pattern fundamentally changes how online information is accessed and consumed. Instead of keyword-driven search, AI systems increasingly deliver synthesized answers on demand. Knowledge portals, enterprise search tools, and digital libraries may integrate such agents to provide direct analysis rather than links. However, this shift introduces new engineering challenges. AI-driven search workflows are computationally expensive since they may run for minutes per query and incur higher costs per query than traditional search. Caching synthesized results, treating complex queries as batch jobs, or even adopting new economic models may be necessary to manage these trade-offs.

From an architectural perspective, Internet-scale retrieval and synthesis is evolving from indexing and document matching to orchestrating complex reasoning pipelines. Designers must ensure these pipelines are observable (e.g., via audit logs), robust to partial failures (e.g., handling sources that become unavailable mid-search), and secure (e.g., sandboxing any browsing or tool use). Despite these challenges, the payoff is significant: users, including non-researchers, can ask complex questions and receive synthesized, evidence-backed answers. In effect, the *Agentic Retrieval-and-Synthesis Pipeline* pattern positions generative AI as a new type of ‘knowledge worker’ within the Internet ecosystem. It transforms human users from information hunters into informed consumers of AI-generated insights.

Pattern 2: Dynamic Intent-to-Workflow Orchestration

Dynamic Intent-to-Workflow Orchestration is a second architectural pattern enabled by generative AI, in which user intent is translated on-the-fly into an executable, multi-step workflow. Rather than requiring developers to rely on vibe-coding, low-code prebuilt flows, or manual integration logic, an AI planner dynamically assembles and executes the necessary sequence of models and services. In this model, the AI acts as a meta-programmer or conductor, coordinating specialized components to fulfill the user’s goal.

*Exemplar – Google’s Opal platform***.** A representative example of this pattern is Google’s experimental AI framework Opal [10], introduced in 2025, that automatically creates “mini-apps” from natural language descriptions. Users specify what they want to accomplish, and Opal generates and executes the necessary chain of AI services, effectively producing the integration logic and API calls on the behalf of users.

For example, given the prompt *“Take a URL to a video and convert it into a blog post with two illustrative images, a short promotional trailer, and an audio podcast summary*,” Opal responds in seconds by autonomously assembling a workflow of AI models that perform the tasks shown in Figure 3 and described below (see [11] for a video walkthrough of this example):

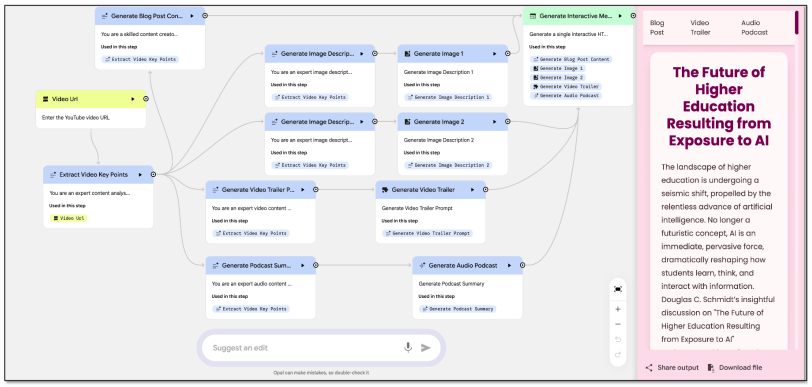

Figure 3: Creating a Multimedia Package Mini-App with Opal.

- *Analyze* the input video (transcribing it and extracting key points);
- *Invoke* an LLM to write a well-structured blog article summarizing those points;
- *Apply* an image generation model to create two relevant illustrations for the blog;
- *Generate* a script for a trailer and uses a video generation model to produce a short promo video;
- *Employ* a text-to-speech model to create an audio summary (podcast); and
- *Compile* all these results into a simple webpage or interactive report for the user to consume (text with embedded images, video, and audio).

*Agentic orchestration in action***.** The power of the *Dynamic Intent-to-Workflow Orchestration* pattern lies in chaining intelligent components on-the-fly. Opal orchestrates multiple AI agents, each specialized for tasks like text

generation, image synthesis, video creation, or speech production. Previously, achieving this outcome required multiple tools and custom scripts. Opal removes that friction by compiling user intent directly into an executable workflow. This approach enables on-demand 'app assembly' from natural language, instead of relying on predefined templates.

This approach moves beyond earlier no-code and low-code platforms built on deterministic, template-driven integrations. While suitable for regulated processes, agentic assembly excels when goals evolve. Instead of requiring users to predefine actions and data mappings, orchestration frameworks like Opal synthesize new workflows dynamically using AI reasoning. The result is dynamic, cross-modal integration that's automated. This approach aligns with the emerging vision of an 'Internet of Agents' [12], where autonomous agents can discover services and collaborate to fulfill high-level user intents at Internet scale.

