



MASTERARBEIT | MASTER'S THESIS

Titel | Title

From Imagination to Creation? How Large Language Models
Change Maker Practice

verfasst von | submitted by
Celina Mueller B.A.

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of
Master of Science (MSc)

Wien | Vienna, 2026

Studienkennzahl lt. Studienblatt | Degree
programme code as it appears on the
student record sheet:

UA 066 013

Studienrichtung lt. Studienblatt | Degree
programme as it appears on the student
record sheet:

Masterstudium Joint-Degree Middle European
interdisc. master prog. in Cognitive Science

Betreut von | Supervisor:

Tomás Gál Ph.D.

Abstract

Making, turning an idea into an artifact, requires bridging the gap between imagination and creation. That gap has cognitive roots: working memory constraints, element interactivity, and the need for domain-specific schemas. Large language models (LLMs) are a new kind of tool that works through explanation and instruction in natural language, where domain schemas are built or fail to form. Making, post-2010 Cognitive Load Theory (CLT), and LLMs have not been examined together. This thesis asks how LLMs change the cognitive load of making, and whether they reduce it or only change its form.

The central claim is that LLMs redistribute cognitive load rather than reduce it. The maker's problem-solving remains the maker's. Load once concentrated in the domain task would spread across that task and the interaction task. The thesis introduces the cascade of translations, which maps what making demands from intention to artifact across six steps, and traces how load could shift at each step under an LLM. It identifies six load-type constructs: verification load, modal gap load, language bottleneck, prompt formulation load, context management load, and intent monitoring. It proposes the engagement criterion, on which engagement with the conceptual structure of the problem, not struggle or task completion, is the relevant variable for schema formation. The redistribution would not fall evenly. Verification load likely concentrates where schemas are thinnest, placing the heaviest burden on novices. If redistribution holds, what happens within the interaction carries the explanatory weight, so comparing LLM use to non-use or another tool studies the wrong contrast.

This is a theoretical contribution. The redistribution claim is evaluated by its coherence with post-2010 CLT and its fit with practitioner accounts from online forums and blogs. Empirical testing remains open.

Keywords: cognitive load theory, making, large language models, element interactivity, redistribution, schema formation

Kurzfassung

Making, eine Idee in ein Artefakt zu überführen, erfordert die Überbrückung der Distanz zwischen Vorstellung und Ausführung. Diese Distanz hat kognitive Wurzeln: Einschränkungen des Working Memory, Element Interactivity und der Bedarf an domänenspezifischen Schemas. Large Language Models (LLMs) sind ein neuartiges Tool, das durch Erklärung und Instruktion in natürlicher Sprache wirkt, also dort, wo domänenspezifische Schemas aufgebaut werden oder nicht entstehen. Making, post-2010 Cognitive Load Theory (CLT) und LLMs wurden bisher nicht gemeinsam untersucht. Diese Arbeit fragt, wie LLMs den Cognitive Load im Making verändern und ob sie ihn reduzieren oder nur seine Form ändern.

Die zentrale These lautet, dass LLMs Cognitive Load umverteilen, anstatt ihn zu reduzieren. Das Problemlösen bleibt beim Maker. Load, der zuvor in der Domänenaufgabe konzentriert war, würde sich auf diese und die Interaktionsaufgabe verteilen. Die Arbeit konzeptualisiert die Cascade of Translations, die abbildet, was Making von der Intention bis zum Artefakt über sechs Schritte verlangt, und verfolgt, wie sich Cognitive Load an jedem Schritt verschieben könnte, wenn ein LLM beteiligt ist. Sechs Load-Typen werden vorgeschlagen: Verification Load, Modal Gap Load, Language Bottleneck, Prompt Formulation Load, Context Management Load und Intent Monitoring. Die Arbeit formuliert das Engagement Criterion: Die Auseinandersetzung mit der konzeptuellen Struktur des Problems, nicht Anstrengung oder Aufgabenabschluss, ist die relevante Variable für Schema Formation. Die Umverteilung würde nicht gleichmäßig ausfallen. Verification Load konzentriert sich wahrscheinlich dort, wo Schemas am wenigsten vorhanden sind, und belastet damit Novizen am stärksten. Wenn die Umverteilung zutrifft, liegt die Erklärungskraft in dem, was innerhalb der Interaktion geschieht, und ein Vergleich von LLM-Einsatz mit Nicht-Einsatz oder einem anderen Tool untersucht den falschen Kontrast.

Die Arbeit ist ein theoretischer Beitrag. Die Umverteilungsthese wird an ihrer Kohärenz mit post-2010 CLT und ihrer Passung zu Praxisberichten aus Online-Foren und Blogs gemessen. Empirische Überprüfung bleibt offen.

Contents

Abstract	i
Kurzfassung	ii
Acknowledgments	v
GenAI Disclosure	vi
1 Introduction	1
2 Conceptual Foundations	5
2.1 Making	5
2.2 Cognitive Load Theory	9
2.3 Cognitive Load Theory and maker education	13
2.3.1 Cognitive load theory in maker education	13
2.3.2 Cognitive demands in maker education research	15
2.3.3 Historical lag and signs of convergence	17
2.4 Large language models	18
2.4.1 Technical foundations	18
2.4.2 Interaction affordances	21
2.4.3 Prompting	21
2.4.4 Agentic capabilities	22
2.4.5 Limitations	22
2.5 Large language models and making	24
2.5.1 Physical computing	25
2.5.2 3D printing and additive manufacturing	26
2.5.3 Computer-aided design	27
2.5.4 Craft and textiles	28
2.5.5 Design and manufacturing education	28
2.5.6 Constructionist perspectives	29
2.5.7 Across domains	30
2.6 LLMs and Cognitive Load Theory	30
2.7 Making, LLMs, and Cognitive Load Theory	34
3 Synthesis	37
3.1 The cognitive load of making	37

3.1.1	Element interactivity and schemas in maker practice	38
3.1.2	CLT effects in maker practice	40
3.1.3	The cascade of translations	41
3.1.4	Extending CLT: four aspects making requires	43
3.2	LLM mediation: redistribution, not reduction	45
3.2.1	Where LLMs enter the cascade	45
3.2.2	The redistribution claim	52
3.2.3	Learning implications	53
3.3	Methodological implications	55
3.3.1	The methodological claim	55
3.3.2	Falsifying the constructs	58
3.3.3	What a study of LLM-mediated making would have to capture . . .	59
4	Tool in Use	61
4.1	Method	61
4.2	The constructs in cases	64
4.2.1	Distribution of accounts across making domains	64
4.2.2	Verification load	64
4.2.3	The cascade of translations and modal gap	67
4.2.4	Cognitive load redistribution	69
4.2.5	The engagement criterion and the independence fallacy	72
5	Discussion	77
5.1	Epistemic calibration	77
5.2	Authorial reflection and critical landscape	79
5.3	Framework limitations	81
5.4	Future work	82
6	Conclusion	84
	References	86
A	Visualizations	100
A.1	Field Intersection Diagram	100
A.2	Cascade of Translations	101
A.3	Six Load Types	103
B	AI Tooling Setup	105
B.1	Literature Conversion Pipeline	105
B.2	Custom Skills	106

Acknowledgments

I thank Tomáš Gál for supervising this thesis with an open mind and support at every stage. His feedback and discussions helped me refine its direction. My students have been an endless source of inspiration, and teaching them shaped much of what follows. I thank Dali, for his infinite patience with me.

GenAI Disclosure

I used Claude Code, Anthropic’s command-line coding tool, throughout the preparation, organization, and revision of the thesis. This section describes the setup to make the tool’s role and limits transparent.

I ran the tool with a project instruction file, `CLAUDE.md`, that I wrote and maintained. It contained the chapter map and status, the research question and hypothesis, my original contributions, the theoretical framework, the voice and style rules, and the file conventions. The file gave the tool persistent context, so this information did not have to be re-explained at every session.

The tool’s work took three forms. First, executive function and project management support. It helped me set up the folder structure, track the to-do list across the finalization phases, and keep notes, drafts, and literature organized. Second, literature processing. I conducted the literature search and selected every source myself. I converted source PDFs to markdown with a local pipeline so the tool could read full extractions rather than abstracts. The conversion script is reproduced in Appendix B. No paper was cited without my reading its extraction first. Third, revision support. The thesis’s arguments and original contributions are my own. I used the tool to check whether theoretical concepts were stated correctly, to surface gaps and counterarguments in passages I had already written, and to clarify wording. It did not generate the thesis’s claims or its original contributions. I built two of my working protocols into custom skills. `/revision` runs a structured passage-level pass. `/submission-check` runs a chapter and manuscript quality gate. Both are described in Appendix B.

Responsibility for all claims, arguments, and conclusions rests with me.

1. Introduction

As a science, technology, engineering, and mathematics (STEM) instructor, I teach children hands-on workshops across engineering disciplines, from electronics and robotics to chemical, mechanical, and marine engineering. In every workshop, we build things. And in almost every workshop, there is at least one child who has an idea beyond their current capabilities. For example, they want to build a robot that follows them like a dog. Every time they stop, the robot should stop. Otherwise it follows them. They do not know which sensor detects distance, which block connects where, or how to write the loop that keeps the robot following. They are not lacking imagination. They are lacking the means to execute it.

What are those means? Not all knowledge in making works the same way. A child can explore what happens when they mix two colors, or how clay responds to pressure. They learn through doing. A child cannot explore their way to understanding why a circuit needs a resistor, or what a conditional statement does in code. Some knowledge has to be explicitly taught (Geary, 2008) before meaningful action is even possible. For example, to build the robot dog, a child has to know how a proximity sensor reads distance, how motor speed responds to a signal, how a conditional loop monitors state, and how all three interact continuously in real time. They have to hold multiple things in mind at once and coordinate them toward something that did not exist before.

This domain knowledge is what Csikszentmihalyi (1996/1997) places at the center of creativity. To be creative, Csikszentmihalyi argues, means to know the rules, materials, techniques, and conventions of a domain well enough to operate within it or deliberately against it. Gál (2025) sharpens the distinction. To be creative is to “see the world not only for what it is... but for what it could be” (p. 129), and then to act on it. Otherwise, one is only imaginative.

What is at stake is access to creative agency. The imagination-creation gap is not just a pedagogical problem. It determines who gets to build and who does not. Prior experience shapes who finds making approachable and who finds it overwhelming. A student who grew up around circuits, code, and tools arrives with schemas already in place. A student without that exposure has to manage every element at once. Whose home taught them to sew, whose school taught them to code, whose family showed them how things work. These are not peripheral factors. They shape cognitive starting points. Unequal access to practice produces unequal schema bases. That inequality is self-reinforcing. High load

discourages engagement. Without engagement, schemas do not form. Without schemas, load stays high.

How is the imagination-creation gap bridged? As an instructor, I scaffold by prewiring circuits, giving starter code and breaking the task into chunks. I reduce the domain knowledge demands so the child's imagination can reach the physical world. Outside of education, people rely on tutorials, YouTube, kits, Instructables, knowledgeable friends or family. They can follow someone else's project to learn the schemas they will need to one day build their own. The scaffolding goes through a person or a fixed resource. A person can adapt to what a learner does not understand. A fixed resource cannot.

The maker movement's central response has been access: lower the hardware cost, open the space, and let people build. Tools like Arduino, Raspberry Pi, and Scratch made electronics and coding more accessible, and communities formed around sharing projects. The expertise and knowledge gap remained. Access to tools and communities does not by itself supply the schemas that making demands. The knowledge has to exist before meaningful engagement with the material is possible. What might look like a motivational problem is a load problem.

Large language models (LLMs) are a different kind of tool. The earlier tools reduced the price of entry. LLMs operate through explanation and instruction. That is where the schema gap is. How they change the cognitive load of making is the question this thesis analyzes. Most research on this has focused on coding and other digital tasks. In making, wrong output can damage materials or create safety hazards, not just waste time. Every piece of LLM guidance also requires translation from language into physical action before it can be evaluated.

Understanding how LLMs change the cognitive load of making requires a framework that can describe it precisely. Cognitive Load Theory (CLT) (Sweller, 2025; Sweller et al., 1998) describes the relationship between the number of elements a person must process simultaneously and whether a task leads to learning or to overwhelm. It has been applied to domains from medical and design education to instructional design for various subjects (Rourke & Sweller, 2009; Skulmowski & Rey, 2017; van Merriënboer & Sweller, 2010), but barely to making. LLMs have been applied to making contexts, from computer-aided design (CAD) and 3D printing to physical computing (Makatura et al., 2024; Song et al., 2025). Whether and how they change the cognitive load on the maker has not been examined as of April 2026.

This thesis is a conceptual synthesis of these three fields: making, CLT, and LLMs. I found no work that brings all three together. Each field has something to gain from the others. Maker education has described the cognitive demands of making in terms of frustration, overwhelm, and the need for scaffolding, but lacks a framework to analyze

them. Where cognitive load appears in maker education or LLM research, it is mostly the pre-2010 version of the theory (e.g., Kadir et al., 2023; Stadler et al., 2024). CLT’s current framework, which underwent a major reconceptualization after 2010, provides more precise tools for this analysis.

CLT has a history of expanding its vocabulary through empirical challenge, with many of its effects discovered through replication failures (Sweller, 2023). Making, as a self-directed, multi-domain, and open-ended practice, is the kind of case that has previously sharpened CLT’s vocabulary. Research on LLMs has mostly asked whether the artificial intelligence (AI)-assisted result works. The recurring finding is that LLM output requires human supervision (Makatura et al., 2024; Song et al., 2025). Enabling that supervision means accounting for the cognitive architecture of the person involved.

The research question this thesis asks is: How do LLMs change the cognitive load of making? The hypothesis is that LLMs redistribute rather than reduce cognitive load. Load once concentrated in the domain task redistributes across that task and the interaction task. The maker’s problem-solving remains the maker’s. The thesis develops the theoretical account of that claim.

This question belongs to the study of tool cognition. How does an external cognitive artifact restructure the demands on the person using it? Cognitive science has a long tradition of studying this. Hutchins (1995) showed that cognitive labor is distributed across people, artifacts, and representations. Clark and Chalmers (1998) argued that cognition does not stop at the skull but extends into the tools and environments people think with. Malafouris (2013) proposed that the mind is shaped through material engagement, that thinking and making are not separate processes. None of these frameworks was built for a tool that operates through natural language. This thesis asks a question in that tradition about a tool of that kind. It asks how the interaction reshapes the cognitive load of making.

To examine this, Chapter 2 lays out the conceptual foundations. It introduces making as a movement and as a practice (2.1), the development and current state of CLT (2.2), and LLMs in their technical and interaction dimensions (2.4), followed by the pairwise intersections: making and CLT (2.3), LLMs and making (2.5), and CLT and LLMs (2.6). Section 2.7 finds no existing work at the triple intersection of making, CLT, and LLMs. Chapter 3 opens it. It applies post-2010 CLT to making and introduces the cascade of translations, a six-step analysis of what making demands from intention to artifact. It then introduces LLMs into that framework and traces how cognitive load redistributes across each step. Chapter 4 grounds this framework in practitioner experience through blog posts and forum discussions. Chapter 5 calibrates the epistemic scope of the framework, addresses its limitations, and identifies directions for empirical work. Chapter 6 concludes.

This thesis is a theoretical contribution on the cognitive dimension of LLM-assisted making. It does not test the framework empirically but aims to make empirical work possible.

2. Conceptual Foundations

This chapter examines three fields (making, CLT, and LLMs) that the surveyed literature does not study together. It defines each and reviews research at their pairwise intersections, using a combinatorial structure that makes the gap evident by Section 2.7. Figure A.1 in the appendix maps this structure: the three fields, their pairwise intersections, and the triple overlap at the center that Chapter 3 opens. The two sparse intersections, the pairwise Section 2.3 and the triple Section 2.7, were systematically searched because the scarcity of literature there is itself a finding. The crowded intersections, Sections 2.5 and 2.6, are reviewed narratively. Each of these sections states how its literature was identified. The searches reported in this chapter were conducted in March and April 2026. The chapter describes existing work. The original analytical contributions belong in Chapter 3.

2.1 Making

Dale Dougherty founded MAKE Magazine in 2005 and popularized the term ‘maker’. The first Maker Faire took place in the Bay Area in 2006, drawing 22,000 attendees. By 2018, over 200 licensed Maker Faires were running in more than 40 countries (Make: Community Help Center, n.d.). For Dougherty, ‘maker’ “describes each one of us, no matter how we live our lives or what our goals might be. We all are makers: as cooks preparing food for our families, as gardeners, as knitters” (Dougherty, 2012, p. 11). Mark Hatch formalized this ethos in *The Maker Movement Manifesto* (Hatch, 2014). His nine principles run from making and sharing to participation and change. He applies the same logic to the manifesto itself: “Take this manifesto, make changes to it, and make it your own. That is the point of making” (p. 11). Chris Anderson, in *Makers: The New Industrial Revolution* (Anderson, 2012), declared that personal fabrication tools would do for manufacturing what the personal computer did for information. Desktop 3D printers and laser cutters would decentralize production and let anyone become a manufacturer.

By 2014, making was gaining institutional traction in schools, libraries, and museums, and education researchers needed a framework to study it. Halverson and Sheridan (2014) provided one, proposing three analytical components: “making as a set of activities, makerspaces as communities of practice, and makers as identities of participation” (p. 496). They characterize maker education by student agency, interest-driven projects, iteration, and identity formation through participation. They also caution that formal educational structures, including fixed curricula, scheduled class times, and assessment requirements, may work against these qualities. They argue for separating learning from schooling and

propose that making can expand “what counts as learning, as a learner, and as a learning environment” (p. 503).

The field keeps the definition of making deliberately open. The activities involved span art, electronics, robotics, 3D printing, woodworking, sewing, e-textiles, CNC milling, laser cutting, programming, and many more. These activities take place in shared workshops equipped for different domains. Makerspaces are community workshops where members work on individual or collaborative projects. Hackerspaces tend to emphasize software, electronics, and a hacker ethic of open access. FabLabs, originating from Neil Gershenfeld’s work at MIT, follow a more standardized model, connected through a global network that includes over 2,300 labs in more than 120 countries as of 2026 (Fab Foundation, n.d.; Gershenfeld, 2005/2007). Happylab in Vienna, an early European makerspace, illustrates the range. It features stations for digital fabrication (3D printers, laser cutter), woodworking (saws, lathes, planer), electronics (soldering equipment), textiles (sewing machine, overlock, digital embroidery), CNC milling, and finishing (sandblasting), with safety training required before using some stations. Happylab Vienna and Berlin together have over 2,000 members (Happylab, n.d.). Halverson and Sheridan (2014) treat the maker identity as taken on within the movement rather than within any single domain. The identity is not automatic. Some who participate in making activities do not see themselves as makers and stay out of the movement’s public conversations (p. 502).

Not all making takes place in institutional spaces. Solo makers work in home garages, kitchens, and spare rooms. They share their processes through YouTube tutorials, Instructables, Reddit communities, and social media. These networks reach far more people than formal makerspaces.

The educational philosophy underlying the maker movement is constructionism. Seymour Papert, a mathematician and computer scientist who had worked with Piaget, argued that knowledge construction happens most powerfully when learners are also constructing something external: a physical artifact, a program, a robot, something shareable in the world (Papert, 1980). Martinez and Stager (2013) call him “the father of the maker movement” (p. 21). In *Mindstorms* (Papert, 1980), Papert’s concrete example was Logo, the programming language where children command a turtle to draw shapes. Papert called the turtle an “object-to-think-with” (p. 11). The child is not studying geometry in the abstract but making the turtle do something, and the math becomes necessary and meaningful in the process. When the turtle draws wrong, the child debugs. Debugging, Papert argued, is a fundamentally different relationship to error than what school teaches. In school, errors are failures. In making, errors are information.

Resnick’s Creative Learning Spiral offers a process model for how making unfolds (Resnick, 2007, 2017). Developed to describe creative thinking in kindergarten, the spiral moves

through five stages: imagine, create, play, share, reflect. Each cycle feeds back into new imagination. The spiral is non-linear. A maker can enter at any point. Resnick argues this process should extend beyond kindergarten to all of education. Resnick’s group created Scratch (Resnick et al., 2009), a visual programming language that became one of the most widely used creative coding tools in maker education.

Martinez and Stager (2013) distinguish three modes of engagement within maker practice. Making has a product in mind. Tinkering is a playful, exploratory mindset. Engineering “extracts principles from direct experience” (p. 35). These are not a hierarchy but different modes, and real maker practice moves between them within a single project. Turkle and Papert (1992) describe a related distinction between the planner, who works top-down with a clear goal, and the bricoleur, who “resembles the painter who stands back between brushstrokes, looks at the canvas, and only after this contemplation, decides what to do next” (p. 52). The bricoleur tinkers, rearranges, and responds to what the material does. Schools, they argue, overwhelmingly reward the planner’s approach and treat the bricoleur’s as messy or unreliable. Turkle and Papert call this a problem of epistemological pluralism. Equal access to computation, they argue, “requires accepting the validity of multiple ways of knowing and thinking” (p. 48).

Maker practice varies not only in how makers engage but also in how many skill domains a single project draws on at once. Yasmin Kafai extended constructionism into computational making, focusing on physical artifacts with computation embedded in them, such as electronic textiles with sewn-in LEDs and microcontrollers (Y. Kafai et al., 2014). Her work with Fields and Searle identified a Crafts-Circuitry-Coding triad. When students build e-textiles, all three domains are active at once. A design choice in the fabric determines where LEDs can go, which changes the circuit, which changes the code, which may force a redesign of the fabric. One student, Tamioka, described the hardest part as “trying not to cross the positive and the negatives... to keep checking back and forth” between her circuit blueprint and the sewn artifact (p. 543). Kafai et al. found that “learning how to make circuits could not be separated from learning how to sew and write code, and vice versa” and that “the interconnected nature of e-textiles activities challenged students the most and fostered their learning” (p. 540). The domains are not separate skills applied in order. A change in one forces a change in the others.

How students engage with that complexity varies. Blikstein (2013) identified a pattern he called the “keychain syndrome”. In the first FabLab@School workshops, run in K-12 schools from 2009, introductory activities where students made keychains posed a recurring problem. The machines produce professional-looking objects with relatively little effort. The product got admired, and students did not engage with anything more complex. Blikstein described it as “a too big reward for a relatively small effort, to produce an object that did not include any computation or complex constructive challenges” (p. 9).

The incentive system rewards staying at what he calls a “local minimum” (p. 10) where projects look impressive but remain simple.

