

A Two-Week Sprint for Knowledge

From Agile to Design Thinking

Jessica Talisman, MLS

Apr 21, 2026 · Intentional Arrangement

In 1986, Hirotaka Takeuchi and Ikujiro Nonaka published “The New New Product Development Game” in the *Harvard Business Review*, describing how Honda, Canon, and Fuji-Xerox organized teams to build copiers and automobiles. Jeff Sutherland read that article, combined it with lean manufacturing principles and object oriented programming, and in 1993 adapted it for software development at Easel Corporation. Ken Schwaber formalized the method at the OOPSLA conference in 1995. By 2001, seventeen software developers gathered at a ski lodge in Snowbird, Utah, and signed the Agile Manifesto, declaring that they were “uncovering better ways of developing software.” Not knowledge or meaning. A manufacturing framework for software.

The Agile Manifesto’s own language is explicit about its scope, yet the industry has applied Agile methodology to virtually everything in a technological environment — including knowledge work, semantic systems and now artificial intelligence. And within the same breath, organizations and experts regale the industry with talk of innovation and the emergence of a new paradigm in ways of working. Yet those ways of working are locked inside of Agile methodologies and Scrum, designed for the factory floor and the codebase — not for the messy, recursive, deeply human work of making meaning out of complexity.

Scrum Is a Manufacturing Schedule

Let’s return to the Takeuchi and Nonaka paper, where the title — “The New New Product Development Game” — describes how product teams at Japanese firms developed physical goods such as the FX-3500 copier and the Honda City car. They draw on the rugby metaphor because the team “tries to go the distance as a unit, passing the ball back and forth.” The unit of delivery is a working prototype of a material artifact. The feedback loop is a machine that works or fails.

Sutherland’s 2014 book, *Scrum: The Art of Doing Twice the Work in Half the Time*, reinforces this with military and manufacturing case studies such as the F-86 fighter production and Toyota. Sutherland’s book rebuilds the argument with the same assumption — that the thing being built has discrete, inspectable, testable parts that, when finished, are finished.

The 2020 Scrum Guide defines the Increment as “a concrete stepping stone toward the Product Goal,” one that is “additive to all prior Increments and thoroughly verified, ensuring that all Increments work together.” These words are sensible if you are building a mobile application, with a login screen and a settings page. But for knowledge systems, these words are destructive, because the components of a semantic architecture are not additive in the way Scrum dictates. A controlled vocabulary is not a subcomponent of a taxonomy the way a login screen is a subcomponent of an app. A taxonomy without disciplined, vocabulary control as its underpinnings does not function well, if at all. You cannot ship half a taxonomy and call it an Increment.

You see, Scrum assumes tame problems that can be time-boxed, estimated with planning poker, decomposed into user stories, tracked on a velocity chart and declared done against a checklist. If only knowledge were this tame. Knowledge organization is anything but tame. More than fifty years old, in 1973 Horst Rittel and Melvin Webber warned that applying the engineering paradigm uniformly to all problems was “bound to fail.” And yet here we are, applying a manufacturing schedule and methodologies to knowledge, expecting that knowledge will fit into the confines of old ways of working — an artifact of the Third Industrial Revolution imposed on the demands of the Fourth.

In 2016, Klaus Schwab wrote about this historical pattern. The First Industrial Revolution, beginning around 1760, gave us steam and mechanical production. The Second, in the late nineteenth century, gave us electricity and the assembly line — the world that produced Toyota, lean manufacturing, and the production logic that Scrum still carries as its core ethos. The Third, beginning in the 1960s, gave us semiconductors, mainframe computing, and eventually the internet — the world in which Agile was born, by software developers, for software development. The Fourth Industrial Revolution, Schwab argued, is fundamentally different, in kind, as it is defined by the fusion of artificial intelligence, cyber-physical systems and knowledge technologies that blur the boundaries between the physical, digital and biological worlds.

The problems of this era are not necessarily engineering problems with stable requirements. We are wrestling with problems such as how to ground machine intelligence in meaning, how to organize evolving knowledge and how to build systems that can explain themselves. These are wicked problems that demand new epistemologies and frameworks for problem solving. Agile may have been the right methodology for the Third Industrial Revolution but we are no longer in the Third Industrial Revolution. The frameworks have not caught up, and the cost of that lag is exacerbated every time an organization forces Third Industrial Revolution principles applied to Fourth Industrial Revolution work.