*Systems challenges and oversight***.** Dynamic, AI-generated workflows introduce new engineering challenges. For example, workflows are ephemeral and may differ across executions, so debugging, observability, and reproducibility become harder than in traditional software systems. Platforms like Opal should provide execution logs and visualizations so users can understand what steps were taken and intervene when necessary. Versioning also becomes non-trivial since organizations that rely on AI-generated mini-apps must be able to reproduce or audit behavior even as underlying models and prompts evolve. Although *no human explicitly codes these workflows*, engineers must still design tooling for testing, monitoring, and governance.

Early use also highlights the need for human oversight. In practice, some outputs—particularly generative media—may be imperfect or misaligned with user expectations, requiring review and refinement. Users often become editors and curators of AI-generated results, adjusting prompts or outputs *post hoc*. Nonetheless, the speed and breadth of output from a single prompt remain striking. Projects spanning multiple formats that once required extensive labor can now be bootstrapped almost instantly.

*Internet-scale impact***.** As platforms like Opal become widespread, they will hasten the growth of AI-generated mini-apps and custom automations. Individuals and small teams can create tailored integrations on demand, bypassing traditional development cycles. These trends raise important questions about quality control, security, and governance. Such concerns are becoming particularly acute as non-developers gain the ability to deploy workflows that interact with sensitive systems. Managing the scope and downstream impact of these AI-created workflows may require new review mechanisms, repositories, or marketplaces where workflows can be shared and vetted.

At the platform level, this trend points toward intensified competition. Opal's deep integration with Google's ecosystem enables a seamless, end-to-end agentic experience, while competitors such as Microsoft or OpenAI are likely to pursue analogous strategies within their own AI platforms. It remains to be seen if these agentic workflow systems will converge via open standards or fragment into siloed ecosystems. The outcome will determine whether the Internet evolves toward a unified 'agentic OS' layer, or instead into vendor-locked silos.

In summary, the *Dynamic Intent-to-Workflow Orchestration* pattern enables users to "program" the Internet by stating intent, with AI handling the translation into executable action. While this pattern democratizes automation and integration, it also shifts complexity from code authoring to governance, observability, and control. The challenge—as well as the opportunity—is to harness this productivity while ensuring that AI-assembled workflows remain transparent, debuggable, and secure.

Implications for Internet Computing Practice

The rise of generative AI and intent-driven systems is poised to reshape the daily practice of Internet technologists, architects, and developers. As shown in Figure 4, the locus of effort is shifting from writing and debugging code toward expressing intent, orchestrating AI-driven workflows, and verifying outcomes. Beyond the two

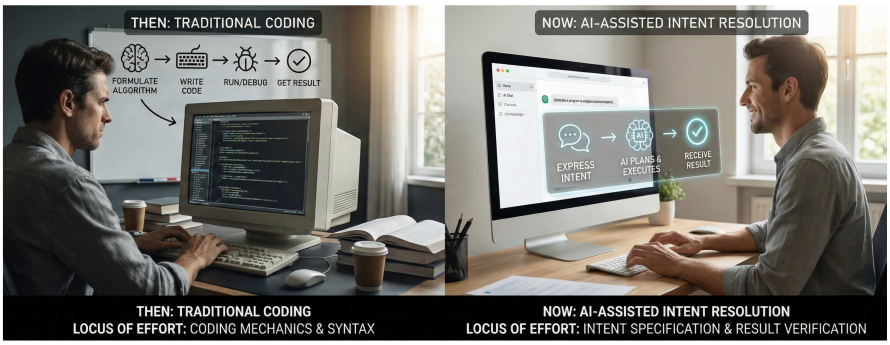

Figure 4: Internet Computing: the Past vs. the Future.

architectural patterns discussed earlier, this transition brings broader changes in how systems are designed, how teams are organized, and what standards, tools, and mental models are required.

*System design priorities are shifting from APIs to intents***.** Where Internet systems once centered on carefully defined endpoints and fixed workflows, intent-driven architectures expose a limitation of rigid APIs: they assume short-lived, deterministic interactions with known semantics. As AI agents increasingly assemble workflows dynamically, future platforms must instead expose higher-level capabilities (e.g., advertised with semantic descriptions, constraints, and policies) that agents can discover, negotiate, and compose at runtime. Rather than invoking a sequence of rigid API calls, agents may interact through natural-language interfaces or service-level contracts that determine *what* outcome is desired rather than *how* it is achieved.

In turn, this approach replaces static user workflows with adaptive planning, where intermediary AI agents reconfigure steps dynamically to meet user intent. Designing for this model requires systems that tolerate variability, support flexible orchestration, and enable agent improvisation. Together, these capabilities transform applications into platforms built for dynamic, intent-driven interaction.