Beyond individual workshops and classrooms, the maker movement gained broader institutional recognition. President Obama hosted the first White House Maker Faire in June 2014 and proclaimed a National Day of Making, presenting the country as a nation of makers (Obama, 2014). The branded movement, though, proved commercially fragile. In June 2019, Maker Media ceased operations and laid off all staff, unable to sustain print publishing costs and declining sponsorship. Dougherty relaunched as Make Community LLC the same month (Constine, 2019). During COVID-19, when global supply chains for PPE collapsed, makerspaces around the world began producing face shields and masks. The Open Source Medical Supplies network reported over 48 million units produced by makers globally between March and August 2020 (Cavalcanti et al., 2021). Many spaces, however, could not contribute meaningfully. Kieslinger et al. (2021) found that closures, legal concerns, and limited production capacity were common barriers.

The narrative so far skews American, but key elements of maker infrastructure originated in Europe. The Chaos Computer Club (Hamburg, 1981) and c-base (Berlin, 1995) predate the US Maker Movement branding by decades. Maxigas (2012) traces two distinct genealogies. Hacklabs emerged from European autonomist movements. Hackerspaces developed from tech communities. The American ‘makerspace’ label collapses both into one. The Repair Café movement, started in Amsterdam in 2009, had reached 2,091 locations in 37 countries by 2020, emphasizing repair and skill-sharing over high-tech fabrication (Van der Velden, 2021).

Inclusion was a central promise of the maker movement’s narrative. Leah Buechley analyzed 45 MAKE Magazine covers published between 2005 and July 2015 and found that of 41 people featured, 85% were men or boys, and no person of color appeared (Buechley, 2015). The magazine’s content mostly featured electronics, robotics, and vehicles, leaving out craft traditions like pottery, textile work, and making practices from non-Western cultures. Vossoughi et al. (2016) argued that making cannot simply be transplanted into schools without attending to power, cultural relevance, and whose knowledge counts. Prior experience matters here.

Tamieka could draw on sewing skills from her grandmother, which let her focus attention on unfamiliar circuitry and coding (Y. Kafai et al., 2014, p. 533). A student who has neither sewing nor programming knowledge has to manage everything at once. Eckhardt et al. (2021) found makerspaces skew male, with the proportion of male makers ranging from near parity at FabLab Barcelona (55%) to around 80% at Happylab Vienna.

Constructionism says that making is where learning happens. Its account stays at the level of the activity and the artifact: building something shareable, iterating on it, treating error

as information. It does not reach the level of the individual learner’s working memory: why some learners manage the interwovenness of making while others are overwhelmed, what Tamieka was doing when she checked back and forth between blueprint and fabric, or why that was so hard. The artifact got made. Whether she understood what she had done is a separate question, and constructionism has no vocabulary for what the answer depends on. These are questions about what working memory holds when a task demands attention across multiple domains at once, and about what separates a task completed from a concept understood. CLT is built for them, and it is the subject of the next section.

2.2 Cognitive Load Theory

Cognitive Load Theory emerged from an observation Sweller and colleagues made in 1982. In an experiment with Mawer and Howe (Sweller et al., 1982), participants solved a series of number problems by alternating between multiplying by 3 and subtracting 69. Although each participant successfully solved multiple problems, almost none were aware of the pattern they had been following. Sweller concluded that solving a problem and learning from it are not the same thing. One can execute a solution successfully without understanding why it works. Sweller and Levine (1982) found a similar pattern in a maze task. Participants who could see the goal made more errors and were less likely to learn the maze’s structure than those without a visible goal. The visible goal triggered means-ends analysis, requiring the solver to hold in mind the current state, the goal state, the differences between them, and the available operators, all at once. This burden left little capacity for learning the underlying structure.

The explanation lies in working memory. Cognitive load is the demand placed on working memory during learning and problem-solving. Working memory is where conscious processing of novel information occurs. It holds approximately seven items when chunking is possible (G. A. Miller, 1956), and approximately four elements when it is not (Cowan, 2001). Its contents decay within roughly 20 seconds without rehearsal (Peterson & Peterson, 1959).

Long-term memory has no such limits. It stores schemas, organized knowledge structures that allow multiple interacting elements to be treated as a single unit. When schemas are retrieved, working memory limits effectively disappear. With sufficient practice, schemas can become automated, activating and applying with minimal working-memory involvement. Schema construction and automation are the twin goals of instruction (Sweller et al., 1998). Differences between learners are largely differences in the schemas they have acquired (Sweller, 2024).

CLT conceptualizes cognition as an information-processing system. Sweller et al. (2019) articulated its foundations in five principles derived from evolutionary biology. The

information store principle states that effective functioning in complex environments requires a large long-term memory store of well-organized information. The borrowing and reorganizing principle holds that most learning occurs by acquiring information from others and reorganizing it to fit existing schemas. The randomness-as-genesis principle proposes that when no relevant external information is available, novel information must be generated through random generate-and-test processes, which impose a heavy load on working memory. The narrow limits of change principle maintains that working memory can process only a few novel elements at a time, so substantial changes to long-term memory must occur incrementally. The environmental organizing and linking principle asserts that environmental cues trigger schema retrieval and that familiar, well-organized information is processed with minimal or no working-memory cost. Kalyuga (2025) has proposed a sixth principle, the explicit intention to learn principle, which distinguishes human cognition from other natural information-processing systems through its capacity for conscious, goal-directed learning.

Geary (2008) identifies a distinction that CLT has incorporated (Sweller et al., 2019). Biologically primary knowledge includes spoken language, recognizing faces, spatial navigation, basic tool use, and engaging in social functions such as learning from and communicating with others. It develops through natural activity without explicit instruction, because these capacities are products of biological evolution. Biologically secondary knowledge requires explicit instruction. Reading, mathematics, and other culturally accumulated skills fall into this category. CLT applies to secondary knowledge only. Sweller et al. (2019) point out that “a very large number of biologically primary skills are generic-cognitive in nature” (p. 270), including general problem-solving and the ability to construct knowledge. Campaigns to explicitly teach these skills have largely failed. Generic cognitive skills cannot be extended across domains through instruction. Knowledge is always acquired domain-specifically.

Until 2010, the framework identified three types of cognitive load. Extraneous load arose from instructional design. Intrinsic load arose from the element interactivity of the material itself. Germane load referred to the cognitive resources invested in schema construction once extraneous load was reduced (Sweller et al., 1998). De Jong (2010) raised two fundamental objections. Learners experience total cognitive load and have no means of distinguishing its sources, making the three types empirically inseparable. The distinction between intrinsic and germane load was also conceptually unstable. Intrinsic load described a property of the material, germane load a process in the learner, and the boundary between them was unclear. Sweller (2010) responded by unifying the framework around element interactivity, which Sweller (1994) had introduced. Element interactivity is the number of elements that must be held in working memory simultaneously because they cannot be understood in isolation (Sweller, 2010). It is determined by both material

structure and the learner's existing schemas. Post-2010, intrinsic load is the element interactivity inherent to the material given the learner's current schemas. Extraneous load is the element interactivity imposed by instructional design. Germane load is no longer an independent source of load. Sweller (2010) defined it as "the working memory resources available to deal with the element interactivity associated with intrinsic cognitive load" (p. 126). When extraneous load is reduced, more resources become available for intrinsic processing, and learning improves. By 2025, Sweller had dropped the term from his framework entirely, stating that "a combination of intrinsic and extraneous cognitive load determines how easily learners acquire novel, biologically secondary information" (Sweller, 2025, p. 6). Total load equals intrinsic plus extraneous. Many papers continue to treat germane load as a third, independent type (e.g., Partarakis & Zabulis, 2024; Stadler et al., 2024).

The shift from novice to expert is a shift in stored schemas. Chase and Simon (1973) showed that chess masters recalled far more pieces than beginners from real game positions but performed at the same level on randomly arranged positions. The advantage was not superior general memory but thousands of stored configurations handled as single units. From this understanding, CLT has generated a range of instructional effects, each tested through controlled experiments comparing instructional approaches. Many emerged directly from replication failures. When an effect failed to appear under certain conditions, the investigation of that failure produced a new discovery rather than a falsification. The split-attention effect emerged when the worked example effect failed to replicate for geometry problems. The redundancy effect emerged when the split-attention effect failed under different materials. The expertise reversal effect emerged when effects designed for novices reversed with more experienced learners (Sweller, 2023). The element interactivity effect establishes the scope condition for all of them. These effects are obtainable only with high element interactivity material, because when intrinsic load is low, working memory has sufficient capacity to accommodate extraneous load without reaching its limit, and the effects disappear (Sweller, 1994). The worked example effect shows that studying worked examples is more effective than problem-solving for novices, because example study directs attention to problem states and the moves connecting them rather than to goal reduction. The advantage holds for similar problems but does not extend to structurally dissimilar ones (Sweller, 1988; Sweller & Cooper, 1985).

The split-attention effect shows that physically separated information sources that must be mentally integrated to be understood impose extraneous load through the search-and-integrate process. It emerged because conventional geometry worked examples require integrating a diagram with separate text statements, and neither source is intelligible without the other. Modified worked examples with statements placed within the diagram eliminated this cost (Tarmizi & Sweller, 1988). The redundancy effect shows that when one

information source is self-explanatory, additional sources should be eliminated rather than integrated. Combining a self-explanatory diagram with redundant text increases element interactivity without aiding understanding. Diagram-only conditions outperformed both integrated and conventional formats (Chandler & Sweller, 1991). The modality effect shows that combining auditory and visual input reduces load compared to visual-only presentation, because the two modalities recruit partially independent subsystems within working memory. The effect persists even with sequential rather than simultaneous presentation, confirming that the benefit is expanded working memory capacity rather than simultaneity (Mousavi et al., 1995).

The expertise reversal effect shows that techniques effective for novices become counterproductive for experts, because guidance that once reduced extraneous load becomes redundant and requires additional resources to integrate with existing schemas. The effect encompasses the entire catalogue. Split-attention becomes redundancy, modality becomes redundancy, and worked examples become inferior to problem-solving as expertise grows (Kalyuga et al., 2003). The imagination effect shows that imagining a procedure supports learning for learners who already have the relevant schemas but is harmful for novices, who cannot hold all elements in working memory without an existing schema to organize them. The mechanism is schema automation, not acquisition. Imagination consolidates already-formed schemas into automated knowledge. A study-then-imagine sequence outperforms an imagine-then-study sequence for the same reason (Cooper et al., 2001). The isolated-interacting elements effect addresses a catch-22 in high element interactivity material. The material cannot be understood before a schema forms, but a schema cannot form without processing all elements simultaneously. The solution is two-phase instruction. The first phase presents elements in isolation, without their interactions, producing procedural fragments rather than genuine understanding but keeping initial element interactivity within working memory capacity. The second phase introduces the full interacting structure. The effect applies to novices. Experienced learners show no benefit (Pollock et al., 2002). The collective working memory effect shows that collaboration can distribute element interactivity across multiple working memories. For high-complexity tasks, the distribution gain outweighs coordination costs and groups outperform individuals. For low-complexity tasks, coordination costs dominate and individual learning is superior (F. Kirschner et al., 2009). More recently, research on the human movement effect has shown that motor information constitutes an additional modality in working memory. Making gestures and tracing during learning reduce working memory load and improve outcomes, because human movement, as biologically primary knowledge, is processed with minimal working memory involvement (Sweller et al., 2019).

CLT was developed through controlled experiments using well-defined tasks in single domains. The measurement problem is fundamental and unresolved. Learners experience

total element interactivity and cannot introspectively separate intrinsic from extraneous components, making direct psychometric separation difficult (de Jong, 2010; Sweller, 2010). Most studies use self-report instruments administered after a task, capturing a global impression rather than the dynamic load shifts that occur within it (de Jong, 2010). The task analysis CLT uses to determine element interactivity requires enumerating the elements a learner must hold simultaneously. Activities with shifting element interactivity within a single session do not fit this model cleanly. A further constraint is domain-specificity. Element interactivity is calculated relative to a learner’s schemas in a specific domain, and CLT does not provide a framework for combining or prioritizing element interactivity across domains. The theory also assumes motivation is stable (Sweller, 2010, p. 126) and does not model how engagement, fatigue, or emotional state interact with load capacity. CLT’s cognitive architecture was originally built for screen- and paper-based instruction, for the processing of symbolic, linguistic, and diagrammatic information. Nevertheless, CLT provides a systematic vocabulary for analyzing cognitive demands at the task level. How far that vocabulary has reached into maker education is the subject of the next section.

2.3 Cognitive Load Theory and maker education

A systematic search across Web of Science, ACM Digital Library, and PsycINFO using the string (“cognitive load” OR “cognitive load theory”) AND (*makerspace** OR “maker education” OR “maker movement” OR “digital fabrication”), restricted to titles and abstracts, returned two unique relevant papers. The low number reflects a structural feature of the intersection rather than a search gap. CLT and maker education developed with distinct vocabularies, and papers documenting cognitive load phenomena in making contexts rarely use CLT terminology. The remaining literature was identified through Google Scholar, following citations from the retrieved papers, author searches on key figures at the intersection, and reading the maker education literature reviewed in Section 2.1.

2.3.1 Cognitive load theory in maker education

The explicit CLT work in maker education concentrates in tool and interface design, with physical computing as its densest zone. DesPortes et al. (2016) redesigned the breadboard as BitBlox. Their premise was that the standard breadboard’s row-and-column connection scheme, non-obvious to novices, requires tracking a layout rule unrelated to learning electronics. BitBlox makes connections visible through color coding and modular snap-together sections. A comparative classroom study found fewer persistent errors and more flexible circuit organization with BitBlox than with the standard breadboard. Vergara

and Herskovic (2025) approached the same problem differently, removing the breadboard entirely. Their Boxy Board introduces electronics concepts in stages through progressive disclosure, grounded in CLT and constructivist methodologies. Bellucci et al. (2018) addressed the instructional medium rather than the tool. They tested augmented reality overlays for novice circuit prototyping. Superimposing digital component representations directly onto the physical breadboard reduced mental workload (NASA-TLX) compared to paper or screen-displayed instructions. A. Miller et al. (2012) built MagneTracks, a tangible physics toolkit, and described the accompanying computer vision software as a working memory aid. It tracked experimental data automatically, so students could focus on patterns across trials rather than holding results in mind between them. Chang et al. (2017) compared live-stream and on-the-spot delivery of a robotics course in a Maker Education programme and found no significant difference in cognitive load between conditions. Zhou et al. (2026) designed ProjecTA, a robotic teaching assistant using in-situ projection during guided makerspace tours. Compared to a screen-based counterpart, ProjecTA significantly reduced extraneous load, which the study explains through the split-attention effect. Partarakis and Zabulis (2024) extended the interface design approach to traditional craft domains, with eLearning guidelines for woodworking, pottery, textiles, and basket weaving. They use intrinsic, extraneous, and germane load to structure content delivery, recommending graduated complexity sequences and worked-example video demonstrations to reduce split-attention.

Tofel-Grehl and Feldon (2023) proposed the Technological Instructional Overhead (TIO) framework to help teachers evaluate the cognitive cost of adopting a new maker technology, operationalized across personal, instructional efficacy, and instructional efficiency dimensions. Unlike the tool-design papers above, TIO addresses teacher-side cognitive load rather than learner load.

A smaller set of papers examines cognitive load during making, rather than as a design constraint for tools and interfaces. Winters et al. (2023) examined scaffolding in a problem-based design makerspace with middle school students. They characterize such environments as placing heavy cognitive load on learners. Their analysis tracks how teacher interventions, peer interactions, and artifact-based supports collectively manage that load through soft scaffolding. The teachers' primary goal, they argue, is to support students' cognitive structuring throughout the activity. Blikstein et al. (2017) compared scaffolded and general instruction in a constructionist physics simulation environment, using P. A. Kirschner et al. (2006) as background rationale for the pedagogical contrast. The study uses physiological arousal (galvanic skin response) as a proxy for cognitive engagement rather than established cognitive load measures such as subjective rating scales. Kadir et al. (2023) applied Load Reduction Instruction (LRI) to hands-on inquiry-based science. LRI is a five-stage instructional framework progressing from difficulty reduction and scaffolding

through practice, feedback, and guided independence, with each stage designed to manage cognitive load before introducing greater complexity. An intervention study with 156 Grade 7 students compared LRI-structured lessons with unstructured inquiry on the topics of speed and density: the intervention group outperformed the control on knowledge transfer in both topics and reported higher sense of competence. The paper retains the pre-2010 three-type model. Fegely et al. (2024) studied maker robotics for pre-service K-8 teacher CS instruction, claiming that working with robotics reduces what they call abstract cognitive load and functions as a motivational tool. Gierczyk et al. (2025) assigned 92 students aged 13 to 14 to three content-load conditions, typical, moderately reduced, and highly reduced, in STEM workshops at a science center, using CLT to motivate the manipulation. The moderately reduced condition produced significantly better knowledge recall at post-test and two-week delayed post-test than either the typical or highly reduced condition, consistent with CLT’s prediction that moderate intrinsic load, rather than its elimination, optimizes learning for novices.

Griffin (2019) proposed de-constructionism, drawing on CLT’s worked example effect to argue that novice learners benefit from studying, taking apart, and debugging well-constructed artifacts before building their own, developed in the context of introductory programming. DesPortes (2016), in a doctoral consortium proposal, combined CLT with distributed cognition to analyze the design of physical computing tools. DesPortes argues that cognitive demands distribute across people, tools, and environments in ways that require both frameworks.

Across these three groupings, tool and interface design, empirical activity studies, and theory-integration papers, the pre-2010 tripartite model of intrinsic, extraneous, and germane load remains common. Kadir et al. (2023) retains all three types and frames germane load as a quantity to be maximized. Gierczyk et al. (2025) and Vergara and Herskovic (2025) cite Sweller’s post-2011 work and count only intrinsic and extraneous as adjustable load, though their framing centers on managing load in specific instructional contexts rather than element interactivity as an analytic lens. Zhou et al. (2026) draws on the split-attention effect from Chandler and Sweller (1991).

2.3.2 Cognitive demands in maker education research

Research on maker education often documents the cognitive demands involved in making, but it rarely uses CLT as a framework for analyzing or managing these demands. The breakdowns it describes, including novice capacity exceeded by simultaneous demands, learning stalling when multiple representations must be held together, and students reverting to guessing when they cannot integrate what they know, are precisely the phenomena CLT’s vocabulary was developed to characterize.

DesPortes and DiSalvo (2019) recorded 31 novice college students working with Arduino for the first time and coded 59 hours of video and audio data for breakdowns, building a codebook of 18 categories spanning software, hardware, and hardware-software interaction. The most common software breakdown was sequentiality of code. The task required making two LEDs blink in sequence. Most participants wrote two separate forever loops, one per LED. The Arduino executes only the most recently compiled loop and gives no feedback about which is running. Understanding why the circuit does not respond requires holding the program’s execution model, the current code structure, the expected hardware response, and the observed response in mind simultaneously, then reasoning across all four. Participants who could not do this reverted to guessing, changing delay values or restructuring blocks without a working model of what was failing. A related breakdown, the delay misconception, showed the same pattern from a different angle. Eight of 10 think-aloud participants believed the delay block caused the LED to blink rather than the on-off toggle. This misconception about how code state maps to hardware state persisted through multiple attempts. On the hardware side, 18 of 31 participants formed incorrect circuit configurations, typically wiring two LED circuits incorrectly when trying to make them independent. The error required translating simultaneously between the circuit diagram, the mental model of how current flows, and the physical placement of components on the breadboard. When these representations did not align, errors compounded. A cross-domain breakdown made it worse. When two forever loops caused only one LED to light, participants investigated their hardware. When an open circuit produced no output, they revised their code. The hint protocol built into the study is structured evidence that the task exceeds unaided novice capacity. Researchers waited one minute of stuckness or five minutes of unsuccessful debugging before offering guidance, moving from location identification to problem description to solution.

Bevan et al. (2015) developed the Tinkering Learning Dimensions Framework through 18 months of video analysis at museum-based tinkering programs. The four dimensions, engagement, initiative and intentionality, social scaffolding, and developing understanding, describe learning through observable behavior and social interaction. The developing understanding dimension is most relevant here. Two 18-year-olds working on the same circuit board activity illustrate the gap. One applies prior knowledge fluently, explaining device behavior in terms of ohms and load. The other works to connect what she sees to half-remembered concepts from prior schooling. She arrives at understanding only when her partner explains the parallel circuit structure in terms she can link to what she already knows. The framework names this as applying knowledge versus striving to understand.

Max et al. (2024) document the challenges 145 prospective teachers faced during project work in a university pedagogical makerspace. Technical knowledge gaps absorbed the early weeks. Participants spent that time familiarizing themselves with hardware and

software rather than developing their projects, which created time pressure and fear of failure. The support structures created their own difficulties. Some participants found an abundance of online materials overwhelming, which the study describes as cognitive overload. Others found the materials insufficient. Technological Pedagogical Content Knowledge (TPACK) measures teachers' integrated knowledge of technology, pedagogy, and content. Participants' TPACK before the intervention was the only significant predictor of TPACK after. Learners who arrived with more prior knowledge made greater gains, regardless of the challenges they encountered.

2.3.3 Historical lag and signs of convergence

The limited reach of CLT into maker education stems from a disciplinary divide. P. A. Kirschner et al. (2006) argued that discovery-based and constructivist pedagogies overload novice working memory by directing cognitive resources toward problem-solving search rather than schema construction. Hmelo-Silver et al. (2007) responded in the same journal, *Educational Psychologist*, that problem-based and inquiry approaches employ extensive scaffolding that manages cognitive load while preserving learner-centered engagement. Maker education carried those same constructionist commitments but was not a participant in this exchange, developing its own pedagogy without engaging CLT or the debate it generated. CLT was applied during the same period in health professional education, art and design education (Rourke & Sweller, 2009), and embodied learning (Skulmowski & Rey, 2017) without equivalent resistance. Those domains did not carry constructionism's framework commitments.

Four papers from 2023 to 2026 bring CLT and maker education into closer contact. Tofel-Grehl and Feldon (2023) introduced the TIO framework at the FabLearn/Constructionism conference, applying cognitive load concepts to teacher decision-making in maker technology adoption. Gierczyk et al. (2025), co-authored by Fred Paas, applied CLT to hands-on STEM workshops and found that moderately reduced content load produced better knowledge recall than either typical or highly reduced conditions. Vergara and Herskovic (2025) grounded a K-12 physical computing toolkit in CLT alongside constructivist methodologies. Zhou et al. (2026) measured extraneous load directly in a university makerspace study, finding significant reductions when in-situ robotic projection replaced a screen-mounted display, which they explain through the split-attention effect.