In other words, this is not a product problem or a platform problem — it is learning to embrace new ways of working. This is an infrastructure problem that is governed by epistemology — a fundamental philosophy pertaining to knowledge.

Why Knowledge Cannot Be Sprinted

An ontology is, per Thomas Gruber’s foundational definition, “a specification of a conceptualization.” A conceptualization is a whole, not parts of a whole. Specifying half of it produces logical incoherence, not a minimally viable product. For example, ANSI/NISO Z39.19, the standard for constructing controlled vocabularies, requires resolving synonymy, homonymy and scope before terms enter production, because any inconsistency propagates downstream into every subsequent use. SKOS, the W3C standard for encoding knowledge organization systems, requires that hierarchical relationships form coherent structures with transitive closure and OWL 2 requires that class axioms be satisfiable. These are preconditions, necessary for logical systems to reason and express meaning.

In practice, I have watched organizations attempt Agile delivery of knowledge systems in two patterns and neither ends well. In the first, the team holds stand-ups, ceremonies and retrospectives. The taxonomist brings tickets and adds fifteen terms to the product vocabulary. Tickets close and a velocity chart goes up. Nothing is integrated, because integration is the hard part and integration does not fit into two weeks so that goes to the backlog. Six months later, the vocabulary is a collection of inconsistently scoped terms, the ontology has orphan classes and the knowledge graph — if it exists — cannot reliably answer queries. But the burndown chart is immaculate and leadership is blindly satisfied because boxes were checked.

In the second pattern, the team borrows from other methodologies while still calling it Agile. The taxonomist conducts multi-week domain analysis and stakeholder interviews — that is ethnography. She does card sorting and concept modeling — that is design research. She iterates with subject matter experts — that is participatory design. She ships, eventually, a coherent model. The sprint cadence was irrelevant to the work. Martin Fowler named this phenomenon in his 2018 keynote: “faux-agile” — the Agile Industrial Complex.

Ron Jeffries, another original signatory of the Agile Manifesto, told developers the same year, 2018, to abandon the word entirely. Dave Thomas declared in 2014 that “Agile” had become “a magnet for anyone with points to espouse, hours to bill or products to sell.” When three of the seventeen authors of a framework say it has been hollowed out, one has to wonder if Agile is appropriate for all technology work and if it is time to revisit ways of working.

Innovation Does Not Fit in a Backlog

The deeper issue is that Agile and Scrum were never designed to inspire or foster innovation. They were designed to deliver known requirements with increasing efficiency. The Scrum Guide’s entire apparatus — product backlog, sprint planning, Definition of Done — presupposes that the problem has been formulated by a product owner who can write it in a user story. But knowledge work, like design, is wicked. Rittel and Webber’s first property of a wicked problem is that “There is no definitive formulation.” The second

property states that there is no stopping rule. And the fifth property illuminates that every solution is a “one-shot operation” because there is no opportunity to learn by trial-and-error without consequence.

Given the nature of wicked problems, it would seem that Agile is ill fitted for wicked problems. And for that matter, ill fitted for the dynamic nature of knowledge, where knowledge infrastructures are never one-shot operations, necessitate stopping rules and without a doubt defy definitive formulations.

Therefore, knowledge organization in an enterprise domain is wicked in exactly this sense. The problem of how to represent “customer” in a telecommunications company’s semantic model has no pre-existing correct answer. Its formulation depends on who is doing what with the representation, and that changes as you build. The scope of the vocabulary shifts as you interview stakeholders. The taxonomy reorganizes as you encounter edge cases. The ontology’s axioms tighten or loosen as the reasoner surfaces contradictions. Don’t be mistaken — this does not bend to the Agile principle of welcoming changing requirements, as the requirements are constituted *through* the work. The problem is under construction, often evolving, during the solving.

Organizations that manage all work through Scrum are, in effect, optimizing for predictability in domains that demand exploration. The results resist innovation and take on the appearance of productivity. Velocity charts rise while the underlying intellectual architecture is flattened into bite-sized deployments, uncoupled from the continuum that is knowledge.