*The composition of development teams will also evolve***.** As AI becomes a core execution layer, new roles are emerging. As shown by the exemplars above, *AI workflow architects* are needed to bridge business intent and AI capability. Likewise, *agent governance engineers* focus on guardrails, monitoring, and incident response when AI deviates from expected behavior. These roles extend today's nascent positions into specialized operational disciplines.

Traditional developers remain essential for building infrastructure, interfaces, and integrations, but they will increasingly collaborate with AI specialists. Likewise, developers' own workflows are also evolving. Coding is becoming a human–AI partnership, where engineers increasingly spend more time guiding, reviewing, and validating AI-generated code than programming software from scratch [2].

Internet platforms and services will need to adapt. Search engines are evolving from presenting result lists into acting as answer- and task-oriented agents. This evolution blurs the boundary between search, browsers, and personal assistants since systems can perform follow-up actions (e.g., booking trips and scheduling meetings) on the user's behalf. Similarly, enterprise Software-as-a-Service (SaaS) platforms are evolving beyond static APIs toward intent-aware services, where embedded AI copilots act as front ends to deeper architectural changes, e.g., systems that accept goals rather than commands, negotiate execution plans, and enforce policy, cost, and security constraints during execution. Users will specify desired outcomes while the AI handles the underlying workflows, which places new demands on those platforms' APIs and orchestration hooks.

This shift is not just about more intelligent clients. While systems like Deep Research and Opal can be implemented today by orchestrating existing APIs, they increasingly stress assumptions embedded in Internet platforms. Rigid API contracts presume fixed operations, stable schemas, and deterministic behavior. In contrast, intent-driven workflows require platforms to discover capabilities, negotiate execution strategies, adapt during long-running tasks, and expose cost, trust, and policy constraints. Internet systems must therefore evolve from passive endpoints into active participants in goal satisfaction.

Developers need to adapt as their tools are augmented by AI. Interactive development environments, code review systems, testing frameworks, and even deployment pipelines are integrating AI assistance across the software lifecycle, from code generation and bug-fixing suggestions to automated documentation and environment configuration. Software development practice is evolving into the composition of AI-driven workflows combined with targeted manual coding. The skill emphasis will shift toward leveraging AI effectively and applying human judgment in areas where precision and oversight are crucial.

*New standards, benchmarks, and shared practices will be essential***.** Just as earlier paradigm shifts from web services to cloud computing and mobile platforms gave rise to new standards, AI will require its own set of standards, patterns, and best practices. The Model Context Protocol (MCP) [13] – an early effort to standardize how AI-enabled tools and models invoke each other – points toward standardized tool invocation, but gaps remain in agent-to-agent communication and cross-vendor interoperability. Equally important is evaluation: current benchmarks focus on static tasks [14], while agentic workflows involve planning, tool use, and dynamic adaptation.

The R&D community in Internet computing needs shared benchmarks that can assess end-to-end agent behavior and performance in dynamic scenarios. In parallel, new mental models and patterns should be codified and disseminated. Thinking in terms of prompts, examples, and non-deterministic behavior becomes as important as functions and classes. Educators and professional communities will play a key role in codifying and spreading these best practices.

Outlook: Engineering the Next Layer of the Internet Stack

We stand at an inflection point comparable to the early Web or the rise of cloud computing. Generative AI is emerging as an intent-centric layer within Internet computing. As this layer solidifies, the choices we make now will determine whether it becomes robust and interoperable rather than brittle and siloed.

History offers clear lessons. While the early Internet was undeniably transformative, it also embodied costly oversights: security was retrofitted rather than foundational, transparency was neglected, and platform power consolidated in ways that later proved hard to unwind. As AI agents become active participants in our systems, we cannot afford to repeat these mistakes. Therefore:

- Security must be integrated from the outset through clear governance and guardrails,
- Transparency must ensure that AI decisions and data usage are visible and auditable, and
- Openness must be preserved through standards and interoperability rather than closed, vendor-specific tools.

Encouragingly, the rapid adoption of MCP suggests a willingness to collaborate, but this momentum must extend to safety frameworks, auditability, and accountability.

Moreover, the opportunity enabled by AI is profound. If designed thoughtfully, these new architectural patterns enable agentic Internet systems that are observable, governable, and trustworthy by design. Individuals—technical or not—can apply personal AI agents to interact seamlessly with digital services, with clear consent, traceability, and control. Routine and complex tasks could be handled rapidly, freeing human attention for creative and strategic work. Just as the Web democratized publishing, generative AI is poised to democratize computational thinking [15]. It allows scientists, educators, and domain experts to create new tools and workflows without traditional coding.