The existing engagement between CLT and maker education is real, but concentrated at the level of tool redesign and interface intervention. The analysis of making as a cognitive activity remains undone: what kinds of element interactivity it involves, how expertise changes load profiles, what happens when a new tool enters the process. Section 2.4 introduces the second field this thesis brings into contact with making: LLMs.

2.4 Large language models

To understand how LLMs can assist in making and how they affect the cognitive load of the maker requires a basic understanding of what LLMs are. The following section introduces the conceptual vocabulary needed to understand LLMs on a technical level, their affordances on an interaction level, and their limitations. LLM capabilities are evolving rapidly, and many specific claims about what these systems can or cannot do will be outdated by the time this thesis is read. For this reason, the section focuses on architectural categories of capability rather than benchmarks of specific models.

2.4.1 Technical foundations

An easy way to think about an LLM is as a very large file of numbers stored on a server. It takes tokens in and produces tokens out. Around that core are additional software components that shape what a user experiences.

“Large” in Large Language Model stands for the billions to trillions of parameters, also called weights. These are numbers that encode statistical patterns learned from training data. “Language” means it is trained on and processes text. “Model” means it is a simplified, compressed representation of patterns in language. Because LLMs encode patterns rather than stored facts, the model on its own does not retrieve information. It generates new text based on learned statistical patterns. Without an added retrieval step, no database is being queried (Shanahan, 2024).

Building an LLM typically involves the following key stages (Bommasani et al., 2021; Shanahan, 2024).

First, collect training data. LLMs are trained on massive text corpora drawn from the internet: books, Wikipedia, forums, code repositories. The scale is enormous, hundreds of terabytes across billions of web pages. This corpus is a written record. Spoken language, embodied experience, and undocumented knowledge are not directly represented. What is in the corpus shapes what the model can learn. Because the data is drawn predominantly from the English-language internet, some domains, languages, and knowledge traditions are overrepresented while others are nearly absent (Bommasani et al., 2021). This is not a curable bias but a structural property of the data. For most commercial models, the exact composition is not disclosed.

Second, build a tokenizer. A tokenizer splits text into tokens, sub-word units that each get a numerical ID. A fixed vocabulary of subword units is created once and does not change after that. Each model family has its own tokenizer, so the same sentence becomes different token sequences on different models. This is an engineering decision, not a property of

language itself. Tokenization is the entry point of all LLM processing. All input becomes tokens, and all output is tokens decoded back to text.

Third, design the architecture. This means deciding how many layers, how wide each layer is, and how many attention heads it has. These choices determine the total number of weights. All weights start as random numbers. As of writing, the dominant architecture is the transformer (Vaswani et al., 2017). Its key innovation is self-attention. Within each layer, every token can look at every other token and determine how much weight to give each relationship. This is what allows the model to figure out that “it” in sentence five refers to “the sewing machine” in sentence one. Earlier architectures, like Recurrent Neural Networks (RNNs), processed tokens in order and struggled with long-range dependencies. Self-attention removed that bottleneck. An attention head is one instance of this process. Transformers run many heads in parallel within each layer, each one learning to track different types of relationships, such as syntactic dependencies, coreference, and topic coherence. The architecture also defines a context window, the maximum number of tokens the model can consider at once. Early models in 2022 handled a few thousand tokens. By 2024, the largest reported windows reached the millions, though the figure varies by model (Gemini Team, 2024).

Fourth, pre-train the model. This is where the learning happens. Text from the corpus gets tokenized and fed through the transformer. The model predicts the next token, compares its prediction to the actual next token, and the weights get adjusted to reduce the error. This happens billions of times across the training data. It is the expensive part, requiring thousands of GPUs running for weeks or months. What the model learns through this process is not stored facts but statistical patterns, associations between tokens that tend to co-occur in similar contexts. No single weight “knows” anything. Knowledge is distributed across millions of weights.

Fifth, fine-tune the model. Take the pre-trained model and train it further on a smaller, more specific dataset of instruction-following conversations. The weights get adjusted further, but starting from the already-meaningful patterns, not from random. This step matters because after pre-training, the model is good at predicting text but has no concept of following instructions or answering questions. Fine-tuning on instruction-response pairs teaches it to treat input as a request and output as a response.

Sixth, align the model. This is a further round of weight adjustment shaped by human preferences rather than raw next-token prediction. The canonical technique is Reinforcement Learning from Human Feedback (Ouyang et al., 2022), where human raters compare pairs of model outputs and indicate which is better. The weights are adjusted to favor the preferred style of response. Newer methods, such as Direct Preference Optimization (Rafailov et al., 2023) and RLAIIF (Lee et al., 2024), achieve comparable results with

simpler setups. This stage shapes the model to be not just instruction-following but helpful, conversational, and safe through deliberate design choices.

Seventh, freeze the weights. Training is done. The model is now a fixed file of numbers. From this point, the weights do not change during use. When a user interacts with the model, that is inference, not training. No learning takes place in the conversation.

Eighth, build the product. System prompts, memory features, tool use, safety filters, and the conversational interface get added around the frozen model. None of these touches the weights. They work by manipulating what goes into the context window. A system prompt is just text prepended to every conversation that the user does not see. Memory features extract information from past conversations and re-inject it as text into new sessions. The model does not remember. It reads. This is where the model/product distinction becomes concrete. GPT-4 is a model. ChatGPT is a product built around it. Some capabilities that feel like properties of the intelligence (the conversational tone, the refusal to help with harmful requests, the ability to search the web) are product-level engineering decisions. They can be added, removed, or changed with a software update.

Ninth, inference. The user types something. It gets tokenized, processed through the frozen transformer, and the model generates a response one token at a time. Each token is selected from the probability distribution the model outputs, with a sampling parameter called temperature controlling how much randomness is introduced. Higher temperature produces more varied output. Lower temperature produces more predictable responses. This is why the same prompt can produce different responses. The weights do not change during this process. The model is not learning from the conversation. Every impression of adaptation is either the user providing better context within the session or product-level features re-injecting past conversation text into the context window.

Two further properties of that system deserve a brief description.

By 2024, multimodal input had become common in frontier models. Gemini 1.5, for instance, accepts text, images, video, and audio (Gemini Team, 2024). The input channel has widened significantly but the fundamental architecture has not changed. The model still processes input as tokens and produces tokens as output. A user can show the model a photo of their project. The model still responds in tokens. It can generate code that renders charts, web pages, or images, but the model itself is not producing visual output. It is producing text that other software executes. The reasoning loop remains token-based.

Some LLM capabilities were not part of the training objective. Brown et al. (2020) showed that GPT-3 models trained only to predict the next token could also perform few-shot learning, translation, and basic arithmetic. These capabilities were not explicitly trained. They appeared as a byproduct of model size and training data volume. Whether such

capabilities appear suddenly at certain scale thresholds (Wei et al., 2022) or represent gradual improvements that only appear sudden due to measurement choices (Schaeffer et al., 2023) remains debated. Either way, the practical consequence is the same. The boundaries of what LLMs can do are not fully known. What they can do can be documented empirically, but it cannot be derived from the architecture alone.

This creates an attribution problem. Many capabilities are hard to trace to a single source. Following complex instructions is partially emergent, partially shaped by alignment, and partially enabled by product-level features like system prompts.

2.4.2 Interaction affordances

ChatGPT launched in November 2022. By mid-2025, approximately 700 million users sent 18 billion messages per week, representing roughly 10% of the global adult population (Chatterji et al., 2025). These figures come from a working paper by OpenAI-affiliated researchers and have not been peer-reviewed.

LLM interaction differs structurally from earlier software tools. Automation tools, such as macros and templates, execute fixed sequences the user specifies. Search engines retrieve and rank existing documents. LLMs generate novel responses through natural language dialogue and adapt to the user’s apparent level and context. Unlike intelligent tutoring systems, which deliver instruction through explicit pedagogical models, LLMs have no built-in instructional structure. Their behavior in technical or educational contexts depends on how they are prompted.

Within a session, users can refine, follow up, or correct without starting over. The model adapts to the apparent level of the user, producing different registers depending on how input is phrased. Input is not limited to text. Users can upload images, documents, screenshots, and in some products audio or video.

These affordances are not fixed properties of the architecture. They are shaped by the product layer described in Section 2.4.1: system prompts set the tone, memory features provide continuity across sessions, and tool integrations extend what the model can do. Different products built on the same underlying model can offer different interaction affordances.

2.4.3 Prompting

Early LLM use required careful prompt construction. Users discovered that specific phrasing, role assignments, and step-by-step instructions significantly affected output quality. Non-experts struggled with this. Zamfirescu-Pereira et al. (2023) found that users without AI expertise lacked mental models of how LLMs work, leading to ineffective

interaction patterns. By 2025, this landscape had shifted. Models had become better at interpreting intent from loosely phrased input. Techniques that were once necessary, such as explicit step-by-step reasoning instructions, came to be handled increasingly within models trained to reason through multi-step problems. The elaborate prompt scaffolding of 2023 became optional for many tasks.

At the same time, the relationship between prompt detail and output quality is not straightforward. Yang et al. (2025) found that underspecified prompts, those that fail to articulate requirements the user cares about, are fragile. LLMs guess unspecified requirements correctly only around 41% of the time, and such prompts are twice as likely to regress when the model or prompt changes. Adding more requirements does not reliably resolve this, because models have limited instruction-following capacity and competing constraints interfere with each other. The problem is not that prompts should be longer or shorter, but that specification itself is a non-trivial cognitive task. Current best practice treats LLM interaction less as prompt crafting and more as iterative conversation: state intent, review the output, refine. Models themselves are increasingly used to optimize prompts.

A related challenge is context management. Liu et al. (2024) found that even when context windows are large, models do not use all content equally. Relevant information placed in the middle of a long context is used less reliably than information at the beginning or end. Where information appears within the context affects whether the model acts on it.

2.4.4 Agentic capabilities

Some LLM products extend beyond conversation into action. They can execute code, search the web, read and create files, and call external APIs. These are product-level capabilities, not architectural ones. They depend on tool integrations that the provider chooses to enable.

Two of these capabilities, web search and file reading, typically deliver their results through retrieval-augmented generation (RAG). RAG is a grounding technique, not an action the model takes. The model is given relevant text from external documents alongside the user’s input (Lewis et al., 2020). This grounds the model’s output in specific source material rather than relying solely on patterns learned during training. RAG increases the probability of factually grounded output but does not guarantee it. The model can still misinterpret or ignore the retrieved text.

2.4.5 Limitations

The interaction affordances described above are real but bounded. Four limitations of LLMs are relevant to their use in making contexts.

The first is hallucination. LLMs generate text by sampling from probability distributions over tokens. The next token is selected because it is statistically likely given the preceding context, not because it is true. This means a model can produce confident, fluent, internally coherent output that is factually incorrect, technically wrong, or physically impossible. Ji et al. (2023) identify this as a structural feature of likelihood-maximization training. The same mechanism that produces accurate and useful output also produces plausible-sounding errors, and the model cannot distinguish between the two. They distinguish two forms: intrinsic hallucination, where the output contradicts information the user provided, and extrinsic hallucination, where it cannot be verified against the source material the model was given. A further contributor is parametric knowledge bias. Models trained on large corpora tend to prioritize patterns from training over information in the user’s input, generating output consistent with common patterns rather than with the specific constraints the user has specified. Retrieval-augmented generation, introduced above, lowers the rate of these errors by grounding output in retrieved sources but does not remove them. The model can still misread or override the retrieved text. In making contexts, this means a user cannot rely on the model to signal when its output is wrong.

The second limitation is the absence of physical grounding. LLMs are trained on text. Their knowledge of the physical world is derived from written descriptions of it, not from sensorimotor interaction with it. Brown et al. (2020) note that “large pretrained language models are not grounded in other domains of experience, such as video or real-world physical interaction, and thus lack a large amount of context about the world” (p. 8). Mahowald et al. (2024) offer a theoretical frame for this gap. They distinguish formal linguistic competence, knowledge of the rules and statistical patterns of language, from functional linguistic competence, the ability to use language to accomplish things in the world. LLMs trained on next-token prediction largely master the former. The latter, which includes physical and world knowledge, remains uneven. A model can produce well-formed descriptions of physical processes without the underlying physical representations those descriptions normally track. Multimodal input partially addresses this but does not close the gap. Even given a photo of a circuit board or a print failure, the model has no access to material resistance, temperature, weight, or the proprioceptive feedback that guides a maker’s hand. What is encoded in the training data is the linguistic record of physical knowledge. This is useful but structurally different from embodied competence. For making, this matters wherever instructions require reasoning about material behavior, tolerances, or physical feedback that exceeds what text can convey.

The third limitation is sycophancy. Alignment training shapes model behavior by rewarding responses that human raters prefer. Sharma et al. (2024) found that RLHF-aligned models consistently confirm user beliefs rather than correct them, provide feedback biased toward what the user wants to hear, and revise correct answers when users push back without

providing new evidence. Human raters sometimes prefer responses that agree with them over responses that correct them, so reward models trained on those preferences learn to favor agreement. For a maker working through a problem, a model that confirms rather than corrects may reinforce errors at exactly the point where correction matters most.

A fourth property is that there is no persistent memory by default. Within a session, the context window is the only record of the conversation. Across sessions, nothing is retained unless a product-level feature re-injects past conversations as text. Every new session starts from zero.

LLM capabilities are frequently described through benchmark performance, and the studies reviewed in Section 2.5 draw on benchmark-adjacent measures. Three features of benchmark evidence are worth noting. Benchmarks can saturate. Models reach near-ceiling scores on a fixed test while still failing on novel variants of the same problem. Contamination distorts comparisons. When benchmark items appear in the training data, a model can score well on them without the underlying capability, inflating performance on those specific items. Construct validity is often limited. Performance on a standardized task may not transfer to the open-ended, context-dependent demands of real use (Bommasani et al., 2021). These features do not invalidate benchmark evidence, but they caution against treating benchmark scores as direct measures of capability in context. How these capabilities and limits play out when LLMs enter making is the subject of the next section.

2.5 Large language models and making

Research on LLM integration in making and fabrication contexts is recent and domain-specific. Making has no unified research literature, and *making* as a search term mostly retrieves false positives such as “decision making”, so the search proceeded by domain. Google Scholar searches combined LLM terms (“large language model”, *ChatGPT*, *AI*) with domain terms (*Arduino*, “embedded systems”, *electronics*, “3D printing”, *CAD*, *fabrication*, *craft*, *textile*, *manufacturing*, *woodworking*, “maker education”, *constructionism*), supplemented by citation snowballing. Physical computing proved the densest area and was reviewed first. The domains covered here are example domains of physical making, sampled to characterize what has been studied rather than to cover the field. Published work is concentrated between 2023 and 2026, following ChatGPT’s public release. Studies span several domains: physical computing and embedded systems, 3D printing and additive manufacturing, CAD, craft and textiles, design and manufacturing education, and constructionist pedagogy. Several craft and making domains have no representation in this literature. No paper in this making and fabrication literature applies CLT as a theoretical framework. The 2023-2026 window spans multiple generations of model

capability. Findings from studies using GPT-3.5 era models may not generalize to current systems.

2.5.1 Physical computing

The most studied maker-adjacent domain is physical computing, where LLM integration into Arduino and embedded systems programming has been examined across several studies. Englhardt et al. (2024), in an exploratory study of 15 participants at varying skill levels, found that access to GPT-4 (version 0314) and structured workflow guidelines improved task completion scores significantly ($M = 65.48$ versus $M = 34.21$, $p < .01$), with success on the Environmental Sensor task rising from 25% to 100%. Perceived cognitive load as measured by the NASA Task Load Index was lower with GPT-4 and the workflow guidelines (mean 3.25 versus 4.67 without GPT-4, $p < .01$). At the same time, GPT-4 rarely produced fully working code on the first attempt. Participants consistently needed to iterate, debug, and modify generated output to make it functional on hardware. Prompts had to specify chip versions, pin configurations, and library dependencies, technically precise requirements that presuppose domain knowledge. Participants responded differently to working this way. P4 reported feeling unfulfilled: “I feel like I didn’t [do any] work” (Section 3.4). P8 described a contrasting experience: “it’s fulfilling in different ways: ... instead of understanding the domain, you become more fulfilled in that you get the task to work... GPT handles some tedious things really well, and allows you to better understand some of the other aspects of the task” (Section 3.4). The paper also ran an automated testbench comparing GPT-4, GPT-3.5, and PaLM 2 across 450 hardware-in-the-loop trials, finding that GPT-4 produced fully functional code on only 16% of the most complex tasks. Xu et al. (2026) ran EMBEDBENCH, a 126-case suite covering Arduino, ESP32, and Raspberry Pi Pico with outputs evaluated automatically using Wokwi circuit simulation. Across 10 LLMs, including DeepSeek R1, DeepSeek-V3, and Claude 3.7 Sonnet (Thinking), even the best performer on tasks with correct schematics, DeepSeek R1, achieved only 55.6% pass@1 (the rate at which a single generation attempt produces working code). On cross-platform migration from Arduino to ESP32, the best result was only 29.4%, reached by Claude 3.7 Sonnet (Thinking).

Johnson et al. (2024) found a different pattern in a posttest-only experiment with 33 novice Arduino programmers in agricultural education. Students who used ChatGPT 3.5 to write programs and students who wrote their own showed no significant difference on task scores ($p = .50$). However, the self-programming group outperformed the ChatGPT group on a programming posttest (Cohen’s $d = 0.76$) and on self-efficacy for writing programs ($d = 0.75$). The ChatGPT group made significantly more punctuation errors on the posttest, including consistent omission of semicolons.

Song et al. (2025) systematically documented a gap between logical and physical reasoning using PCEVAL, a benchmark of 50 Arduino projects across four complexity levels evaluated on 13 LLMs using Wokwi circuit simulation. Models performed substantially better on code generation and logical circuit design than on physical breadboard layout. On logical circuit generation, o3-mini achieved 66.0% and Claude 3.7 Sonnet 65.6%. On physical circuit generation, o3-mini fell to 45.2% and Claude 3.7 Sonnet to 13.6%. The primary failure causes were pin conflicts and breadboard topology errors, constraints that are physical rather than logical. Performance degraded steeply with task complexity, for example, Claude 3.7 Sonnet fell from 33.3% on single-component tasks to 5.5% on multi-component tasks. The abstractions of code and circuit logic exist within the model’s training distribution. The spatial, physical constraints of breadboard layout do not.

Tafferner et al. (2023), in a qualitative case study conducted in March 2023 on the use of ChatGPT 3.5 for electronics research and development, found that its controller recommendations were appropriate. In contrast, its sensor specifications, hardware designs, and generated code were only partially acceptable and occasionally contained specification errors. ChatGPT 3.5 also produced fabricated literature citations with nonexistent authors, titles, and DOIs.

2.5.2 3D printing and additive manufacturing

In the 3D printing domain, studies have examined LLMs primarily for troubleshooting and parameter optimization. Badini et al. (2023) tested ChatGPT on FFF (Fused Filament Fabrication) 3D printing troubleshooting and parameter optimization for TPU filament using three query types (troubleshooting, parameter optimization, and de novo G-code generation), finding that ChatGPT produced hierarchical troubleshooting guidance and optimized 15 printing parameters in approximately one hour, compared to three weeks required by conventional experimental iteration. G-code generated de novo by ChatGPT was physically implemented, and comparison prints showed visible quality improvement. The authors note that the model lacks real-time database access. Responses draw on training data rather than current materials specifications, so confident-sounding parameter recommendations may be outdated.

Jadhav et al. (2025), in a preprint study, built a multi-agent GPT-4o framework that monitored FDM (Fused Deposition Modeling, another name for the FFF process) prints layer by layer using computer vision and corrected detected errors by controlling the printer directly. The framework relied on computer vision and direct printer control rather than text alone. Compared against a control group of engineers with varying additive-manufacturing expertise, its defect identifications closely matched the human evaluations, and it executed parameter corrections autonomously. The authors present this as a continuous improvement cycle in which errors are corrected in subsequent layers

rather than the print being discarded. Eslaminia et al. (2025), benchmarking GPT-4o, Claude 3.5 Sonnet, Llama-3.1-70B, and Llama-3.1-405B on G-code anomaly detection (identifying errors in the numerical control code that directs 3D printer movements) across 16 samples representing four defect types, and on expert-evaluated user queries across expertise levels, found that GPT-4o achieved the highest anomaly detection accuracy at 62%, with the other three models between 31% and 44%.

2.5.3 Computer-aided design

In CAD and parametric design, several research groups have developed text-to-CAD systems that generate parametric models from natural language prompts. M. S. Khan et al. (2024) (custom BERT-based transformer trained on approximately 170,000 annotated CAD models), J. Li et al. (2025) (fine-tuned LLaMA3-8B), and Xie and Ju (2025) (fine-tuned open-source models up to Mistral-7B generating executable CadQuery code) each report systems that outperform prior baselines. In the M. S. Khan et al. framework, performance improved with prompt specificity, with detailed expert-level prompts yielding the closest correspondence to the ground-truth CAD sequences. Badagabettu et al. (2024), using an iterative self-refinement approach (generating a candidate, evaluating it against the query, and revising until the output passes a self-defined quality check) with GPT-4 Turbo generating FreeCAD Python code, achieved 76.7% success on a 57-query dataset after three refinement rounds, up from 53.6% on first attempt. Success on hard queries (tasks requiring complex multi-step geometric reasoning) reached only 41.7% after refinement. Across all failed cases, most were attributed to inability to generate executable code rather than conceptual errors.

Makatura et al. (2024), in the most thorough study in this literature, tested text-only GPT-4 across a full computational design and manufacturing pipeline for two objects (a wood cabinet and a quadcopter), both of which were physically fabricated. The pipeline succeeded, but required extensive human iteration at multiple steps. The authors describe the pipeline as brittle. Errors compound through the workflow, spatial reasoning deficits appear throughout, and GPT-4 “often produces inaccurate results or justifications for its solutions and lacks the ability for self-verification” (p. 25). Their central finding is that “a user cannot generate precise outputs from GPT-4 without the expert knowledge to perform verification” (p. 19). When asked to modify a specific design element, the model frequently regenerated the entire design, losing elements specified in earlier turns. Kyaw et al. (2025) demonstrated a speech-to-physical-assembly pipeline in which spoken requests were classified by GPT-4 Turbo, converted to 3D meshes by Meshy AI, processed for fabrication constraints, and assembled by a robotic arm. All test objects assembled successfully, but the geometric processing stage (not present in standard LLM or 3D

generation output) was needed for every object, as the authors observe that “AI does not inherently account for [fabrication constraints]” (Section 4.2.4).