Design Thinking Belongs in the Swamp

Donald Schön drew the distinction that “In the varied topography of professional practice, there is a high, hard ground where practitioners can make effective use of research-based theory and technique, and there is a swampy lowland where situations are confusing ‘messes’ incapable of technical solution.” Agile lives on the high ground while knowledge work lives in the swamp. Schön’s term for how competent practitioners navigate the swamp was “reflection-in-action” — the iterative, tacit recalibration that happens while the work is happening.

Design thinking, in the lineage of Herbert Simon, Richard Buchanan and Schön, is the methodology of the swamp. It assumes the problem is under construction. Its five phases — empathize, define, ideate, prototype, test — are not a linear pipeline but a recursive, iterative process that expects the definition of the problem to change as understanding deepens.

Research in the *Handbook of Human-Centered Artificial Intelligence* confirms this alignment — integrating design thinking into AI development makes systems more user-focused, iterative and impactful, emphasizing data governance, transparency, explainability and bias mitigation. Carnegie Mellon University’s Tepper School of Business now teaches a dedicated program on design thinking with AI, arguing that the combi-

nation addresses “complex, systemic problems” that linear frameworks cannot manage successfully. IBM’s enterprise design thinking practice, developed over two decades, has been applied across client engagements precisely because it keeps human experience at the center while technology evolves underneath — design thinking is “timeless,” while technology is “timestamped.”

A 2026 study in *Scientific Reports* examined the relationship between higher order thinking, generative AI chatbot use and engineering creativity, finding that higher order thinking had significantly greater importance for creative outcomes than AI tool use alone — reinforcing that cognitive depth, not tool velocity, drives innovation. Research in the *Proceedings of the Design Society* further argues that generative AI is shifting the designer’s role from executor to paradigm-setter, making design thinking more essential as AI handles production tasks while humans must guide framing, ethics and meaning.

To be clear, these frameworks are not different flavors of the same methodology. Scrum and design thinking represent totally different epistemologies. Scrum assumes the problem was handed to you while design thinking assumes the problem must be found and wrestled with.

Is AI a Knowledge Tool?

This brings us to the question the industry avoids and I, for one, have been itching to ask. Is AI a knowledge tool? How do you classify it?

A large language model generates plausible sequences of text. Returning to the Bender, Gebru, McMillan-Major, and Mitchell paper mentioned earlier, the authors defined a language model as a system for stitching together linguistic forms according to probabilistic patterns, “without any reference to meaning.” Without reference to meaning, as LLMs out-of-the-box are absent of external grounding. They optimize for plausibility, not truth and therefore as a standalone, can be best categorized as an information-processing tool.

It becomes a knowledge tool only when grounded in a semantic layer — controlled vocabularies, taxonomies, ontologies, knowledge graphs — that provides reference and provenance. The industry’s own corrective measures and associated marketplace confirms this as fact. Retrieval-augmented generation, knowledge graphs for LLMs and what recent discourse calls “context engineering” are, underneath the packaging, the semantic infrastructure that should have been built first. But it was bolted on after the fact to prevent hallucination that was pretty much guaranteed the moment the model was deployed without one.

The RAG paper from 2020 called out this problem, identifying that parametric memory is opaque and cannot be updated and that retrieval from structured sources is how provenance is captured and encoded. But it seems the industry has a memory problem or it was too early for most to hear the message.

Ironically, provenance is a core element of knowledge management and ontology work, as provenance is a principled tenant of the discipline and problem space. Without provenance, knowledge cannot exist. Organizations are only now discovering they need it, because their generative systems cannot explain themselves. Call it lineage, call it traces. But if you are building for knowledge, call it provenance, as provenance includes traces, lineage, citations, the temporal and the spatial.

So if generative AI is an information processing tool, I posture that for AI to become a knowledge tool, there must be provenance and descriptive logic. And to be frank, nobody sprints their way into provenance — it is architected and sustained.

Knowledge Is a Spiral, Not a Sprint

The irony does not escape the moment. The same Nonaka who co-authored the 1986 paper that inspired Scrum went on to develop the SECI model of knowledge creation in 1995, describing knowledge as a spiral between tacit and explicit forms, moving through socialization, externalization, combination and internalization.