Realizing this vision requires clear principles. Robustness ensures AI-driven systems degrade gracefully rather than fail catastrophically. Transparency gives users insight into what their AI systems do and why. Human-centered design keeps intent, values, and oversight at the core. The technical community, including professional bodies like the IEEE, has a critical role to play in codifying these principles into standards and best practices, much as it did for software engineering in earlier eras.

Ultimately, the future of Internet computing will be shaped less by who writes code than by who designs the systems that translate human intent into safe, scalable action. Software engineering is shifting its focus from the production of low-level implementations to the architecture and governance of intent-driven systems [16]. As development becomes increasingly AI-mediated, humans must continue to articulate goals and exercise judgment, while AI systems determine and execute the means to achieve them. In this model, programming evolves into the *orchestration of intelligence*—human and artificial—across distributed networks.

Our responsibility is to design the digital highways and rules of the road that let people and autonomous agents operate together safely and productively. When we succeed, computing recedes into the background, freeing human attention for ideas, judgment, and problem-solving. Like the power grid—largely invisible yet essential—intent-driven computing must be dependable and trusted, preserving the very qualities that made the Internet transformative in the first place.

References

[1] J. M. Wing, “Computational thinking,” *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, Mar. 2006.

[2] A. Carleton *et al.*, *Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research and Development*. Pittsburgh, PA, USA: CMU/SEI Press, Nov. 2021.

[3] A. E. Hassan *et al.*, “Towards AI-native software engineering (SE 3.0): A vision and challenge roadmap,” *arXiv preprint* arXiv:2410.06107, Oct. 2024.

[4] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt, “ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design,” in *Generative AI for Effective Software Development*, A. N. Duc, P. Abrahamsson, and F. Khomh, Eds. Cham, Switzerland: Springer Nature, 2024.

[5] OpenAI, “Introducing deep research in ChatGPT: A new agentic capability for multi-step research,” *OpenAI News*, Feb. 2, 2025. Available: https://openai.com/index/introducing-deep-research.

[6] Reuters, “OpenAI launches new AI tool to facilitate research tasks,” Feb. 3, 2025. Available: https://www.reuters.com/technology/openai-launches-new-ai-tool-facilitate-research-tasks-2025-02-03.

[7] D. C. Schmidt, “Unbinding Prometheus: How Generative AI is Liberating Computational Thinking from Coding (part 1 of 8),” *YouTube*, video, 2025. Available: https://youtu.be/fv_44TE8xxM. Accessed: Dec. 29, 2025.

[8] R. Ghose *et al.*, “Agentic AI: Finance and the ‘do it for me’ economy,” *Citi Global Perspectives & Solutions*, Jan. 2025. Available: https://www.citi.com/citigps/agentic-ai-finance-do-it-for-me-economy.

[9] S. Johnson and D. Hyland-Wood, “A primer on large language models and their limitations,” *arXiv preprint* arXiv:2412.04503, 2024. Available: https://arxiv.org/abs/2412.04503.

[10] T. Tan, “Google’s Opal changes the game for agents and automation,” *The Strange Review*, Jul. 30, 2025. Available: https://thestrangereview.substack.com.

[11] D. C. Schmidt, “Unbinding Prometheus: How Generative AI is Liberating Computational Thinking from Coding (part 5 of 8),” *YouTube*, video, 2025. [Online]. Available: https://youtu.be/px82h-7tVeA. Accessed: Dec. 29, 2025.

[12] Y. Wang *et al.*, “Internet of agents: Fundamentals, applications, and challenges,” *IEEE Trans. Cognitive Communications and Networking*, Oct. 2025, doi: 10.1109/TCCN.2025.3623369.

[13] Ars Technica, “MCP: The new ‘USB-C for AI’ that’s bringing fierce rivals together,” Apr. 1, 2025. Available: https://arstechnica.com/information-technology/2025/04/mcp-the-new-usb-c-for-ai-thats-bringing-fierce-rivals-together.

[14] X. Liu *et al.*, “AgentBench: Evaluating LLMs as agents,” in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2024. Available: https://arxiv.org/abs/2308.03688.

[15] Douglas C. Schmidt, ``The Coming Commoditization of Computational Thinking,'' Communications of the ACM, 2026 (to appear).

[16] V. Terragni, A. Vella, P. Roop, and K. Blincoe, “The future of AI-driven software engineering,” *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 5, Art. no. 120, Jun. 2025, doi: 10.1145/3715003.

Author Bio

Douglas C. Schmidt is the Dean of William & Mary’s School of Computing, Data Sciences & Physics, after spending over 20 years as a computer science professor at Vanderbilt University. He recently served as the Pentagon’s Director, Operational Test & Evaluation and was a program manager at the Defense Advanced Research Projects Agency and the Chief Technology Officer at Carnegie Mellon University’s Software Engineering Institute. He has published many papers and books on software-related topics.