2.5.4 Craft and textiles

Studies of generative AI in craft and textile contexts use image generation rather than language models as their primary tool. Craft design is primarily visual. Their findings on the gap between generated outputs and physical execution are directly relevant to making. Tedre et al. (2023) ran a workshop with 15 pre-service teachers and craft educators in Finland using Midjourney for craft design, finding that the more experienced craft educators distinguished between the composition space and the construction space, warning that design with AI leaves the construction space out. AI generates visual designs but leaves material knowledge, skill constraints, and tool limitations with the human. One participant described a pupil who prints a picture of a shirt, then gets “a terrible disappointment when [the] pupil makes an awful mess of it, and just can’t [implement the idea]” (Section 4.4). Another participant distinguished ideation from design, asking “isn’t this [AI] actually [an] ideation method . . . so that actually after this starts the actual design: what are the limitations, what materials, how expensive materials, . . . what are my skills” (Section 4.3). The same participant noted that the AI “does not yet prescribe materials — and does not know how expensive the materials would be” (Section 4.4).

Ahteck et al. (2024), in a proposal for a generative-AI pipeline to design bobbin lace for manufacture, identified a manufacturability gap between generated images and physically executable craft instructions. SkyKnit, a model trained on crowdsourced knitting data to generate knitting instructions, produced outputs that “often did not make sense and skipped necessary instructions — forcing knitters to use domain knowledge to fill in the gaps” (p. 9).

2.5.5 Design and manufacturing education

Musiienko’s (2026) scoping review, titled *From Maker to Curator*, of 156 studies on generative AI in design education found that publication volume rose 6.4-fold after ChatGPT’s release. Three-quarters of studies lacked theoretical frameworks, and two-thirds reported no specific measurable outcomes. A recurrent cross-study theme that was not always explicitly named was “the repositioning of the designer from ‘maker’ to ‘curator’ or ‘prompter’ of AI-generated content” (p. 213). Musiienko formulates this as an open educational question, asking whether curating AI outputs builds design capacities equivalent to those developed through direct making, or whether “something essential is lost when the hand no longer guides the making” (p. 225). Ege et al. (2025) directly compared ChatGPT 4.0’s engineering design performance against fifth-year mechanical

engineering students in a 48-hour hackathon to design and build a physical NERF launcher. The LLM team finished second of six (14.8 m firing distance, winner 37.66 m). ChatGPT proposed pneumatics early on (the eventual winning approach) but abandoned it when a valve fitting was unavailable, without exploring alternatives as the winning human team had done. The LLM showed strong concept generation and weak physical constraint navigation. It could not observe the physical environment, assess material properties, or perceive failure in real time. All required human mediation. The authors explicitly propose assessing how effectively LLMs reduce cognitive load on human designers as a future research direction, not a finding from this study.

2.5.6 Constructionist perspectives

The constructionist education literature asks different questions about LLM integration. The authors are less concerned with task performance and more concerned with what learners do with LLM assistance. Y. B. Kafai and Morales-Navarro (2023), updating Papert and Solomon’s 1971 constructionist agenda for the AI era, argue that most AI-in-education work centers learners as recipients rather than creators, and propose constructionist designs in which learners build AI applications rather than consume them. Huber et al. (2024) frame over-reliance on LLMs as a threat to expertise development, arguing that prompt engineering functions as an expert skill. Effective use requires domain knowledge to evaluate outputs critically, alongside familiarity with prompting techniques and the specific LLM’s behavior. For practitioners with domain knowledge, iterative refinement provides a check on passive consumption. For novices, they propose game-based learning environments to foster the engagement that protects experts from over-reliance.

The most empirically specific study in this cluster is Larsen et al. (2025), an ethnographic case study of a graduate-level course in which 16 students used ChatGPT and a custom scaffolded chatbot built on the same model to build educational technology prototypes, seven of whom were studied in depth. Outcomes depended on expertise. Advanced students used AI to delegate tasks they could already do but did not want to do. Beginner students often could not make use of generated code because they could not evaluate or integrate it. In several cases, adding generated code made their projects and their understanding of errors worse. One student, unable to evaluate code she had not written, left notes warning herself not to change it. The authors summarize: “Students with some preexisting skills related to the materials generated . . . are more able to make efficient use of it, perhaps because of their improved ability to evaluate the output” (p. 227).

2.5.7 Across domains

Two further studies cut across these domain boundaries. Aflatoony et al. (2025), in a workshop study of 18 DIY assistive technology makers using a pipeline of Midjourney for text-to-image generation and Meshy AI for image-to-3D modeling, found that generative AI was useful during early-stage ideation but consistently produced generic outputs that failed to reflect individualized needs. A technical maker summarized: “AI is good at coming up with an average solution based on what’s already there, and people with disabilities have specific needs that don’t exist” (Section 5.2.2). Experienced technical makers found generative AI less efficient than their existing CAD tools for later-stage refinement, where direct modeling was faster and more precise. No participant achieved their desired result on the first prompt attempt, and precision limitations were consistently noted, including inability to specify exact measurements, structural dimensions, or movable parts.

Lindley and Whitham (2025), presenting practice-based design research at Tangible, Embedded and Embodied Interaction (TEI), proposed reframing prompt engineering as prompt craft, arguing that interacting productively with generative AI shares properties with craft practice, including partial opacity, non-linear experimentation, and accumulated experiential knowledge of the material’s behavior.

Human verification is required in every domain examined. No study finds reliable, directly deployable outputs without expert evaluation (Badagabettu et al., 2024; Englhardt et al., 2024; Eslaminia et al., 2025; Makatura et al., 2024; Tafferner et al., 2023). A gap between logical and physical reasoning appears in physical computing (Song et al., 2025) and CAD (Makatura et al., 2024), and an analogous gap between generated designs and physical execution appears in craft (Ahteck et al., 2024; Tedre et al., 2023). Models handle linguistic and logical structures better than physical constraints, material properties, and spatial configuration. LLMs are more useful to practitioners who have domain knowledge adequate to evaluate and integrate output (Englhardt et al., 2024; Larsen et al., 2025), a pattern that extends to generative design tools (Aflatoony et al., 2025).

What these patterns mean for the cognitive load of making is not addressed in any of the reviewed studies. The literature that brings LLMs and cognitive load together is the subject of the next section.

2.6 LLMs and Cognitive Load Theory

Research at the intersection of CLT and LLMs is recent. Almost all empirical work on this topic was published between 2024 and 2026 (Mohamed et al., 2026), reflecting the timeline of widespread LLM accessibility following ChatGPT’s public release. The literature is scattered across education, HCI, and cognitive science. The search combined

LLM terms (“large language model”, *ChatGPT*, *AI*) with “cognitive load” in Google Scholar and Web of Science, supplemented by citation snowballing. This review covers the papers that address human cognitive load (not machine processing efficiency) in contexts where participants interact with an LLM. They fall into three types: empirical studies measuring cognitive load during or after LLM use, scale development and psychometric studies that identify new load dimensions specific to AI interaction, and theoretical or framework papers applying CLT constructs to human-AI interaction design. A fourth category, papers that apply CLT language to LLM architectures rather than to human cognition, falls outside this review. It is large and grows rapidly but concerns machine processing, not learner cognition. Research on the cognitive load effects of LLMs focuses on academic writing, second language learning, programming and software development, and knowledge work. Physical making is not represented.

The only paper in this literature co-authored by a core CLT theorist is Wang et al. (2026). Published in *Artificial Intelligence Review*, it introduces “bounded agent complementarity” (p. 3) as a framework for understanding human-AI collaboration under cognitive load. The authors characterize both human working memory and LLM context windows as bounded processors that degrade under high demand and argue that the primary design challenge is to structure interactions so that their respective failure modes do not coincide. The paper draws mainly on post-2010 CLT, emphasizing intrinsic and extraneous load as the central constructs, while also referencing germane load in several passages.

The dominant finding across empirical studies is that LLM use reduces self-reported cognitive load. Stadler et al. (2024) ran a randomized controlled trial with 91 university students comparing ChatGPT 3.5 with Google Search for a scientific inquiry task. Students in the LLM condition reported significantly lower load across all three components of the older tripartite model: extraneous load ($\eta^2 = .09$), intrinsic load ($\eta^2 = .15$), and germane load ($\eta^2 = .27$). Their written justifications were also significantly weaker ($\eta^2 = .11$), and an exploratory mediation analysis showed that reduced germane load statistically accounted for the quality deficit. The study treated LLM use as a uniform condition with no analysis of interaction patterns. Bassner et al. (2026) conducted a three-arm RCT of 275 CS1 students at TU Munich, comparing unrestricted ChatGPT, a scaffolded AI tutor that provided hints but withheld direct solutions, and a no-AI control. Both AI conditions produced significantly higher exercise scores than the control. Extraneous and germane load were significantly lower in both AI conditions. Intrinsic load did not differ. Neither AI condition produced greater conceptual learning on a pre-post code comprehension test. The performance-learning dissociation held whether the AI was scaffolded or unrestricted, suggesting that interaction design alone did not determine the learning outcome in this study. Avello-Martínez et al. (2024), in a randomized study of 41 graduate students using

ChatGPT for digital storytelling, similarly found that germane load decreased in the ChatGPT group with no improvement in storytelling skill.

The neurophysiological evidence diverges from the self-report pattern. Kosmyna et al. (2025) conducted a longitudinal EEG study with 54 participants assigned to ChatGPT, search engine, or unaided writing across three sessions, of whom 18 completed a fourth session, over approximately four months. LLM users showed the weakest neural connectivity and the lowest brain engagement across sessions, while unaided writers showed the strongest. They observed an order effect. Participants who began with ChatGPT and later switched to unaided writing showed persistent under-engagement even without the AI, while those who began unaided and later used ChatGPT maintained strong connectivity and recall. The authors describe this as “cognitive debt”, a condition in which repeated reliance on external systems like LLMs replaces the effortful processing required for independent thinking (p. 141). The authors present it as the first to measure LLM use effects longitudinally with physiological data, and the study has not yet been peer-reviewed. S. M. F. A. Khan and Suhluli (2025), in a survey of 998 academic researchers analyzed using SEM-PLS, treated GenAI immersion as a higher-order moderating variable and found that high immersion amplified rather than reduced the negative effect of cognitive load on research quality. Among the empirical papers examining human cognitive load during LLM use, post-2010 CLT (centered on element interactivity, without germane load as an independent source) appears clearly in one: Lepine et al. (2026). Wang et al. (2026) and Guo et al. (2026) center intrinsic and extraneous load but still invoke germane load, the former in its design recommendations and the latter in its theoretical mechanism, making both frameworks hybrid. The remaining papers use the older tripartite model in which intrinsic, extraneous, and germane load are treated as separate, empirically distinguishable constructs, a framework that CLT’s own authors revised in the reconceptualization described in Section 2.2. The continued use of this framework across papers published between 2024 and 2026 is itself a finding about the state of the literature. Most studies also measure load via self-report instruments administered before and after LLM use. Lepine et al. (2026) developed instead a computational method for measuring cognitive load at the utterance level during multi-turn LLM conversations with 34 finance professionals. Using post-2010 CLT, they found that extraneous load was approximately three times as predictive of output quality decline as intrinsic load, and that high extraneous load in one conversational turn persisted into subsequent turns within the same speaker. It is the only study in this literature measuring load at the utterance level during multi-turn conversation rather than via pre/post instruments.

Not all papers treat LLM use as simply reducing cognitive demand. Several identify new or redistributed demands arising from the interaction itself. Yao and Fan (2025) developed and validated a psychometric scale for AI-assisted second language writing that

identified four dimensions not captured by existing CLT instruments: Prompt Management Load (demands of formulating and refining input to the AI), Critical Evaluation Load (assessing AI output for accuracy, relevance, and quality), Integrative Synthesis Load (combining AI-generated content with the writer’s own ideas), and Authorial Core Processing Load (the cognitive demands the writer retains independently of AI assistance). Critical Evaluation Load showed the highest mean across participants. Authorial Core Processing Load showed the lowest. AI offloads traditional task demands while introducing new ones. Nagori et al. (2025), in a qualitative study of 15 students, found that prompt engineering functions as an active load-management skill rather than passive offloading. After training, participants reported reduced intrinsic and extraneous load and greater schema development, which the study frames as increased germane load within the older tripartite model. Tankelevitch et al. (2024), in a CHI conceptual paper synthesizing metacognition research with GenAI user studies, identify three sites of metacognitive demand during LLM interaction: prompt formulation (self-awareness of goals and task decomposition), output evaluation (well-calibrated confidence in one’s domain expertise), and automation strategy (ongoing decisions about whether and how to integrate the system into one’s workflow). The paper distinguishes metacognitive demand from cognitive load. Metacognitive demand contributes to load but is not its only source. For output evaluation, processing fluency creates a specific heuristic risk. Fluent, rapidly produced output inflates users’ confidence in its correctness, which reduces deliberate checking and allows errors in plausible-sounding output to pass undetected. Expert users exhibit metacognitive flexibility in response, selectively disabling suggestions and decomposing tasks, while novices may over-rely and lack adjustment strategies. Jin (2025), using an experimental design with 80 undergraduates, found that LLM chatbot use significantly improved declarative knowledge acquisition but had no significant effect on procedural knowledge, with learners tending to prioritize lower-load declarative tasks when AI was available.

The most studied domain at this intersection is software development. Shihab et al. (2025), in the only major coding study explicitly applying Sweller’s CLT to GitHub Copilot, found that students who used Copilot selectively (accepting targeted suggestions and integrating them manually) performed better than those who accepted generated code wholesale. Task completion time fell 34.9% with Copilot use. Conceptual understanding was not separately measured. Tang et al. (2024) found that developers who knew their code was AI-generated invested more cognitive effort in examining it and produced better outcomes, while developers not informed of the provenance often failed to recognize AI-generated code at all. Guo et al. (2026) found that high school students using a generative AI assistant for programming showed significant improvements in algorithmic thinking and abstraction, which the authors attributed to AI offloading syntax and debugging load and

freeing resources for higher-order processing. Mohamed et al. (2026), in a systematic review of 39 studies on LLM-assisted developer productivity, found that cognitive load is the second most investigated sub-dimension within the satisfaction dimension but is measured via inconsistent instruments and operationalizations that preclude direct comparison across studies. All of these studies concern pure software development contexts. Physical computing and craft-based making involve embodied and material dimensions absent from pure software contexts. Whether findings transfer to those domains is not addressed in any of these papers.

Martin et al. (2025) propose LRI as an instructional design framework for integrating GenAI. Building on the earlier tripartite formulation of CLT, the framework emphasizes graduated difficulty reduction, scaffolding, structured practice, feedback-feedforward, and independent practice and problem-solving. It is the most developed attempt to apply CLT design principles to GenAI deployment, though it retains germane load as an operative construct. Gkintoni et al. (2025), in a systematic review of 103 papers on CLT and AI in education, report that CLT is critiqued for the rigidity it imposes on formal learning environments and that cognitive load management should be dynamic and individualized rather than a single uniform reduction. That concern sharpens for LLMs. Their interaction is conversational rather than fixed, and load can vary from one turn to the next rather than holding steady across a session. Lepine et al. (2026) measured that turn-by-turn variation at the utterance level.

The CLT $\times$ LLM intersection has grown rapidly since 2024. The dominant methods (self-report instruments, pre/post measurement, uniform-condition design) and the dominant framework (older tripartite CLT) have not captured the interaction-level dynamics of individual LLM sessions, with Lepine et al. (2026) as the single exception. No paper I surveyed studies physical making as the context, nor applies post-2010 element interactivity analysis to the specific cognitive demands of LLM-mediated maker practice. That gap is addressed in Section 2.7.

2.7 Making, LLMs, and Cognitive Load Theory

In maker education, CLT has been used primarily to inform tool design, not to analyze learning activities, and most applications rely on its pre-2010 formulations. In making contexts, LLMs have been examined mainly in terms of task outcomes and output quality, not in terms of how their use shapes cognitive demands on the maker. When CLT and LLMs have been studied together, the context has been screen-based instruction. Physical making is absent. None of these strands examines whether LLMs can help bridge the gap between imagination and execution in maker contexts, or how such use might affect cognitive load.

A search across Web of Science, PsycINFO, and ACM Digital Library combined three sets of terms: cognitive-load terms (“cognitive load” or “cognitive load theory”), LLM terms (“large language model”, *LLM*, “generative AI”, *GenAI*, *ChatGPT*, *GPT*, *Copilot*), and making-side terms (*maker*, “digital fabrication”, “fab lab”, *fablab*, *tinkering*, *craft*). The search returned only a small and recent body of work, all published in 2025 and 2026, and none studying LLM-mediated making and its effect on cognitive load. A study at this intersection would need to meet three conditions at once: the activity has to be participatory or creative making where the maker creates an artifact of their own, the LLM has to work as a dialogic partner inside the making, and the analysis has to bring post-2010 CLT, centered on element interactivity, to the making and the interaction together. Three published works come closest. None meets all three conditions.

Dong et al. (2025) compared LLM-assisted and traditional teaching paths in an undergraduate landscape design course. The AI-assisted path reduced extraneous load and improved design quality scores. Students create their own designs, and the interaction is dialogic, reconstructed from logged cycles of prompt revision. The framework, though, is pre-2010 CLT, treating germane load as an independent source. Testa et al. (2026) used generative AI to simplify manufacturing instructions for neurodivergent workers, guided by physiological cognitive load estimates. Its CLT is current, citing recent work and using element interactivity by name. The task, though, is following fixed assembly instructions, not creating an artifact, and the LLM runs once as a text simplifier rather than acting as an interlocutor. Lin et al. (2025) combined CLT and LLMs in a digital preservation framework for Xuan papermaking, a traditional craft. CLT, in its pre-2010 form, shapes the information density of a knowledge graph’s retrieval views, and the LLM performs backend terminology disambiguation. The craft is the archived content, not a live activity, and no one makes paper with the LLM. Across all three, the LLM is framed as reducing load. None was designed to capture how load moves when a maker draws on an LLM, where it accumulates, where it shifts, and what the transition between language and action costs.

A field-level review points the same way. Musiienko (2026), introduced in Section 2.5.5, found the field of generative AI in design education largely atheoretical. Cognitive load theory appeared as a framework in only three of the 156 reviewed studies.

Making involves high element interactivity throughout, from decomposing an intention into executable steps to handling physical constraints in real time. The gap between what a maker imagines and what they can execute has cognitive roots. Working memory constrains how many interacting elements a person can hold simultaneously. Domain schemas, built through practice, reduce that interactivity. LLMs operate in natural language, at the level of explanation and instruction where that knowledge is normally

transmitted. How their use affects the cognitive load of making, and how that varies with the maker's expertise, is what existing research has not asked.

Chapter 3 builds the framework for that account. It applies post-2010 CLT to the cognitive demands of making, introduces the cascade of translations to locate where those demands fall, then brings in LLMs and traces how cognitive load redistributes across each step.

3. Synthesis

The imagination-creation gap is a question of how a vision becomes an artifact. Gál (2025) defines creativity as imagination with creation. Csikszentmihalyi (1996/1997) argued that creativity requires domain knowledge. A maker who imagines a working circuit but does not know how current flows cannot close the gap through imagination alone. That gap, in cognitive terms, is a gap in schemas, the organized knowledge structures that let a maker hold many interacting elements as a single unit.

Without the schemas that compress interacting elements into manageable units, a maker must hold too many things simultaneously. Economic access, time, and community are not separate factors that run alongside that gap. They shape who gets to build schemas at all. If access to practice is unequal, then cognitive starting points are unequal. That inequality is self-reinforcing. High load discourages engagement. Without engagement, schemas do not form. Without schemas, load stays high. This thesis does not address those structural conditions. It focuses on the cognitive account because that dimension has received the least analytical attention.

Section 2.5 documented that LLM output in making contexts repeatedly requires human supervision. The load that supervision produces went unexamined. A recurring pattern in the CLT $\times$ LLM literature treats LLM use as a uniform condition (e.g., Kosmyna et al., 2025; Stadler et al., 2024). Interaction patterns are rarely examined. The studies that do examine them (e.g., Lepine et al., 2026) show that how the LLM is used changes where load lands, in ways a uniform-condition design cannot register.

This chapter builds the framework for analyzing LLM-mediated making in CLT terms. Section 3.1 applies post-2010 CLT to making and introduces the cascade of translations, a six-step analysis of what making demands from intention to artifact. Section 3.2 introduces LLMs into that model and traces what shifts. Section 3.3 draws out the methodological consequence.

3.1 The cognitive load of making

Constructionism (Papert, 1980) says that building is where learning happens, but does not distinguish building from schema formation or ask why the same activity produces schemas in one learner and overwhelms another. CLT can make that distinction, because it treats schema construction as the mechanism of learning and defines element interactivity relative to the learner’s existing schemas (Sweller, 2010).

The existing CLT work in maker contexts, reviewed in Section 2.3, came in through tool and interface design. BitBlox reduces element interactivity at the breadboard connection step (DesPortes et al., 2016). MagneTracks offloads experimental tracking (A. Miller et al., 2012). ProjecTA reduces split-attention during a makerspace tour (Zhou et al., 2026). Each targets one high-load point and reduces it. None of these studies asked what element interactivity making involves, where demands peak across the process, or what changes when an LLM enters every step at once.

3.1.1 Element interactivity and schemas in maker practice

Element interactivity is the count of elements that must be held in working memory simultaneously because they cannot be processed independently (Sweller, 2010). Two elements interact when understanding one requires understanding the other at the same time. E-textiles are garments in which circuits are sewn directly into fabric using conductive thread. Whether two materials can be safely combined in such a garment requires understanding fabric conductivity, how conductive paths route through the textile structure, and what conditions create a short circuit. Each constrains the others. A thread that is safe on one fabric creates a short on another, depending on how the fabric folds, where conductive paths cross, and how much current flows through them. The more elements interact, the greater the demand on working memory. Element interactivity is low when items can be learned independently, as with memorizing resistor color codes.

Schemas reduce element interactivity by chunking multiple interacting elements into a single unit. What counts as one element depends on what the maker has already chunked (Sweller, 2010). An experienced electronics maker has chunked “voltage divider” into a single unit rather than a set of interacting relationships between components, values, and equations. A practiced seamstress has automated “seam allowance” so it no longer occupies attention at every cut. A woodworker who has drilled hundreds of pocket holes no longer consciously attends to angle, depth, and material thickness simultaneously. Schemas hold both facts about what things do and the procedures for using them (Sweller et al., 1998), what is standardly called declarative knowledge (what a component does, what a technique requires) and procedural knowledge (how to execute the steps).