The spiral does not have an end, as is true with spirals, emblematic of life. Each rotation creates new knowledge that becomes the substrate for the next rotation. There is no Definition of Done because the definition of what is being done is itself being constructed. Scrum took the team structure from Nonaka's first paper and discarded his next decade of work because it did not fit neatly with Scrum. The mechanisms of knowledge work, as Nonaka spent a decade arguing, are metaphor, tacit skill, apprenticeship, shared space and the patient conversion of hunches into models and models back into practice.

The Ontology Pipeline™ mirrors this spiral. You start with a controlled vocabulary — a disciplined process involving agreements as to what terms mean. You mature it into a taxonomy by introducing hierarchy, structuring terms into parent-child relations. Thereafter, the taxonomy is matured into a thesaurus by adding associative relationships and synonymy, encoded in SKOS. The thesaurus is baseline work from which an ontology is developed, adding formal semantics using OWL and RDF. You architect a knowledge graph, assembling components to include the knowledge assets developed through the Ontology Pipeline™ iterative framework.

Each stage prepares the ground for the next and stages cannot be skipped or bypassed. You cannot ship the knowledge graph without an ontology, and the ontology must be logically consistent and coherent to make sense of data. A knowledge graph without vocabulary control risks becoming a confident liar or failing to work with natural language. Because knowledge assets are dimensionally interconnected, knowledge is indeed a spiral, unable to be contained by manufacturing logic.

The Pattern and the Cure

Organizations budget for platforms — Confluence, SharePoint, LLM subscriptions, vector databases, a RAG vendor. But they do not budget for descriptive logic and knowledge,

perhaps because the problem is not solved by a single tool and is seemingly too amorphous in an Agile, data-centric world. The controlled vocabulary is cut unless it delivers value for the database. Often, the taxonomy is cut because it does not demo well and its value sits in the plumbing, delivering its punch over time. The result is an ever-growing pile of documents with a chatbot stapled to the front, confidently answering questions in the voice of a system that has no reference for any of the terms sprinkled throughout the documents.

Then the system hallucinates. It contradicts itself across sessions. It surfaces outdated policy as current and surfaces private Outlook messages from deactivated employees. I have seen this in real life more times than I can count. And the optics are worse when a vendor tool is responsible for revealing private chats and email transactions. Usually the executive sponsor asks why the investment has not worked, ignoring the evidence of faux pas such as the surfacing of private communications. All in a day's work, when burndown charts and velocity set the precedent for ways of working and reward systems.

Executives ignore the fact that the system was never grounded. It was never grounded because grounding requires the slow, expensive, disciplined work of knowledge organization — work that does not fit into a sprint, does not demo well and fails to produce a velocity chart. The organization chooses to bypass investment in the descriptive knowledge apparatus because the lines are too blurry to clearly define according to Agile speak. Because knowledge is infrastructure, not a product or a platform. Knowledge is the gas in the tank, the fuel for meaning, the material that justifies logical reasoning.

Design thinking gives that knowledge work a home. It is comfortable with ambiguity. It treats problem discovery as an equal to problem solving. It certainly does not demand a shippable increment every fourteen days. Instead, design thinking demands that you understand what you are building before you build it — and that you keep revising that understanding as the work proceeds. Library and information science, with over a century of standards for knowledge organization, provides the methods while design thinking provides the epistemology. Together, they offer what Agile never could — a framework suited to the nature of knowledge itself.

A two-week sprint to deliver what? If you are building knowledge systems, two weeks will get you nothing coherent. Knowledge is not built fast and broken into shape. It is built carefully, with provenance, governance and patience. Knowledge is not a product to be manufactured and no — you cannot buy off-the-shelf knowledge that perfectly matches a domain or organization. Knowledge is nuanced, specific to each organization, domain and person.

In the spirit of VC-speak, knowledge is the moat, and that moat is not Agile.

Jessica Talisman, MLS is a Semantic Engineer, Information Architect, and knowledge infrastructure strategist with more than 25 years of experience. She is the founder of the Ontology Pipeline™ and publishes *Intentional Arrangement* on Substack.