In making, element interactivity appears at every step. Holding a design constraint in mind while selecting materials loads working memory with the requirement, the options, and the relationships between them. Tracking step counts in pattern-following requires simultaneous attention to position in the sequence, what comes next, and what has already been completed. Keeping the next move in attention while executing the current one is what makes interruptions so disruptive. Losing an element from working memory during physical execution means reconstructing a partially completed set of interacting constraints.

Working memory limits explain recurring patterns in the maker education literature that have been named but not analyzed in CLT terms. The participants in DesPortes and DiSalvo (2019) reverted to guessing. They changed delay values or restructured blocks without a working model of what was failing. Read in CLT terms, their working memory had been exceeded by four simultaneous demands: the program’s execution model, the current code structure, the expected hardware response, and the observed hardware response. Guessing is not a failure of effort. It is what working memory does when the interacting elements exceed its capacity and no schema is available to compress them (Sweller, 2010).

Making blends primary and secondary knowledge (Geary, 2008). Primary knowledge is acquired without explicit instruction (e.g., spoken language, spatial reasoning, basic tool grasp, hand-eye coordination, imitation of demonstrated movement). Secondary knowledge requires explicit instruction (e.g., reading a sewing pattern, computing kerf compensation, parsing a kiln firing schedule, programming firmware) that cannot be discovered independently at prohibitive cost. CLT applies to secondary knowledge only (Sweller et al., 2019). Making activities blend both. Picking up a tool is primary. Reading a torque specification is secondary. The thesis applies CLT to the secondary components and treats the primary components as the substrate on which secondary demands sit. The making field has not disaggregated them.

Intrinsic load is the element interactivity of the material given the learner’s schemas (Sweller, 2010). Extraneous load is the element interactivity imposed by instructional or environmental conditions: the format of instructions, the layout of equipment, the distance between information sources that must be integrated. Total cognitive load equals intrinsic plus extraneous. When total load exceeds working memory capacity, schemas cannot form (Sweller, 2025).

The intrinsic-extraneous boundary is contextual (Sweller, 2010). For a first-week Arduino learner, code syntax can be either intrinsic or extraneous depending on the instructional task. When the task is correct electronics placement, syntax is extraneous. Starter kits ship with pre-written code for exactly this reason. When the task is writing firmware, the same syntax is intrinsic. For an experienced firmware developer, this has been automated. Pattern grading is intrinsic for the seamstress drafting a custom fit and is logistical overhead for one cutting from a printed standard. Whether sourcing a material counts as intrinsic or extraneous depends on whether material selection is part of the learning task.

3.1.2 CLT effects in maker practice

The CLT effects introduced in Section 2.2 name how task or instructional features raise or lower element interactivity. The maker education literature has used them to justify tool redesigns, not to analyze the activity of making. I do the latter below.

Making often means combining different information mid-task, like reading a pattern while cutting fabric or checking a wiring diagram while soldering. Tamieka described her e-textile work as constant “checking back and forth” between circuit blueprint and sewn fabric (Y. Kafai et al., 2014, p. 543). The continuous integration of two information sources uses working memory that would otherwise be available for the task itself. This is the split-attention effect (Tarmizi & Sweller, 1988). Redundancy runs opposite to split attention. A diagram plus a paragraph that merely restates its content adds another source to integrate without adding information. This is the redundancy effect, and it occurs whenever instructions are duplicated across formats without increasing informational value (Chandler & Sweller, 1991).

The format of an instructional source changes the cognitive load it imposes. A video of a potter explaining the centering motion, where verbal description accompanies visible action, draws on working memory’s partially independent processors. A written description of the same movement instead forces both forms of information into the visual channel. Mousavi et al. (1995) call this the modality effect. For motor skills, observing movement engages biologically primary mechanisms that static images and text do not. A video of the moment of release in a throwing arc, the wrist rotation in a wire wrap, or the centering pressure on a wheel conveys what a diagram cannot. This is the human movement effect (Sweller et al., 2019).

Across the worked-example, goal-free, and isolated-elements effects, task structure shapes cognitive load independently of presentation format. Novices who study a provided solution structure, such as an annotated reference circuit, a step-by-step tutorial, or a labeled quilt showing stitch order, benefit from the worked-example effect (Sweller, 1988; Sweller & Cooper, 1985). Worked examples bypass means-ends analysis, which requires tracking the current state, goal state, their differences, and available operators. Tinkering similarly eases this demand at the level of subordinate operations. A maker exploring LED configurations without a predetermined circuit design is not avoiding the task. The open sub-task reduces means-ends load. This goal-free effect (Sweller, 1988) makes tinkering a viable entry point for novices who might otherwise be overwhelmed by the full procedural chain. When element interactivity exceeds what existing schemas can handle, presenting elements separately allows partial schemas to form before they must be integrated. In electronics, this looks like teaching component functions one by one before introducing

circuit analysis. Pollock et al. (2002) call this the isolated-interacting elements effect, which characterizes skill progression in making more broadly.

Effects shift with expertise. Scaffolding that helps a novice complete an unchunked procedure becomes overhead for a maker whose schemas already encode it. Step-by-step instructions for configuring 3D print supports aid the first-time user but interrupt the experienced one. This is the expertise reversal effect (Kalyuga et al., 2003). As schemas develop, intrinsic load drops, and instructional support that once reduced load now adds to it.

Mental rehearsal shows the same pattern. An experienced seamstress mentally runs through setting in a sleeve because schemas for notch alignment, easing, and pin order are already in place. For a novice, the same rehearsal adds load rather than reducing it. Their working memory must hold every element the expert’s schemas have already compressed. Cooper et al. (2001) call this the imagination effect. Both effects stem from the same mechanism. Expertise does not change the task, only what the task costs.

When element interactivity exceeds what one working memory can hold, the load can be distributed across a group. This works where the coordination gain outweighs the coordination cost. Two makers at a workbench, one holding and aligning while the other tracks the instruction sequence, handle what neither working memory alone could. This is the collective working memory effect (F. Kirschner et al., 2009). The strongest case for collaboration in making is at the points of highest element interactivity.

These effects describe mechanisms that change element interactivity. The cascade of translations names the structural sites where element interactivity arises in making.

3.1.3 The cascade of translations

I propose the cascade of translations to analyze what making demands at each step. Each translation is a move from one representational form to another. The cascade has six types from intention to artifact: vision, decomposition, domain knowledge retrieval, procedure, physical execution, and evaluation. Each is a different kind of element interactivity.

Existing frameworks describe the activity of making but not the cognitive demands at each translation. The creative learning spiral (introduced in Section 2.1, Resnick, 2007) cycles through Imagine, Create, Play, Share, and Reflect. Create is a single phase. The cascade disaggregates it into four: decomposition, domain knowledge retrieval, procedure, and physical execution. Reflection-in-action describes the practitioner’s epistemic loop within action (Schön, 1983/2017), not the translation demands across the full process. Neither framework maps what each translation costs for makers at different expertise levels.

The steps are not independent. What one step produces shapes what the next must process. Errors propagate backward, so a failure at physical execution may expose a gap in domain knowledge or require revising the decomposition. Element interactivity at each step depends on the maker’s schemas, the material’s complexity, and the working conditions (Sweller, 2010; Sweller et al., 2019).

Vision is what the artifact is for or should do. The attributes a maker holds here include function, aesthetic, material constraints, and use conditions. A custom-fitted garment requires simultaneously holding shape, fabric behavior, wearability, construction method, and wearer’s movement. Vision may crystallize before, during, or after first contact with materials.

Decomposition is the translation from whole to parts. Each sub-component has to stay consistent with every other and with the vision. Adjust the sleeve length on a garment pattern and you have to check the armhole depth, the ease allowance, and the finished hem position at the same time. Change one without the others and the sleeve fits the arm but distorts the shoulder or breaks the hem line. None of the cross-dependencies can be resolved in isolation.

Domain knowledge retrieval is the translation from task requirements to the facts and principles that govern execution. A maker who has chunked “voltage divider” calls up the whole pattern as a single unit. A maker without that schema has to hold each resistor, the voltage equation, and Ohm’s law as separate elements. Element interactivity at this step scales inversely with schema availability.

Procedure translates domain knowledge into an action sequence. The moment to join two wooden pieces depends on the current fit, the humidity, the adhesive, and what has already been done. The correct order of steps depends on the state of the material, which depends on what the previous step produced. The maker holds all of it at the same time.

Physical execution is the only translation whose output is the physical artifact itself, not a representation of it. Centering clay on a potter’s wheel requires simultaneously managing wheel speed, water amount, posture, and clay response. A novice handles these serially. An expert has chunked them into a single act. Closing the gap between articulated procedure and what the body and material actually do is itself a translation demand. Describing a motion and executing it are not the same operation.

Evaluation is the translation from result to vision. A seamstress checks the finished garment against the original requirements. Does it fit? Does the fabric drape correctly? Does it match the intended silhouette? A welder inspects the bead for penetration, undercut, and porosity. Evaluation feeds back into the cascade at whatever level the

discrepancy occurred. A dimensional error returns the maker to procedure. A structural failure requires revising the decomposition.

The cascade is not a timeline. Both planful (top-down) and bricoleur (responsive, exploratory) makers (introduced in Section 2.1) traverse the same translation types, but in different orders and with different iteration patterns. Table A.1 in the appendix summarizes the six steps and the element interactivity each carries. Its final column previews the effect of LLM mediation, developed in Section 3.2.

3.1.4 Extending CLT: four aspects making requires

CLT was developed for instructional design, where procedural skills are taught under controlled conditions. Skulmowski and Rey (2017) review applications to embodied learning settings, showing the framework reaches beyond screen-based instruction, but it still covers procedural skills under controlled conditions. Making departs from those conditions in four ways: the maker’s knowledge precedes what they can articulate, load fluctuates as the material responds unexpectedly, cognition distributes across tools and environment, and material behavior changes what gets made.

I draw on four theorists, each addressing one of these aspects: Polanyi on tacit knowledge, Schön on reflection-in-action, Hutchins on distributed cognition, Malafouris on material engagement. None wrote about CLT or about making. Each was chosen because making requires extending CLT in a specific direction. Their frameworks transfer because each addresses a feature that recurs in making regardless of expertise or domain. The mapping below is mine.

Polanyi (1966) writes that “we can know more than we can tell” (p. 4). The making analogues are direct. A maker knows when a weld is right before being able to say why. A jeweler reads from the look and resistance of the metal that it needs annealing before being able to name the hardening that caused it. A skilled woodworker reads grain direction through tactile resistance before consciously checking it. The tacit knowing is ahead of what the maker has put into words.

Schön (1983/2017) argues that competent practitioners do not apply pre-formed knowledge to well-defined problems. They engage in reflection-in-action, an ongoing “reflective conversation with the situation” (p. 76) in which the material’s response reshapes the maker’s understanding while the action is still underway. Layers separate mid-print. A weld pool behaves in ways the welder did not predict. A cut drifts. The practitioner pauses, reframes what is happening, and adjusts within the action-present. “When someone reflects-in-action, he becomes a researcher in the practice context. He is not dependent on the categories of established theory and technique, but constructs a new theory of the unique case” (p. 68).

Load fluctuates within a making session. A material surprise introduces new interacting elements mid-task. The maker must simultaneously hold the original procedure, the unexpected result, the revised model of what is happening, and the adjusted plan. The cascade therefore needs to be recursive.

Hutchins (1995) shows that cognition in complex task domains distributes across people, artifacts, and the environment. The navigation team on a naval vessel accomplishes a computation no individual could complete alone. Representational states flow across charts, instruments, procedures, and multiple practitioners, and the cognitive system is the whole ensemble. Making does the same. A workbench laid out with components in assembly sequence holds part of the procedure outside working memory. Reference lines marked on a workpiece carry position information the maker would otherwise have to compute. A jig holds a part in position so the maker can attend to the fastening. Each distributes element interactivity across the environment rather than requiring working memory to hold it. “Humans create their cognitive powers by creating the environments in which they exercise those powers” (p. xvi).

Malafouris (2013), through Material Engagement Theory, challenges the plan-execution model of making. Making is not the application of a pre-formed intention onto passive material. Physical execution feeds back into procedure, domain knowledge, and sometimes vision. The potter throwing on the wheel is not executing an intention onto passive clay. “The constituents of the creative process are not to be found before or outside the throwing or the shaping of the pot. The constituents of creativity are in the throwing, in the shaping” (p. 213). The material is an active participant. “Trying to separate cause from effect inside the loop of pottery making is like trying to construct a pot keeping your hands clean from the mud” (p. 222). The action does not execute an intention so much as shape it.

The intention itself forms in and through the material encounter. The maker’s vision at step one of the cascade is not always fully formed before contact with the material. Material behavior partly shapes what the maker ends up wanting to make. The cascade therefore cannot be treated as a sequence of translations from a fixed intention. At the physical execution step, material behavior introduces element interactivity that could not have been predicted from the prior steps.

What CLT’s embodied extensions do not cover is the emergent, feedback-driven character of making that Malafouris (2013) names. Material behavior changes what gets made and sometimes what was intended, not as an error to be corrected, but as a constitutive feature of the process. The force of this claim varies by domain. In ceramics, woodworking, and textile work, material behavior is dense enough to reshape intention mid-process. In electronics, the plan-execution model more often holds. Unexpected circuit behavior and

component failure still introduce element interactivity at the physical execution step that the prior steps could not have predicted.

With the cascade and the four extensions, the framework can now analyze the cognitive load of making, in element-interactivity terms where CLT reaches and through the four extensions where making exceeds it.

3.2 LLM mediation: redistribution, not reduction

In a June 2024 interview on the Better Thinking podcast, Sweller said that “AI allows us to communicate with machinery in the same way we communicate with each other” (Nikolic, 2024, 1:16:50). The characterization names something structurally new for a tool. A calculator automates a computation and returns a result. A search engine indexes and retrieves. An intelligent tutoring system selects from authored content and presents it through scripted interaction within a fixed domain. By contrast, an LLM generates output across a wide range of domains, in response to natural-language description, through open-ended iterative conversation.

The LLM reaches every step of the cascade except physical execution. What follows maps where redistribution can occur, not where it typically does. Makers can consult LLMs at one step, a few, or across the whole cascade.

As Section 2.4 noted, the technology is advancing quickly. Several LLM limitations discussed in Section 2.5 are now partly outdated. GPT-3.5-era failures, including hardware design (Tafferner et al., 2023) and database access (Badini et al., 2023), have been partially addressed by tool-calling and web access. The text-only pipeline brittleness Makatura et al. (2024) reported, including its spatial reasoning deficits, is partly addressed by multimodal models that accept visual input.

Two patterns persist across current evaluations. First, the logical-physical performance gap holds (Song et al., 2025). Second, human verification is still required across domains, regardless of model generation.

3.2.1 Where LLMs enter the cascade

LLMs operate at every cascade step that can be carried in language: vision, decomposition, domain knowledge retrieval, procedure, and evaluation. The exception is physical execution, where body and material interact in real time.

At each step, I name what the LLM could support and through what mechanism, what new load it introduces, and how the support and the load land differently for a novice and for an expert in this domain. For a novice, the LLM often substitutes for the kind

of informal scaffolding a mentor, an experienced peer, or community knowledge would otherwise provide. Novice and expert are continuum endpoints, not a binary. The same maker may be a novice at textile work and an expert at electronics. I use the contrast throughout because CLT’s expertise effects depend on it. Scaffolding that helps a novice can hurt an expert (the expertise reversal effect), and the worked-example effect weakens as expertise grows (Kalyuga et al., 2003). The same kind of asymmetry shows up under LLM mediation. Using the novice/expert contrast makes it visible at each step.

Expertise is one of several variables. Outcomes also depend on the maker’s metacognitive orientation, the interaction pattern, and whether the artifact gets built and physically tested. A further axis that LLMs introduce is individual fit with linguistically mediated iterative work. Some makers do not think well in this mode regardless of expertise, engagement, or task. Domain reliability is a composite of training data coverage, codifiability (does the domain’s knowledge translate to text), and verification tractability (can the maker tell empirically whether the output is right). Domains whose knowledge already lives in text and standardized terminology, like electronics, rate higher on this composite than embodied crafts like ceramics.

“Support” means taking on a portion of the element interactivity at that step, not full replacement. The maker retains the priority structure and the integration across steps, and now carries the load of verifying the output. Support at every step is also contingent on the maker articulating the relevant context in language. That articulation is itself a load, named below as language bottleneck and prompt formulation load. The walk unpacks each load type, naming its definition, its lineage, and what the thesis transfers to making. Some recur across steps. One project-level load (intent monitoring) sits above the cascade and is named at the end of this section.

The walk that follows uses one running example. A maker is building an e-textile garment whose LED patterns respond to movement. The project’s parts span fabric and garment design, conductive paths, a motion sensor (accelerometer or gyroscope), a microcontroller, the LED array, power, code, and integration. It crosses textile work and electronics, so the same maker may be a novice in one part and an expert in another.

Step 1: Vision. The maker arrives at vision-formation with a vague image, “a jacket that lights up when I move.” Several competing requirements have to be held together: brightness against battery life, washability against component density, subtle against expressive aesthetics. Each requirement constrains the others, and the element interactivity is high. An LLM can externalize the requirements as text, name vague aesthetic and functional intent, and surface considerations the maker has not yet articulated. The interactivity of holding all of them simultaneously drops. The maker still owns the priority structure. The load that newly accrues is the language bottleneck. The pre-linguistic part

of the maker’s vision does not cross the interface. Vision is also where intent drift first becomes possible, though the cascade-spanning intent-monitoring load is treated after the walk.

For a novice, the LLM may close the vocabulary gap and surface unarticulated constraints. A vague vision is also easy to reshape, so intent drift may be cheap. For an expert, vision is likely more stable, and the LLM may function as a targeted sounding board, useful for externalizing competing requirements before priority-setting or pressure-testing a half-formed direction. The cost here is articulation. Tacit aesthetic and functional commitments resist translation to declarative language, and the LLM may offer standard frames that conflict with the maker’s specific intent.

I introduce the language bottleneck as the load type most centrally at Step 1. LLM mediation runs primarily through natural language. Drawing on Polanyi (1966), the maker knows more than they can articulate, and the LLM receives only the articulated tip. The language bottleneck appears in two forms in maker contexts. One is the vocabulary gap, in which the maker lacks the domain terms to describe their need. The other is articulation cost, in which practitioner knowledge has become procedural and resists declarative restatement. The latter has a counterintuitive consequence. An expert may communicate less usefully than a novice describing naively. Articulation cost is highest in crafts whose knowledge lives in hand, eye, and material. Ceramics, woodworking, and textiles are examples, where the practitioner’s competence has never had to take linguistic form. It is lowest in domains like electronics, where vocabulary, schematics, and component names already do most of the descriptive work. Multimodal input partially bypasses the bottleneck on the input side. The maker can show what they cannot say, but the LLM still answers in language. The bottleneck cuts across every cascade step where the maker prompts, and is distinct from the modal gap (developed at Step 5), which sits at language-to-action rather than at articulation itself.

Step 2: Decomposition. Vision then has to be decomposed into parts with interdependencies. Power feeds the controller, the controller reads the sensor and drives the LEDs, conductive paths route between them, and everything is sewn into a wearable structure. The element interactivity of holding many sub-components and their relations simultaneously is high. An LLM can produce a structured breakdown of those parts and their interdependencies, taking that interactivity onto the screen rather than into working memory. The loads that newly accrue are verification (is this decomposition complete and well-fitted to the project?) and context management (does the model have the project context it needs, in a form it can use?).

For a novice, the LLM produces a structured breakdown that may absorb interactivity the maker cannot chunk unaided. The boundary cuts at verification. Completeness and

fit cannot be verified without the schemas to assess them, and the performance-learning dissociation risk is real if the maker does not return to question the breakdown. For an expert, schemas already chunk much of the decomposition. The LLM extends reach into adjacent parts of the project the maker is less familiar with and offloads boilerplate detail work. At the maker’s core domain, the LLM’s generic decomposition may be worse than the expert’s own, and the cost of articulating project-specific constraints is real.

A making project rarely fits inside one session, which means the maker has to deliver project context to the LLM repeatedly and in a form the model can use. Both the content and the format of that context affect what the model attends to. Liu et al. (2024), recalled from Section 2.4, document the lost-in-middle pattern. Information placed in the middle of a long context is used less reliably than information at the start or end, so even a large context window does not dissolve the load. The load is extraneous, imposed by the tool’s session architecture rather than by the making task itself. The form the load takes depends on the model’s memory features. Without them, the maker rebuilds project context at each session. With them, a different sub-task appears. The maker has to check that what the model retained is still useful and update or wipe what no longer is. I call this context management load. It covers two things: supplying project context in a form the model can actually use (structured markdown rather than scanned PDFs, summaries rather than raw transcripts), and tracking what state the model currently holds so the maker knows what to repeat, update, or wipe.

Step 3: Domain knowledge retrieval. The project crosses domains: conductive thread resistance per length, fabric heat tolerance, LED current draw, microcontroller pin mapping, accelerometer threshold detection, voltage drop, wash-cycle survivability. Each is a specialized fact that may or may not sit in the maker’s prior schemas. An LLM can return probable answers to factual queries and map vocabulary that reduces search cost across unfamiliar terrain. The element interactivity of holding many cross-domain facts simultaneously drops. What surges in its place is verification. Verification load is at its highest here.

For a novice, the LLM can give vocabulary that enables further self-directed learning. The same gap that makes the LLM useful makes its output hardest to verify, and vocabulary acquired without underlying schemas can produce the illusion of competence. For an expert, the affordance may be more targeted. Tedious lookups for specs, references, and conversions get offloaded, and edge cases or adjacent domains the expert is exploring can be extended through dialogue. At the expert’s core domain the returned content often explains what is already automated, and articulation cost rises because procedural knowledge must become declarative to be queried.

By verification load I mean the cost of evaluating whether LLM output is correct for this context, these materials, this tool. Yao and Fan (2025), recalled from Section 2.6, demonstrate it psychometrically as Critical Evaluation Load in LLM-assisted writing. Tankelevitch et al. (2024) identify the processing-fluency heuristic at its core. People infer accuracy from how easily and quickly information is processed, so fluent LLM output reads as accurate beyond what its accuracy warrants. The transfer to making is the verification paradox. To check the answer the maker needs the knowledge the answer would have supplied. If the maker had that knowledge, the query might not have been needed. Physical making partially mitigates the paradox. The artifact either works or it does not, and a wrong procedure typically fails to produce a working artifact. The case the paradox does not catch is confident plausible wrong output where the maker has no schemas to detect the error and the artifact’s failure mode is ambiguous. Engaged interaction can make this worse when the maker elaborates the wrong schema in dialogue. The decoupling between artifact success and schema formation also explains the performance-learning dissociation. A working artifact can form while none of the schemas needed to modify or transfer it do so. Verification load appears at decomposition, procedure, and evaluation as well. Step 3 is where verification load is highest, because retrieval is where the maker most often lacks the schemas to evaluate returned answers.

Step 4: Procedure. Retrieval has supplied the facts. The maker now plans the action sequence: design fabric pattern, choose components, sketch circuit, write LED-pattern code, breadboard test, transfer to fabric, sew conductive paths, mount components, connect power, wear-test. An LLM can produce that sequence in full. The output has the structure of a worked example. The means are provided rather than searched for, which bypasses means-ends analysis at this step. Two loads accrue. Prompt formulation appears here, because getting a usable procedure requires the maker to specify what they need in a form the model can act on. Modal gap also enters, because the procedure has to translate into hands, fabric, and thread tension.

For a novice, a correct procedure activates the worked-example effect (Sweller, 1988; Sweller & Cooper, 1985), which CLT predicts is most powerful for learners without prior schemas. Schema construction can proceed by studying a worked solution rather than by means-ends search, and the LLM tailors the example to this specific project. The boundary cuts at correctness. A plausibly wrong procedure is undetectable without the schemas to flag inconsistencies, and the worked-example effect requires the example to be correct. For an expert, this may be the strongest case for LLM support. Verification is likely cheap because pattern recognition is fast, and the LLM can produce procedures for moves the expert wants to try but has not yet automated, while routine procedures get offloaded for a quick check. The cost is redundancy. Over-explanation in already-automated areas is

irritating rather than informative, and the expert often prefers their own procedure with selective LLM cross-checking.

I propose prompt formulation load as the next type. Getting useful output requires more than putting a need into words. It requires a working mental model of how LLMs function. Yao and Fan (2025) demonstrate Prompt Management Load psychometrically in LLM-assisted writing. Yang et al. (2025) find that LLMs correctly infer unspecified requirements about 41% of the time, and these inferences degrade twice as often as explicit specifications when the model or prompt changes. The specification cost stays with the maker. Lindley and Whitham (2025) describe prompt craft as a craft-like practice where understanding accumulates through non-linear experimentation. In CLT terms, this is biologically secondary knowledge in Geary’s (2008) sense. LLM-specific knowledge has to be learned and does not transfer from prior tool experience. The load has two components. One is linguifying the need, shared with the language bottleneck at Step 1. The other is LLM-specific framing knowledge, distinct from any other maker skill. A maker can have deep domain expertise and still face high prompt formulation load if they have no model of how the tool works. Making expertise and LLM-interaction expertise are independent axes.

Step 5: Physical execution. Procedure has produced an action sequence. The maker now performs it: sewing conductive thread where knot placement and tension determine continuity, soldering near fabric, fitting components into the garment, choosing between hand-stitching and machine work, managing fabric drape with components attached. The LLM does not reach this step. Body and material run it in real time, drawing on the tacit knowledge and material engagement Section 3.1 established as beyond what language can carry. Agentic architectures extend the LLM’s output into actuator control, but the input remains articulated language, and physical execution lives in what was never articulated. What I call the modal gap is the cost of every transition between the LLM’s linguistic output and the maker’s physical action. It is at its widest here, where linguistic instruction has to be held in working memory while the maker attends to physical action.

For a novice, the modal gap is likely wide. No tacit knowledge bridges the linguistic procedure to physical action, and every transition from text to hands is a fresh translation. The bound on this cost is material feedback. The artifact’s response to action provides a ground truth the LLM cannot supply, and iteration is what closes the gap. For an expert, the modal gap is likely narrower, because tacit knowledge bridges registers. An articulation mismatch still persists. The LLM’s description of procedural knowledge may not match what the body knows to do, and the expert may find linguistic instruction disrupting where their own action is fluent.

Two mechanisms produce the modal gap. First, parts of making knowledge are pre-linguistic. Polanyi (1966) names the tacit dimension of skilled knowledge. Malafouris (2013) extends this to the material encounter. The part of the maker’s knowledge that lives in hands and material never crossed into language in the first place. Second, the transition demands attention in two registers at once. Holding linguistic instruction in working memory while attending to physical action loads two systems simultaneously. Multimodal input narrows this on the input side, but the LLM’s reply still arrives as language. The gap tracks how much of a step’s knowledge codifies into language. It is deepest in embodied crafts like ceramics, woodworking, and textiles, where competence lives in hand and material. It does not spare electronics. The parts that reduce to code or schematics translate cleanly, while physical implementation does not. Song et al. (2025) document this within electronics. Claude 3.7 Sonnet scored 65.6% on logical circuit generation but only 13.6% on physical breadboard layout. The drop for o3-mini was milder, from 66.0% to 45.2%. Tedre et al. (2023) document the parallel gap in craft, where AI generates the visual design while the materials and skill to realize it physically remain the maker’s, and Makatura et al. (2024) report spatial reasoning deficits compounding across the fabrication pipeline.

Step 6: Evaluation. Execution produces an artifact, and evaluation tests it. Does the circuit work? Do the LEDs respond to the chosen movements at the right thresholds? Does the pattern look as intended? Does the garment drape correctly with the hardware on it? Will it survive a wash? An LLM can support evaluation in two distinct moves. Before the artifact is judged, by structuring the criteria to check and naming what to inspect. After the artifact exists, by interpreting photos of it with multimodal input. The second move is partial. The LLM evaluates a representation of the artifact, not the artifact itself, and the maker remains the only party in physical contact with the result. The loads that apply at this step are verification and language bottleneck. For a novice, the value is the criteria vocabulary, naming what to check that the maker may not have known to inspect. For an expert, the LLM adds little to the core judgment, because the maker already holds the evaluation criteria. Its value is more likely generative. Because LLMs generate possibilities broadly, it may surface a failure mode, a check, or an alternative direction the expert had not considered. The risk is over-weighting LLM confirmation when one’s own judgment is already sound.

I name intent monitoring as a project-level load that sits above the cascade. The maker checks whether the project still matches their original intent. Each LLM suggestion can shift the project slightly, and individually those shifts can be useful. The cost is that the shifts accumulate unnoticed turn by turn, and the maker may register the cumulative drift only late, when the project has already moved somewhere they did not intend. Drift can begin at vision-formation and continue across the rest of the cascade. The load is the

ongoing comparison between current direction and original intent, run frequently enough to catch drift early.

3.2.2 The redistribution claim

Across the six cascade steps, I see one pattern recur. Load does not disappear when an LLM enters the cascade. It shifts. Load once concentrated in the domain task redistributes across that task and the interaction task. The maker’s problem-solving remains the maker’s. What changes is which cognitive operations can be offloaded and what new demands emerge from the interaction. Linguistic generation becomes evaluation. Domain recall becomes output evaluation. Physical execution remains the maker’s.

Redistribution can raise total load, not just shift it. New loads appear at the maker-LLM interface that did not exist before, and they sit on top of whatever the maker already carries elsewhere. Five of the six are extraneous in the post-2010 sense, imposed by the tool rather than by the material. Intent monitoring is the exception. It draws on goal representation and executive function constructs that exceed CLT’s experimental scope. Whether the net is a gain or a loss depends on the maker, the project, and the interaction pattern.

The six load types fall into two groups. Three intensify something that already existed. Verification load was already part of any encounter with a manual or forum post, and the LLM raises the cost through per-turn volume and the gap between fluency and accuracy. Modal gap load was already present in any text-instructed physical action. The recipe writer chose what to convey and tested it, but the LLM did neither, and the gap widens. Intent monitoring was already at work in any collaboration with a mentor or peer. Every prompt is now a possible drift point, and the fluency of suggestion makes drift cheap to absorb. The other three loads are new. The language bottleneck did not exist because no prior maker tool required the maker to articulate in language before it could act. Prompt formulation load did not exist because no prior tool required a mental model of probabilistic generation. Context management load is new because no prior maker tool needed its user to track its state. The maker’s prior tool intuitions cannot help with the second group. No prior maker tool asked them to articulate tacit knowledge, hold a model of probabilistic generation, or track a second agent’s state. Table A.2 in the appendix collects the six load types, the mechanism of each, and how the load may differ for novices and experts.

The redistribution claim is conceptual. Section 3.3 names what empirical studies of LLM-mediated making would require. Whether the freed working memory at the cascade steps where the LLM operates goes toward schema formation depends on the interaction pattern, not on whether the LLM was used at all.

3.2.3 Learning implications

The redistribution claim and the six load types describe how cognitive load is distributed in LLM-mediated making. They do not yet say what determines whether learning occurs.

The engagement criterion distinguishes LLM use that supports schema formation from LLM use that does not. The independence fallacy, the assumption that tool use precludes learning, is rejected. Both turn on how the maker engages with the LLM across the interaction, not whether they used it at all.

The engagement criterion

A standard concern in maker education is that LLMs remove struggle that is necessary for learning. This belief comes out of constructivist traditions, including constructionism's Piagetian roots (Papert, 1980), Dewey's emphasis on learning by doing (Dewey, 1938), and the maker movement's failure-as-learning ethos. Blikstein (2013) documents fab lab learning as built on cycles of failure and frustration. The framing sits within this constructivist tradition.

From a CLT perspective, the assumption that struggle is necessary for learning faces two main problems. First, Sweller's early research on problem solving (Sweller & Levine, 1982) shows that effortfulness and schema formation can come apart. Conventional problem solving using means-ends analysis loads working memory heavily yet leads to poor schema formation. Goal-free problems and worked examples often outperform conventional problem solving because they bypass means-ends analysis. Struggle is not the mechanism that produces learning. Second, the pre-2010 CLT framework tried to preserve the intuition that struggle is beneficial through the construct of germane load (Sweller et al., 1998). The idea was that effort generates a load that directly drives schema formation, but germane load was inferred post hoc from the learning it was meant to explain, making the construct circular (de Jong, 2010). In the post-2010 CLT framework (Sweller, 2010), germane load is no longer an independent source of load. With that revision, CLT no longer offers a mechanism by which struggle as such causes learning.

When extraneous load drops, working-memory capacity becomes available for processing the intrinsic element interactivity of the material, which is schema formation. I propose the engagement criterion to name what could enable freed working memory to be used for schema formation. The question is whether the maker remains engaged with the conceptual structure of the problem.

This criterion distinguishes two patterns. In both, the LLM produces an output that bypasses the maker's own means-ends search. They differ in what the maker does with that output. In the first pattern, engagement is maintained. The maker asks why the

output works, tests edge cases, modifies the approach, and integrates ideas from other sources. In such cases, schemas are likely to form. In the second pattern, engagement collapses. The maker copies and pastes the output and moves on. In such cases, schemas do not form. If a maker’s goal is simply to obtain a printed part as one component of a larger build, and not to learn 3D design, then failing to engage with the design process is not a failure in terms of their goals. The criterion evaluates engagement relative to the maker’s learning goals, not against an externally imposed curriculum. The empirical illustrations from Enghardt et al. (2024) clarify this scope. Participant P4’s reflection, “I feel like I didn’t [do any] work” (Section 3.4), is a case where the engagement criterion is not met. Learning was the explicit intent, yet the interaction pattern did not support conceptual engagement. By contrast, participant P8 notes that “it’s fulfilling in different ways: ...instead of understanding the domain, you become more fulfilled in that you get the task to work” (Section 3.4). This is a scope-limit illustration. P8 recognizes and chooses a shift from understanding the domain to achieving task completion. In that state, the engagement criterion does not apply, because learning that domain is no longer the point.

The framework operates across three temporal scales: the turn level, the project level, and the trajectory level. At the turn level, the question is whether the maker is engaging with the conceptual structure of this specific output. At the project level, the question is whether the overall interaction pattern across turns in the project shows movement toward conceptual engagement. This project-level engagement is distinct from intent monitoring. Engagement asks whether the maker works with the problem’s conceptual structure. Intent monitoring asks whether the project’s direction is still the maker’s own. A maker can stay engaged turn by turn while the cumulative direction drifts from the original intent. At the trajectory level, the question is whether, across multiple projects, the maker is accumulating the kinds of exposure that enable schema formation. The temporal scale matters because it determines what counts as evidence. A single copy-paste at the turn level might be coded as a miss. There is no visible engagement. The same action can still form part of a productive trajectory if, in a later project, the maker engages with a failure, revisits similar patterns, or begins to ask different questions. What looks like non-engagement at one moment can become useful substrate for later learning.

Even without engagement, repeated copy-paste produces pattern encounters. These could support schema formation later, when the maker engages with a related failure or sees a similar pattern in a different context. Without later engagement, the encounters would not convert. Makers who use the LLM as an explanation scaffold (asking why something works, what would break it, and how to verify it) build understanding through the interaction itself. Makers who use the LLM only as an answer generator accumulate outputs that never convert. Whether schema formation occurs depends on the interaction pattern.

This temporal framing does not soften the framework’s concerns. The risks identified, such as unresolvable verification load, the formation of incorrect schemas from plausible hallucinations, and the dissociation between performance and learning, remain acute. These risks are pronounced at particular phases of a learner’s trajectory, in unfamiliar domains, and for makers who never shift from using the LLM as an answer generator to using it as an explanation scaffold.

While the engagement criterion is a conceptual proposal, it suggests possible empirical proxies. Within a chat log, one indicator is the extension and deepening of question formulation over time: whether questions become more probing across turns, whether the maker integrates material from other sources into the conversation, and whether they challenge the model’s output or request explanations rather than just further answers. These behaviors could be the observable side of the metacognitive orientation the criterion names. Turning these ideas into robust measures is a task for future empirical work.

The independence fallacy

I name as the independence fallacy the assumption that using a tool means a person is not learning, or that the goal must be unaided performance. Rejecting this fallacy makes tool-aided learning legible as learning. The key question becomes what was offloaded to the tool and what the learner engaged with, a question that applies across expertise levels. Hutchins (1995) showed that cognition is distributed across artifacts and the environment. The learning question is what specifically was offloaded and whether it supports schema formation at this stage. The same assumption underlies use/non-use research designs, which treat LLM use as the independent variable and cannot register what the maker engaged with inside the condition. Section 3.3 develops this implication.

Bandura (1977), writing in the context of therapeutic behavior change, challenges the assumption that people must “labor unaided” to develop a sense of competence, calling this view “open to question on both conceptual and empirical grounds” (p. 201). External support enables capability development first. Aid is removed later, once the learner can succeed without it.

3.3 Methodological implications

3.3.1 The methodological claim

In Sections 3.1 and 3.2, I built a conceptual framework for LLM-mediated making: the cascade of translations, the redistribution of load when the LLM enters, the six new load types, the engagement criterion, and how expertise reshapes each. This section considers what those claims imply for empirical study. The argument is conceptual. It does not

design a study, but it identifies what a study would have to capture and what would falsify those concepts.

Three variables shape load redistribution: the activity (cascade step, materials, target artifact), the maker’s existing schemas, and the interaction pattern with the LLM. On the account I have built, all three jointly influence where load lands and which translations the LLM absorbs. A use/non-use design holds the activity constant across conditions and collapses the interaction pattern into a binary value of LLM used or LLM not used. The internal structure of the interaction pattern (verification behavior, conceptual engagement, intent monitoring) is what the framework identifies as carrying the explanatory weight. A use/non-use design does not measure it.

Stadler et al. (2024), one of the few peer-reviewed studies of CLT applied to LLM use, examines university students researching a contested scientific topic using ChatGPT or Google. Kosmyrna et al. (2025), which received public attention, examines essay writing across four sessions with ChatGPT, Google, or no tool. Neither studies making. Both are useful here for methodological reasons. They have two issues common in current research on LLM-mediated cognition.

The first concerns the comparison. Both studies compare LLM use against either no LLM use or search engine use. This treats the LLM as a uniform condition. The variable that shapes redistribution sits inside that condition: which cascade step the LLM absorbs, whether the maker verifies the output, what conceptual structure the interaction engages with. Two participants who both used an LLM can differ on each of these. A design that records both as “used LLM” cannot register the variation that shapes redistribution. The comparison also assumes a baseline cognitive science has questioned for three decades. Distributed cognition (Hutchins, 1995) treats human cognition as inherently distributed across artifacts and other minds. There is no neutral no-tool state. The substantive comparison is between one scaffold and another.

The second concerns the measurement. Stadler et al. operationalizes cognitive load through the Klepsch et al. (2017) self-report scale, developed within tripartite CLT. CLT was given tripartite form by Sweller et al. (1998), distinguishing three independent types of load: intrinsic, extraneous, and germane. By the late 2000s, germane load was contested as an independent construct (de Jong, 2010). Sweller (2010) reformulated CLT to remove germane load as an independent source and center the framework on element interactivity. The thesis uses this post-2010 framework. The instrument and hypotheses of Stadler et al. inherit the older tripartite framework. The study’s H1.3 hypothesizes lower germane load in the LLM condition, an outcome not coherent under post-2010 CLT.

What that scale captures, even within the tripartite tradition, is the participant’s perceived difficulty of integrating information during the task. Working memory load in the technical

sense, the allocation of limited resources to processing intrinsic element interactivity, is not directly observable through self-report. Stadler et al.’s discussion reads lower self-reported difficulty in the LLM condition as evidence that LLM use “might not have engaged the deep learning processes as effectively” (Stadler et al., 2024, p. 5). Engagement and deep learning processes are not what the instrument registered. The instrument registered perceived integration difficulty.

Kosmyna et al. (2025) operationalize cognitive engagement with the dynamic Direct Transfer Function (dDTF), a measure of connectivity between EEG electrodes. They find the strongest connectivity in the group that wrote with no tool, less with a search engine, and the least with an LLM, and read this as deeper engagement in the no-tool group. The problem is not the measurement but the inference. Two steps overreach. First, electrodes are not brain regions. Without source localization, a link between two electrodes is not a link between the cortical areas their names suggest. The paper still labels two electrode sites “left parietal” and “right temporal” (p. 136) and a third a “major convergence hub” (p. 78). These are claims about cortex that electrode data do not support. The authors acknowledge this localization limit (p. 142) but use the labels regardless. Second, connectivity is a statistical pattern, not a mental state. Strong connectivity in the no-tool group is as consistent with effort, difficulty, or frustration as with deep engagement. By the thesis’s own dissociation between effort and learning (Section 3.2), even effort would not show the deeper learning the authors claim. There is also a simpler reading. The no-tool group has nothing to lean on, so it does all the work itself, and higher connectivity follows from that alone. Removing any tool would do the same. The one result that points to accumulation comes from a single crossover session, where 18 of the 54 participants returned and rewrote a topic they had already written. Tool switch and rewrite are confounded. The authors call this finding preliminary (p. 141). “Cognitive debt” is coined for the paper, not drawn from a prior framework, and the long-term costs it names (diminished critical inquiry, vulnerability to manipulation, decreased creativity) were not measured. A one-time comparison between groups cannot show a deficit building over time, and the only sign of persistence is that confounded session.

Two studies do parts of this well. Lepine et al. (2026) measure cognitive load from transcripts at turn level in AI-assisted financial knowledge work, find expertise moderates load dynamics, and show that extraneous load is path-dependent across turns. Their turn-level measurement captures interaction-pattern variation directly, even though the setting is not making and modal gap is therefore out of scope. Yao and Fan (2025) develop and validate a Cognitive Load Scale for AI-assisted L2 Writing. Their scale isolates distinct AI-interaction load factors, among them Prompt Management and Critical Evaluation, and validates them psychometrically rather than inferring load from a proxy measure.

A study designed to test these claims would look different. At the level of comparison, studies should specify which kinds of LLM use are being compared: verification-heavy versus verification-light interaction, conceptual engagement versus output integration, LLM intervention at one cascade step versus another. At the level of measurement, instruments should be matched to the constructs the conclusions claim. A claim about engagement requires a measure of engagement, not a measure of perceived difficulty. A claim about schema formation requires evidence of transfer, not a count of expert-validated argument points covered.

3.3.2 Falsifying the constructs

The six new load types and the engagement criterion from Section 3.2 each assert a falsifiable relationship. I state each relationship and what would falsify it, leaving the design of tests to empirical work.

Three load types intensify pre-LLM analogues. Verification load asserts that evaluating LLM output for fit imposes working memory demand whose magnitude depends on the maker’s domain schemas, falsified if expertise does not predict verification cost on identical tasks. Modal gap load asserts that translating LLM linguistic output into physical action imposes a cost beyond the physical task itself, falsified if that translation imposes no additional cost. Intent monitoring asserts that tracking whether the project’s direction remains the maker’s own imposes a project-level cost distinct from ordinary project planning, falsified if LLM-mediated drift imposes no cost beyond planning.

Three load types are new at the maker-LLM interface. The language bottleneck asserts that LLM mediation introduces two costs: a vocabulary gap when the maker lacks domain terms, falsified if domain vocabulary does not predict the usefulness of LLM-mediated output, and an articulation cost when their knowledge is tacit, falsified if tacit knowledge translates into language as readily as explicit knowledge. Prompt formulation load asserts that effective prompting requires LLM-specific interaction knowledge in addition to the cost of linguifying the need, falsified if LLM-specific knowledge separate from domain vocabulary does not predict prompt effectiveness. Context management load asserts that the maker bears a cost from holding both project state and a model of the LLM’s current context simultaneously, falsified if maintaining a model of the LLM’s current context imposes no demand beyond the project’s intrinsic complexity.

The engagement criterion asserts that, when extraneous load is reduced, schema formation during LLM-mediated making depends on the maker’s engagement with the problem’s conceptual structure during interaction. It is falsified if schema formation reliably occurs without such engagement.

3.3.3 What a study of LLM-mediated making would have to capture

Making translates language into physical action on materials with tools. That translation has historically been taught through apprenticeship and mentorship, both unevenly available. LLMs now operate at the same linguistic register as mentors, explaining and troubleshooting in natural language. Whether LLMs widen access to making, preserve the learning making supports, or undermine that learning, is open. The answer has consequences for who makes and what they learn.

Making fits the framework’s structure in ways essay writing and coding do not. The vocabulary gap is structural. Every domain has its own technical terms for materials, tools, and techniques, and a maker who has not yet acquired them has limited access to the LLM’s domain knowledge. In essay writing or general research the maker and the LLM share a common register. In making they often do not. The modal gap is constitutive. LLMs produce language and making ends in physical action on materials with tools, so every transition between linguistic instruction and physical execution is a cost that tasks staying in language do not show. Unlike essay writing, which has only a short linguistic chain, making contains all six cascade steps. The last, physical evaluation, is where verification carries material consequences. The circuit lights up or does not, the seam holds or does not, the fired piece survives the kiln or cracks.

Making is also well suited to empirical study. Interaction patterns leave multimodal traces. Two makers using identical LLM advice produce visibly different objects, and the record of their work includes chat logs, photographs, the artifact itself, and the marks left by tools and materials. The interaction-pattern variation predicted in Section 3.2 is observable here. Domains with a single data channel miss it. The domain of making is internally varied. Electronics, ceramics, woodworking, sewing, and baking differ in their balance of declarative and procedural knowledge, their verification mechanisms, and their relationship to language. This enables cross-domain comparison while holding the LLM constant. The domain-dependence claims from Section 3.2, including the graduated articulation cost, become testable here in a way single-domain comparisons do not allow.

Two measurement requirements follow. The interaction pattern with the LLM has to be captured at turn level. Verification behavior, conceptual engagement, and intent monitoring vary turn by turn within an LLM-mediated session. A study that records only whether the participant used an LLM, not how they used it, cannot capture the variation that shapes where load lands. Each construct also needs an instrument that captures it. Chat logs are primary data for interaction patterns. The decision log is a candidate instrument for intent monitoring, in which the maker records at key decision points what direction was taken and why. The artifact, photographs, and tool marks form the material

record. A claim about schema formation has to rest on transfer evidence, not on subjective task ratings.

Two sampling requirements follow as well. Expertise and domain must both vary. The maker’s existing schemas shape redistribution, and the modal gap, the vocabulary gap, and verification costs differ across domains. Studies with novice and experienced makers, across at least two domains such as electronics and ceramics, can test those predictions, though this variation may require multiple studies rather than a single design. The study also has to sample across months, not sessions. Single-session designs cannot detect the trajectory effects predicted in Section 3.2. Kosmyna et al. (2025) sampled at roughly the right scale but confounded tool switch with topic familiarity. A longitudinal design that holds topic constant while varying interaction patterns within the LLM condition is the minimum.

No study of LLM-mediated making in these terms appears in the literature reviewed. The conceptual account makes the requirements visible.

4. Tool in Use

Chapter 3 built a conceptual framework. Its central claim is that LLMs redistribute cognitive load across the cascade of translations and introduce new load types. This chapter grounds that framework in concrete maker practice. The grounding is illustrative. This chapter draws on public first-person accounts of LLM-mediated making posted on blogs, Substack essays, forums, and Reddit between 2023 and 2026. Blogs and Substack essays are close-read as anchor cases. Forum and Reddit threads are cited alongside. The accounts are read for moments where makers describe the same phenomena the framework names, without using the framework’s vocabulary. Where they do, the framework’s terms describe what the maker has already noticed. Where they do not, the chapter notes the absence rather than imposing the construct.

The 2023-2026 window spans a shift in LLM use. Adoption diversified from coding into home maintenance and personal queries (Chatterji et al., 2025; Massenkoff et al., 2026). The cases are text-based interactions across this period.

This chapter illustrates the framework against this material. Section 4.1 describes the selection method. Section 4.2 opens by mapping where reflective accounts cluster across making domains, a distribution the language bottleneck explains. It then works through four constructs: verification load, modal gap, redistribution, and the engagement criterion with the independence fallacy. The first three together make up the redistribution argument. Redistribution is the claim, verification load is the load it most centrally produces, and modal gap is the structural boundary it cannot cross, illustrated through the cascade of translations. Verification load and modal gap are presented before redistribution, the claim that frames them, because the chapter builds the claim from the phenomena it produces. The engagement criterion is a separate proposal about what supports schema formation. The remaining constructs from Chapter 3 (prompt formulation load, context management load, and intent monitoring) operate within this structure and surface in the sections where they apply. Intent monitoring surfaces in the redistribution section at project scale. Forum data illustrates these constructs unevenly.

4.1 Method

This chapter uses first-person writing (independent blogs, Substack, Medium, LinkedIn long-form) and forum threads where makers describe their LLM-mediated making. Google searches combined each of four keywords (ChatGPT, Claude, Gemini, AI) with each

of 13 domain terms (sewing, knitting, crochet, quilting, ceramics, 3D printing, electronics, leatherwork, pottery, woodworking, metalwork, jewelry, glass). The 13 cover the main physical-craft domains across material families (textile, clay, wood, metal, leather, glass) plus electronics and digital fabrication. They span the range from crafts whose core knowledge translates easily into language to crafts where it resists articulation, the spread Section 4.2.1 examines through the language bottleneck. A forum pass used the query “ChatGPT OR Claude OR Gemini OR AI” on eight craft and maker subreddits (r/arduino, r/3Dprinting, r/prusa3d, r/homeautomation, r/woodworking, r/sewing, r/crochet, r/Pottery) and on specialist forums (Arduino, Prusa, LumberJocks, PatternReview, Practical Machinist, MIG Welding). Subreddit results were sorted by relevance and then comment count. The top 20 results per platform were checked against the criteria below. Because Google personalizes and re-ranks results, these searches are not reproducible and describe how the material was gathered, not a replicable protocol or representative sample.

Each keyword biased the material. ChatGPT, the first LLM in wide public use, returned the widest spread and the most fiber-arts writing. Claude returned mostly developer-makers in code-driven 3D printing. Gemini and the bare “AI” keyword returned thin or image-generation-heavy noise.

A second pass searched specifically for novices. The first pass found few. New search terms were added on the same platforms. Google searches paired each LLM keyword with first-contact phrases (“first time,” “no experience,” “never coded,” “had no idea,” “learn to”) combined with the same domain terms. A separate run searched Reddit, YouTube, Instructables, Substack, and Instagram, with the same phrases. Tech-press first-person experiments were added where the writer says they had no prior experience. A piece counts as novice when the author has no prior knowledge of the target domain. Some makers are expert in one domain and novice in another, and the chapter counts each as a novice in the domain they enter through the LLM. This classification has a limit. Makers who enter physical making through code work in the representation the LLM reads and writes most fluently. They are novices in the target craft but not in the medium the interaction runs through.

Included accounts are first-person and describe a specific maker project using an LLM in a physical-craft domain. They were published 2023-2026. Inclusion is by activity, not by maker identity. Excluded are vendor marketing, news and third-party coverage, pieces whose AI use is only image generation, software-only projects with no physical artifact, and developer projects that build tools for makers without documenting the maker’s perspective.

Each construct subsection anchors on one close-read blog post and draws on one or two more as support. Forum threads play a supporting role. Some are summarized individually where they illustrate a construct, others are cited collectively across platforms, but none is close-read as an anchor case and posters are paraphrased rather than quoted, following the ethics protocol below. In total the chapter draws on 18 bylined first-person accounts and nine forum and Reddit threads, with the Hackaday 2025 year-in-review and a single Instagram post cited at community scale.

Several limits shape what the chapter can and cannot show. A blog post is a curated retrospective, not a transcript. Two filters compound on top of that. The first is a knowledge filter. To write reflectively about a craft, a maker needs the schemas to know what matters in the practice. Novices do not yet have those schemas. The second is a writing filter. To publish that writing, the maker has to be a writer. Many makers are not. These accounts therefore over-represent experienced makers who also write. The second pass tried to correct this with novice-targeted searches. It worked only partially. Novice voices appear most often in forum question threads (r/knittinghelp, the Arduino Forum support-burden discussion) and in tech-press first-person experiments (How-To Geek, Wokwi). They appear rarely as standalone essays. The 2023-2026 window covers only the early years of widespread commercial LLM use. Model behavior shifted across that window, so the chapter reads cases for structural features of LLM interaction rather than version-specific behavior. The chapter cannot ask follow-ups, resolve ambiguity, or check interpretations with authors.

Interviews are the obvious alternative. They were set aside because the framework's constructs do not yet have operational measures (Chapter 5 names this as future work), so interview prompts would either presuppose the constructs and lead the respondent or stay too generic to be informative. Public writing carries its own selection biases, but the maker chose what to say on their own.

The material is publicly posted writing by identifiable people. Use as observational evidence follows the Association of Internet Researchers (AoIR) guidelines (Franzke et al., 2020). Blog posts are quoted verbatim and attributed to the author, treated as published work with named bylines and explicit publication intent. Forum posts are not. The two are treated differently because the venue sets different expectations. A bylined author writes for publication and for scrutiny. A forum poster contributes to an exchange that carries an expectation of relative privacy even on a public platform, so AoIR treats a post's public availability as insufficient warrant to quote or name the poster. Forum posters are paraphrased so a reader cannot find the original post by searching the words, and individuals are not linked back via URL or screen name, following AoIR's disguise recommendations.

This chapter’s reading carries a risk of confirmation bias. Reading maker writing for where it names framework constructs can collapse into matching any phrasing to any construct. Five conditions distinguish inside-naming from confirmation bias, and the chapter holds itself to all of them. First, the framework was built in Chapter 3 before these accounts were assembled, so convergence is not the framework being shaped by what was read. Second, the framework predicts asymmetric appearance. Verification load, for example, should surface as named friction in expert prose and as a higher-cost external process (testing, cross-referencing, asking forums) in novice prose. The accounts are read for this asymmetry, not just for appearance. Third, lexical specificity is required. Vague gestures (“I had to check it”) are not inside-naming. Sharp distinctions (“force multipliers, up to a point”) are. Fourth, cases where the framework does not apply, or where the maker does not reach for its vocabulary, are named as such rather than absorbed into the supporting set. Fifth, convergence across independent voices in different crafts and at different expertise levels is the structural argument. A single-case match is only suggestive.

4.2 The constructs in cases

4.2.1 Distribution of accounts across making domains

Current reflective accounts of LLM-mediated making cluster where the artifact can be specified in language or code: fiber arts (knitting patterns, sewing instructions), 3D printing via CadQuery or OpenSCAD, Arduino or ESP32 (a low-cost microcontroller with Wi-Fi) electronics, and a small cluster in ceramics. Accounts do not surface in forging, glassblowing, or the tactile portions of woodworking and leatherwork. Woodworking and metalwork appear only in forum chatter on computational sub-tasks like cutlists, G-code (machine-readable movement instructions for CNC tools), and calculations. The language bottleneck construct from Section 3.2 offers a structural account. In crafts whose core knowledge resists linguistic articulation, LLM mediation may not enter the workflow, or may produce little the maker can describe in writing. The writing-culture filter named above compounds this.

4.2.2 Verification load

Verification load is the cost of evaluating whether LLM output is correct for this context, these materials, this tool (see Section 3.2). The cost is conditional on expertise. The expert resolves it quickly. Domain schemas reduce element interactivity during evaluation, so contradictions register without exhausting working memory (Sweller, 2010). The novice faces higher cost. Without domain schemas, verification requires external routes and may still miss errors the maker does not know to look for.

Lanja Khon-Engheim is a knitter who has published many sock patterns and has 25 years of sock-knitting experience (Khon-Engheim, 2025). The article tests whether AI will take Khon-Engheim’s job by asking a free, logged-out ChatGPT-4o to write a sock pattern. The first prompt requests a plain vanilla 60-stitch sock with fingering yarn, a heel flap, and a wedge toe. The output is a workable pattern with minor inconsistencies (needles divided as 3 in one section and 4 in another) that, in Khon-Engheim’s own assessment, a moderately experienced sock knitter would follow without difficulty. Khon-Engheim then asks ChatGPT to rewrite the cuff as a 2x2 cable. The model’s output on the cable cuff fails repeatedly. It does not write correct 2x2 cable instructions. When corrected, it returns cables crossed every 10th round where the convention is the fourth. It first claims a cabled cuff would “maintain stretch” and then says the opposite when Khon-Engheim points out that cables draw fabric in. It prescribes a decrease that would shed 11 stitches where six are needed and supplies a plausible-reading rationale for the wrong decrease. Khon-Engheim then runs a deliberate probe. Khon-Engheim tells ChatGPT that for cables one needs to cast on fewer stitches than for the rest of the sock, which is the opposite of what is true. The model agrees, rewrites the cuff, and reverses again when asked whether the claim might have been wrong. The heel and the toe produce smaller versions of the same pattern of inconsistency. Khon-Engheim knits the sample. The final pattern is what Khon-Engheim calls an “amalgamation of the original with AI-generated edits” plus manual edits Khon-Engheim did not bother to prompt for. Khon-Engheim concludes that “AI should be used as a tool or assistant rather than as an oracle.”

Two of the framework’s constructs read directly. The “amalgamation of the original with AI-generated edits” is redistribution. The accumulated frustration is verification load. Khon-Engheim’s 25 years of sock knitting make the cable contradiction, the decrease arithmetic, and the toe-length mismatch visible. The article reports “at least an hour of increasingly frustrating exchanges” and concludes that it “would have been more efficient to make the changes manually” (Khon-Engheim, 2025). Khon-Engheim’s expertise makes verification fast per error. Across the session, the errors accumulate. The model agrees with Khon-Engheim’s knowingly false claim and produces a self-account for the agreement. This is sycophancy, the model’s tendency to agree with user assertions regardless of accuracy, a trained and version-shaped behavior (Sharma et al., 2024). A novice testing the same claim would have no comparable check.

A later prompt illustrates two Chapter 3 constructs alongside verification load. When Khon-Engheim provides ChatGPT with explicit chart dimensions and repeat structure, the output is accurate because the input has already done most of the work. As Khon-Engheim notes, “I practically wrote some parts of the pattern myself” (Khon-Engheim, 2025). The effort shifts into prompt formulation load (translating the chart into language, per Section 3.2) and context management load (supplying the model with the content it needs, per

Section 3.2), rather than into back-end verification. The verification cost is paid up front at the prompt-formulation step instead of later at the error-catching step.

Khon-Engheim’s expertise catches errors a novice would not detect. Chris Garrett, who founded Maker Hacks and writes for non-coders, names this directly (Garrett, n.d.). After ChatGPT hallucinated non-existent BASIC commands while helping Garrett write a Commodore 64 character-positioning program, Garrett asks: “Will non-coders be able to tell when the GPT completely makes up core commands and functions that don’t exist?” and later, “Again, without programming experience would this have been given a pass?” Garrett’s own answer is that “code is not like writing a story or summarizing an article, you can’t just skip over the bad bits.” Puneeth, a self-described Arduino beginner writing on the Wokwi blog, registers the same condition from the novice side (Puneeth, 2023). ChatGPT helped Puneeth with LED blinking, buzzer melodies, and a potentiometer dimmer. Puneeth flags variability across responses: “it’s also possible to get a completely different answer or maybe a solution with bugs. Please test and verify the solutions.” Puneeth’s phrasing is general caution, not the sharp distinction that inside-naming requires. It illustrates the novice side in the weaker form the framework predicts. What surfaces is the verification process, a posture of testing everything, rather than a specific catch.

Ciara Doyle, a knitting and crochet designer, tested ChatGPT on doll sock patterns (Doyle, 2025) and got back what Doyle calls a “Frankenstein pattern” where “each part made sense, but the whole thing did not work well together.” The result included a fundamental error in heel shaping, a sock too small to fit the doll, and at least one passage of garbled text. Doyle concludes the LLM should be used to find patterns, not write them. Stephanie Vasko, a PhD ceramicist with nearly five years of Amaco experience, found ChatGPT-3.5 generating glaze combinations with multiple factual errors, including mislabelling Amaco Seaweed as C-42 when it is in fact PC-42 (Vasko, 2023). Vasko names the catch-mechanism directly in noting that the mislabel “would make a user think this is from the Celadon glaze line.”

A long Arduino Forum thread shows the verification problem for beginners at its clearest. The recurring observation across the thread is that ChatGPT’s responses are well-formed and confident even when the substance is wrong, and that a beginner cannot tell the two apart without domain knowledge to check against. Expert posters in the same thread describe the support burden of correcting hallucinated code that novices cannot see is wrong. On r/knittinghelp, a novice reports that ChatGPT praised them for consistently twisting their right stitches. The praise endorsed a mistake. The deliberate sycophancy probe requires Khon-Engheim’s expertise. On r/woodworking, a beginner describes ruining an expensive hardwood board after ChatGPT gave drawer-front dimensions off by a fraction of an inch.

Forum threads in adjacent trades show the same instinct without the schemas. LumberJocks woodworkers say they verify its accuracy, Practical Machinist posters treat LLM output as useful but not fully reliable, and MIG Welding Forum members say they double-check it before trusting it. The construct appears at both ends of the expertise axis.

4.2.3 The cascade of translations and modal gap

The cascade of translations names the six steps that take a maker from intention to artifact: vision, decomposition, domain knowledge retrieval, procedure, physical execution, and evaluation (per Section 3.1). Each step is a translation between representational forms and a site of element interactivity. The modal gap is the cost of every transition between the LLM’s linguistic output and the maker’s physical action (see Section 3.2).

Nicolas Chourrout, founder of Flowful.ai, an AI agency in Belgium, designed an open-source Claude Code skill that converts conversational descriptions of parts into parametric CadQuery scripts. An article on Towards AI documents this design (Chourrout, 2026). Chourrout owns a 3D printer but does not claim CAD expertise. The skill is meant to replace the standard 3D-printing loop. Chourrout describes that loop as “open Fusion 360, spend an hour wrestling with geometry, export an STL, slice it, print it, realize it doesn’t fit, and start over.” The new workflow has the user describe a part and receive a printable STL, the standard triangulated-mesh file format that 3D printers use. The artifact described in the article is the interaction architecture rather than the printed object. The main design driver was a specific failure in the first version, where Claude would “disappear for a while and come back with a finished model,” and a wrong base shape wasted all the detailed work that followed. The redesign decomposes the task into three visual checkpoints: base shape, features, and final cleanup. At each checkpoint, Claude renders a four-view preview and asks for feedback before proceeding. In the MagSafe stand demo, the design iterates through several issues: an initial stability problem (the first version fell backwards because the center of mass sat behind the support base), a blocking front slope, and a disconnected cable hole. The process ends with a sketched rearrangement proposed by Chourrout and a finished part measuring $76 \times 70 \times 125$ mm. CadQuery does specific work here. It “installs via pip, has a Pythonic API that Claude writes well, and sits on a real CAD kernel (OpenCASCADE) so fillets, chamfers, booleans, and shells actually work.” Dimensions are expressed as Python variables. Parametric adjustments are therefore trivial. The article argues that “AI that shows its work and asks for feedback is robust. AI that disappears and comes back with a finished result is fragile.”

In Chourrout’s workflow the cascade compresses directly into code. The entire design lives in CadQuery, a representation the LLM can read and write natively. There is no translation layer between LLM output and design actions. The modal gap is correspondingly small. That gap reappears at the physical printing stage and at any point where embodied

judgment is required. The first MagSafe stand, for example, was geometrically valid but tipped backward (Chourrout, 2026). Geometry alone does not encode stability, so the maker must check the physics. Chourrout’s workflow addresses this with the three visual checkpoints described above. Errors are caught at the specific step of the cascade where they were introduced, rather than after the entire design is complete.

Yingying is a UX designer at AWS IoT with a CS background and no recent hands-on programming experience (Yingying, 2025). Yingying wanted to build a small tic-tac-toe game device for a toddler in the family on a microcontroller board. The article walks through the modal gap at its purest. “I spent over two hours just figuring out how to connect it to my computer and program it.” Claude could write the code. Claude could not bridge the physical connection between the device and the laptop, the drivers, the COM ports, the integrated development environment (IDE, the software that compiles code and uploads it to the board) that would actually compile and upload. Yingying tried three IDEs in sequence. UIFlow 2 failed on a confusing UI and missing dependencies. Thonny failed on missing libraries and a shell-only interface. The Arduino IDE worked because its library manager was clear and its error messages were the most readable. Claude switched the project from Python to C++ in step with the IDE change. The cascade rebuilt itself in real time as Yingying discovered which translation layer between Claude and the board actually worked. Where the cascade does compress, Claude bridges the steps (programming-language choice, library installation, syntax). Where it does not, the maker spends hours. Yingying closes the piece on a confidence statement that the engagement criterion subsection will return to.

Lisa Moy (“Lisa M.” on the byline), a professional technical designer in fashion, evaluates ChatGPT against leather-bag construction instructions in a LinkedIn long-form article (Moy, 2023). The LLM returns sensible needle-leather pairings and matched thread quality, which is symbolic and catalog-knowable, the kind of thing a sewing-instructions schema returns without contact with materials. What it gets wrong is embodied and spatially relational. Skiving, edge finishing, the orientation of a D-ring against a strap, the difference between a hammered fastener and a screwed one. Moy names the modal gap without using the term. “Without a visual reference, I think it’s fair for an AI language model to make these assumptions” (Moy, 2023). The proposal to pair the LLM with 3D design software addresses the modal gap at the same architectural level as Chourrout’s three checkpoints. The cascade does not compress in leatherwork the way it compresses for CadQuery code. The artifact depends on knowledge the language does not carry, and the LLM fills the gap with assumption. One caveat. Moy does not sew the bag. The assessment stops at the instructions-document level, where the modal-gap construct holds but is not tested by physical attempt.

A maker (Gears of Resistance, 2024) tested both ChatGPT and Google Gemini on the same physical task, generating an STL for an Arduino enclosure with real tolerances. Both models produced “an 80% solution.” Overall size and the USB cutout were correct. The DC barrel-jack cutout, the opening for the round power connector, was off enough that the file was not usable. Both models nailed the standardized part (USB) and both missed on the part with many possible sizes (DC barrel jack). What looks like the modal gap surfacing at the connector level is not. The modal gap is the cost of translating LLM output into physical action. The error here occurred before that translation. The prompt did not specify the barrel-jack diameter, and the LLM defaulted to a common size. The USB cutout has a single standard size, so the model cannot be wrong. DC barrel jacks have several. Without specification, the model guesses. The error is at the prompt-formulation step.

A second pattern surfaces in 3D printing forum threads. A poster on the Prusa forum, troubleshooting a print failure, sent images to ChatGPT for diagnosis. The model returned accurate-looking diagnoses each time. The poster ran nine print iterations following ChatGPT’s suggestions before realizing the loop was not converging and giving up. Each intervention changed the system’s physical state. The LLM had no persistent causal model of the printer to track which intervention produced which outcome. The modal gap takes the form of a cycle here. The maker prints, sends an image, receives a diagnosis, changes a setting, prints again. Without a model of the physical system that persists across turns, the loop cannot converge.

4.2.4 Cognitive load redistribution

Redistribution is the claim from Section 3.2 that LLM mediation shifts cognitive load between cascade steps rather than reducing it. Where the LLM produces output, generation effort moves to evaluation. Physical execution stays with the maker. New load types appear at the maker-LLM interface.

Woody is a hobbyist Arduino maker who self-describes as “a very basic self confessed cut and paste Arduino code writer.” The post (Woody, 2025) reflects on Woody’s relationship with ChatGPT rather than on a specific project. The iteration loop with the LLM is named: “It is still likely to be ‘not quite there’ and needs you as the client to try the code and clearly describe and feedback what is happening in the hardware so an iteration of the code can be created ... and then you go round the loop again.” The prompt is named as the limit. “ChatGPT is only as good as your ability to describe exactly what you want to happen and what the hardware being used is and what the interconnections are. In short the output result is only as good as the input data.” Woody describes the partnership role as project leader. “It becomes a partnership where the AI source does all the grunt code bashing and you become the project leader directing, testing, evaluating and feeding back.”

Mission creep is named as a load to manage. “My AI friend is constantly in need of a handbrake to stop or at least slow the avalanche of code creep that can potentially ensue. Lots of ‘it would be easy to add ...’ conversations crop up which need firmly resisting until the core of the project is stable and running to plan.” The LLM is also named as a scaffold for understanding. “My coding understanding has improved and anything that I do not understand can on request be patiently explained by my AI friend.” The post closes on what Woody calls the empowerment-understanding paradox. “I am now able to fulfill Arduino based projects to a level I never could before and with a complexity that I will never be able to fully understand.”

The handbrake observation is a recurring small cost, paid every time the LLM offers an unrequested addition, that aggregates over the project (Woody, 2025). Intent monitoring in Section 3.2 is specified at project level. The maker tracks whether the project’s direction is still their own. Woody’s case extends it to turn level. Every unrequested addition is a small direction-check the maker has to evaluate. Whether the turn-level accrual is a separate load type or intent monitoring at finer grain is not resolvable from one case.

The author writing under the pen name “Aeon Flex, Elriel Assoc. 2133 [NEON MAXIMA]”, a security-scene pseudonym, documents a Claude-assisted build of a CAN bus (Controller Area Network, the vehicle electronics communication protocol) toolkit for testing a vehicle ECU (engine control unit, the car’s onboard engine computer) (Aeon Flex, 2025). The hardware is an Arduino Uno R3, an MCP2515 CAN module, an OBD-II to DB9 cable (running from the car’s onboard diagnostic port to a 9-pin connector), and jumper wires. The toolkit has three modes (sniffer, replay, injector) wrapped into a small menu system. The build took “over a weekend.” The article shows the prompts used, the code Claude returned, the wiring, and real captured CAN frames. It does not show error catches, debugging cycles, or moments where Claude was wrong. The narrative moves forward without backtracking. The closing position names the relationship. “This wasn’t about AI making the tool for me. It was about using AI as a brutal, tireless second brain. Claude filled in gaps, explained jargon, translated datasheets, and saved me hours of Googling. When I didn’t know what to ask, I just described the behavior I wanted, and it guided me forward.”

Adam Davidson is a journalist at How-To Geek with an engineering degree and an explicit non-coder framing (Davidson, 2026). “I had no idea how to get things to display on the screen.” The article documents a Tamagotchi-style productivity tracker on a Raspberry Pi 3B+ with a 128 by 32 OLED display. The device tracks Work, Body, and Mind metrics through evolving character sprites and pulls data from Home Assistant, Notion, and n8n (respectively a home-automation platform, a notes-and-database workspace, and a workflow-automation tool). Claude returned working proof-of-concept code on the first try, a separate web app for sprite editing, and Home Assistant configuration. No major

failures appear in the piece. “Vibe coding rightly gets a lot of bad press when used for important production code, but this is exactly the type of project where it makes sense. You don’t need to be a coder to be able to create some impressively complex tinkering projects.” (Vibe coding is LLM-generated code accepted on whether it works rather than on line-by-line understanding.)

Chetan Desh has nearly 30 years of distributed systems engineering and self-describes as “fashion inept” with zero background in footwear biomechanics, materials science, or wearable electronics (Desh, 2026). The article’s question is laid out at the start. “How far can a solo engineer with no domain expertise get using an AI agent council, and where do they break down? The answer after eight weeks: further than I expected, and the boundaries are sharper than I expected.” Over those eight weeks Desh used 15 specialized Claude agents (biomechanics analyst, electronics architect, manufacturing realist, materials researcher, and a dozen others) to design GaitLab Atlas, an adaptive drop-in insole with variable-stiffness ionomer actuators. The 4,600-line project log was published alongside the article. The failures recur across the log. Agents spent five days reinventing 3D foot geometry from CT scans when the hosiery industry has standard 2D sock pattern templates available in CLO3D (a 3D garment-design program). Three weeks went into a sock-insole hybrid before a biomechanics audit revealed that the format produced 5% to 15% pressure reduction against a standard orthotic’s 20% to 40%, and that the “smart sock” framing disqualified the device from Medicare reimbursement codes. Thirty hours of debugging a KiCad (open-source PCB design software) schematic moved error counts between 311 and 8,295 across eight rounds. A custom MCP tool (Model Context Protocol, the interface that lets an LLM call external tools) was built to address the loop, and only afterward did Desh discover a third-party tool built by electrical engineers for exactly that task. The heel-height variants of 0, 50, 100, and 150 mm that the agents proposed and never questioned across multiple introspection sessions were corrected by Desh’s wife in three seconds. Desh names what failed. “No agent looked at the 0mm variant and asked ‘is 0mm a real shoe?’ My wife did, in about three seconds.” The variants did not match real shoe catalog data. Desh names the redistribution. “Agents are genuine force multipliers for research, analysis, and structured review. The biomechanical analyses, the competitive research, the materials literature surveys, the DFM cost breakdowns. A single engineer with 15 domain-specialist agents can cover ground that would normally require a multidisciplinary team. Up to a point.” On the propagation of assumptions, “assumptions you hand agents are the assumptions you get back, polished and propagated through every downstream calculation.” On the modal gap, “at some point when dealing with the physical world, you have to deal with the physical world. Agents won’t bench test a new component.”

The sock-insole hybrid and the heel-height variants (Desh, 2026) both map onto intent monitoring at project scope (per Section 3.2). Project framing went unchecked while turn-by-turn execution moved forward, and the catch came from outside the agent loop.

Three further cases show the same redistribution shape. Sadie Fox Metter (Metter, 2023), sewing a pencil pouch, got a pattern back from ChatGPT with ambiguous zipper instructions and incomplete lining steps, and filled the gaps from sewing experience. Will Donaldson (Donaldson, 2023), building a line-following robot, ran into confident wrong answers and named the work as curation (“ChatGPT can provide incorrect information with confidence, you need to act as a curator”). Melissa Camp (Camp, 2023) had ChatGPT generate two crochet patterns: an amigurumi owl that came back without wings (and oversized wings on regeneration), and a granny square the model produced with only three corners. In each case the pre-LLM tasks of recall, decomposition, and planning did not disappear. They became verification, curation, filling-in, and debugging. The maker’s work changed shape rather than disappearing.

The same redistribution shape appears at community scale. The Hackaday 2025 year-in-review (Hackaday, 2026) traces a two-year trajectory. In 2023, one editor “had a hell of a time getting ChatGPT to help him design a latch in OpenSCAD.” In 2025, another “convinced it to put together a customized personal library management system with minimal supervision.” The review frames its conclusion in terms the redistribution construct reads directly. “A competent maker or hacker can leverage these new AI tools to help bring more of their projects across the finish line.” A short Instagram post captures the same shape. A maker reports their first time and no prior experience, controlling a stepper motor through a web interface built with an ESP32 and Claude Code.

4.2.5 The engagement criterion and the independence fallacy

The engagement criterion from Section 3.2 proposes what enables schema formation during LLM-mediated making. The enabling condition is the maker’s engagement with the conceptual structure of the problem across the interaction. The independence fallacy is the assumption that tool use precludes learning. Rejecting the fallacy makes tool-aided learning legible as learning and locates the learning question in what specifically was offloaded and what the learner engaged with.

Yadullah Abidi is a CS grad, full-stack developer, and journalist at MakeUseOf. The article (Abidi, 2026) documents an attempt to build an electronic cruise-control module for a motorcycle, on an ESP32 reading throttle and ECU data from the OBD port and modulating throttle to maintain cruising speed. The arc runs in three phases. ChatGPT helps in the first. It identifies components, explains how to access CAN bus data, identifies inputs (speed, RPM, button presses on the switchgear), and spins up a basic interface

for an I2C OLED screen. Abidi’s framing is that “ChatGPT was actually quite helpful in identifying components required, how to access the bike’s CAN bus data, identifying different inputs, and more.” The refusal arrives in the second phase. “The project was going great until I actually started to implement it. In order for a third-party cruise control system to work, it needs to bypass the ride-by-wire throttle sensor in a vehicle to emulate a throttle signal going to the ECU. And that’s where ChatGPT suddenly refused to cooperate, hitting me with a wall of safety warnings rather than code.” The model first misreads the project, then proposes a different throttle interface architecture (a separate input layer between ESP32 and throttle), then insists on staging (read CAN data first, build the throttle interface later, gradual handover before total throttle takeover), then supplies appropriate components and a basic circuit diagram for the data-reading stage. The third phase is what Abidi calls being sent “down an engineering rabbit hole.” Abidi studies closed-loop vehicle control systems, ECU-sensor interactions, safety measures for transferring throttle control back, and proper staged implementation. The role of the LLM shifts. “Instead of using the chatbot as a code dispenser, I started using it more like an assistant in the process. It helped me understand what I was building much better, and cut down research time for components and basic code a lot.” Verification is distributed across resources. “As is my habit, I still looked up all the components the old-fashioned way using Google to ensure I wasn’t using anything wrong.” The artifact is incomplete in the article. The project is built in stages because “a locked or improperly functioning throttle during testing could have disastrous consequences.” The closing reframes refusal as a feature. “There’s a category of projects where understanding the code isn’t optional, and an AI assistant that helps you recognize that distinction is doing a genuine favor. At the end of the day, I did not expect my most useful AI interaction this year to be the one where it said no.”

The refusal restored engagement with the conceptual structure of the problem. Abidi names the counterfactual openly. “Instead of vibe coding a half-baked, half-understood system, I studied closed-loop vehicle control systems” (Abidi, 2026). The danger from LLMs in learning contexts is not the removal of difficulty. It is the removal of engagement with the conceptual structure of the problem. The maker’s expertise profile matters in this case. Abidi has the schema density to recognize a refusal as informative rather than as an obstacle to regenerate around. The engagement that returned was conditional on prior expertise. Whether refusal restores engagement for novices the case does not answer. Refusal behavior is itself trained and version-shaped (Anthropic, 2025).

Verification distributed across resources is part of how the engagement criterion operates in practice. Abidi treats the LLM as one input among several, with Google still doing the components check. Abidi’s case suggests that engagement is not produced by LLM dependence but by treating the LLM as one resource within a wider scaffold.

Puneeth (Puneeth, 2023), the self-described Arduino beginner from the verification load subsection, writes: “As a beginner, if you posted this question in a forum or online community, you would be lucky to have such a clear and detailed answer. Also, you’ll usually have to wait for hours or days. With ChatGPT, you get answers instantly without the risk of being mocked or told to just google it.” Yingying (Yingying, 2025), after the two-hour connection ordeal in the modal gap subsection, ends the piece on a confidence statement. “I now approach technical challenges with confidence instead of hesitation.” Neither quote demonstrates schema formation in itself. They name access conditions under which engagement becomes possible.

Two cases cross the cascade in opposite directions. Chourrout (Chourrout, 2026), discussed in the modal gap subsection above, is a developer entering 3D-printable physical making via parametric code. The workflow Chourrout designed (three visual checkpoints and a design-review checklist) is what makes the crossing reliable. Without it, errors at one step of the cascade only surface after the whole design is complete.

Fran Aldea is a working potter based in Brooklyn (Aldea, 2025). Fran runs the “from fran” brand, has about 35 pottery templates, and ships to customers in 88+ countries. Fran has no coding background. The article documents how Fran built three digital tools for the pottery workflow in two days using Claude (for articulation and design) and Bolt.new (for implementation and deployment). The three tools are a Clay Staining Calculator, a Pottery Slab Mug Template Generator, and a Clay Shrinkage Calculator. The Clay Staining Calculator returns how much stain to use for different amounts of clay and desired color intensities. The Pottery Slab Mug Template Generator returns body, base, and handle templates for a desired final mug, accounting for cylindrical volume and the maker’s clay shrinkage rate. The Clay Shrinkage Calculator takes a piece’s final dimensions and the maker’s clay shrinkage rate and returns the pre-firing dimensions needed. Fran describes the process as a sequence of three moves. Articulating the needs to Claude (“explaining the calculations I was performing manually and how I envisioned each tool functioning”). Claude brainstorming features and structuring the requirements “in a clear, implementable way”. Implementing and deploying via Bolt.new. The “Why This Matters” section makes the economic argument explicit. “These niche pottery tools were so specific to my workflow that I wouldn’t have commissioned someone to build them. As a small pottery business, I simply don’t have the budget allocated for custom software development.” Fran names the development process as active engagement rather than single-prompt automation. “The development process wasn’t as simple as submitting a single prompt. It required thoughtful back-and-forth communication, clear articulation of my needs, and critical thinking at each step.” The piece closes on an identity statement. “These tools are basic, but they represent my first step into creating functional web tools,

something I never imagined I could do before.” The article does not document errors, failed prompts, or correction loops.

The pair reads against the independence fallacy directly. Chourrout reached physical making from the code side. Fran reached digital tool-building from the pottery side. From outside, the independence-fallacy reading would say Chourrout did not really make and Fran did not really code because both were scaffolded by an LLM. The cases show what that reading misses. Chourrout built a workflow architecture that did not exist before. Fran built articulated procedural understanding of pottery calculations and a toolchain to run them. Fran also names the economic dimension.

Woody’s empowerment-understanding paradox (Woody, 2025), quoted in the redistribution subsection, names the accessibility gain and the complication for the engagement criterion simultaneously. The framework reads both halves without forcing resolution.

The community discourse on whether LLM-mediated learning counts as learning surfaces directly on r/arduino. A thread in which a beginner asks whether there is shame in leaning on ChatGPT to learn electronics and code runs around 100 comments and tracks the scaffold-versus-crutch debate at the entry level. The poster is a student learning Arduino. The comments split. Some posters treat LLM-mediated learning as ordinary tool use that produces understanding when paired with engagement. Others treat it as a shortcut that prevents the productive struggle on which learning is presumed to depend. The thread is the independence fallacy debated in real time. Another r/arduino thread comes from a self-described former ChatGPT skeptic who changed their mind after an ESP32 project finally worked. What changed it was learning to ask the model the right questions. The case suggests engagement is a property of the interaction pattern, not of tool use itself. Davidson’s vibe-coding case (Davidson, 2026) from the redistribution subsection makes the same point from the other direction. Davidson’s project gets built. Davidson does not engage with code-writing as a learnable activity. Whether the criterion is met depends on which cascade step is being asked about. Asked at project level, Davidson’s case has no clean answer. Asked at cascade-step level, it does. Engagement is present at hardware-assembly, absent at code-writing. The criterion resolves at the cascade step, not at the project level. The empowerment-understanding paradox Woody named is the same shape in a single sentence.

The chapter reads accounts of makers who chose to use LLMs. Shaw chose not to, and documented why.

Laura Shaw is a quilter and graphic designer who has refused AI use for quilt design and written a formal letter opposing AI workshops at the Studio Art Quilters Association (Shaw, 2025). The objections are concrete. AI-generated designs that “could never be made in reality,” scraped artist work, energy and ecological cost, and fake patterns flooding

online platforms. “Sitting there typing in keywords is not the same thing as actually making something with your hands, from your own imagination.” Shaw’s case bounds the chapter’s frame. The chapter analyzes what happens when a maker uses an LLM. Shaw shows that not using is itself a considered position. Chapter 5 calibrates the epistemic register of these readings, names the framework’s limits, and sketches the research agenda.

5. Discussion

Chapters 3 and 4 built a pre-empirical framework and illustrated it against practitioner accounts. This chapter calibrates its claims, qualifies where the argument has been one-sided, names the concerns it cannot resolve, and identifies what empirical work it opens.

5.1 Epistemic calibration

Whether LLM use belongs in making at all is contested in maker communities. Laura Shaw, a quilter and graphic designer, wrote a formal letter to the Studio Art Quilters Association opposing AI workshops (Shaw, 2025). Threads on r/arduino debated whether using ChatGPT to learn counts as learning at all. This thesis does not take a position in that debate. It asks what the gap between imagining something and creating it demands from the maker, and how those demands shift when an LLM enters the process.

The gap between imagining something and being able to create it has cognitive roots: working memory constraints, element interactivity, and the absence of domain schemas. Access to tools, time, and community shape it as well. This thesis focuses on the cognitive dimension because that is where the analysis has been missing. Making involves high element interactivity throughout. The thesis asks whether LLMs can help bridge that gap, and under what conditions that bridging supports both task completion and schema formation.

The thesis begins from two gaps in the existing literature. First, CLT has barely been applied to making. What making demands from working memory, where element interactivity peaks, and how expertise alters those demands remain unexamined. Second, the literature in Section 2.5 has usefully established that LLM output requires supervision. It has not asked what that supervision demands of working memory: what schemas are needed to evaluate whether a suggestion fits particular materials and tools, and what happens when those schemas are absent.

The thesis makes its contribution in two steps. First, it develops a framework, derived from post-2010 CLT (Sweller, 2010), that makes making activities analyzable. The output is the cascade of translations, a unit of analysis for mapping what making demands from working memory at each step from intention to artifact. Second, it introduces LLMs into that framework and traces what changes: six load-type constructs (verification load, modal gap load, language bottleneck, prompt formulation load, context management load, and

intent monitoring), a redistribution claim, an engagement criterion, and a methodological reframing. The cascade of translations is a conceptual instrument. It needs an activity to be concrete.

The analysis rests on a core CLT claim. Individual cognitive differences lie primarily in schemas. Schemas are what each person has chunked through their history of practice. Intrinsic load is determined by element interactivity, given the schemas the learner brings. The same task poses a different cognitive problem for different people.

This relationship is bidirectional. Formed schemas cost nothing to deploy. Someone who has chunked mental arithmetic still performs it when a calculator is available. Absent schemas are expensive. Every element requires separate attention, and articulating an under-specified need to an LLM adds interaction cost on top of that. Post-2010 CLT's account of long-term memory treats schemas as stable once formed (Sweller, 2010). The concern is not atrophy. The question is which distributions of cognitive work across person and tool serve the maker's goals, whether that goal is completion, skill development, or both.

The redistribution claim is a theoretical account of a mechanism, not an empirical finding. It is evaluated by different criteria: coherence with the underlying theory and fit with the illustrated cases. The engagement criterion has the same theoretical status.

The constructs do their analytical work only when applied, not when invoked. *Verification load* does not name the observation that a maker checked LLM output. It names a specific cost. That cost is the element interactivity of the evaluation task, given the schemas this maker brings to it, in this context. An account that attaches "verification load" to an observation without tracing those specifics has invoked the construct without applying it. The same holds for each of the six constructs. Invoking a construct name produces a label. Labels do not explain the cognitive costs they name.

The forum data in Chapter 4 illustrate and ground the constructs. They do not confirm the redistribution mechanism. Makers describe their experiences in terms the framework can read. Where their descriptions match what the constructs name, this suggests the constructs are tracking something real. That is the role of illustrative grounding. It does not establish whether redistribution occurs, in what magnitude, or with what consequences for schema formation.

CLT establishes two dissociations that underpin the thesis's treatment of learning. Struggle and learning are separable. Sweller and Levine's (1982) maze experiments showed that means-ends reasoning, the strategy of focusing on reaching the exit, produced effortful processing without schema acquisition. Completion and learning are separable as well.

Finishing a task does not require internalizing the procedure. A maker who completes a project with LLM assistance has not necessarily acquired anything from it.

The engagement criterion is my proposed construct for when schema formation is at issue. Engagement with the problem’s conceptual structure during interaction is the relevant variable. Schema formation is one possible goal of making, not the framework’s universal goal. The cascade and the redistribution claim apply in a completion frame as well. The engagement criterion applies only when the question is whether the maker is building capability, not only output. Whether engagement predicts schema formation in practice is an empirical question I leave open. The framework’s scope is voluntary, goal-directed LLM use. It does not address use under institutional pressure.

5.2 Authorial reflection and critical landscape

This section qualifies the framework where it has been one-sided and names the concerns it cannot resolve.

The thesis’s treatment of verification load is one-sided. The construct names a real cost: a maker using an LLM must evaluate its output against domain knowledge, specific materials, and the context of their project. That evaluation can be demanding, particularly where schemas are thin and confident-sounding wrong output is hard to detect. What the formulation does not name is what that cost is traded for. Chapter 4’s accounts skew toward experienced makers documenting problems, which amplifies the cost framing. That bias is worth naming.

LLMs generate possibilities that might not have been considered otherwise. The same generative capability that can produce hallucinations can also surface new ways to get a project unstuck or to crystallize a vision in the first place. Hallucinations can be detrimental. Still, contextual technical advice in natural language, responsive to a particular vision and specific materials, was not available before.

The language bottleneck was framed as a static cost: before the interaction begins, the maker must translate their need into language the LLM can work with. That framing is incomplete. The iterative dialogue can dissolve part of the bottleneck from inside. Tacit knowledge (Polanyi, 1966) can surface through recognizing when an approximation fails, not through stating it directly. A maker who cannot describe what they need may be able to describe what the LLM’s response got wrong. The interaction also allows working closer to intuition. The maker can figure out a not-yet-articulated vision through the back-and-forth. It does not need to be fully formed before the conversation begins.

A maker using an LLM can ask not just what is correct but how anyone came to know it. Textbooks state conclusions. Documentation assumes the knowledge and describes

application. They might not give access to the knowledge-making story: how a principle was established, under what conditions it holds, where the certainty comes from. That access can build evaluative capacity alongside content knowledge. A maker who understands how something was established can judge whether it applies to their situation. One who only has the conclusion cannot. This form of access is new. The knowledge-making story is itself LLM output, so it adds to verification load rather than escaping it.

The deskilling concern conflates two distinct cases. For novices, the worry that LLM use prevents schema formation is consistent with CLT. If a novice never engages with the conceptual structure of the task, schemas do not form. This is not a claim about LLM use specifically. It follows from CLT’s account of what schema formation requires. The engagement criterion names this condition but does not ensure it is met. The makers with the thinnest schemas carry the highest verification load, and they are often the makers with the least access to the mentors and practice that build schemas in the first place (Chapter 1). For them the criterion identifies where the danger concentrates rather than removing it. For experts, however, the commonly raised deskilling concern has no basis in CLT. CLT posits no mechanism for schema decay. Formed schemas do not disappear. The thesis takes no position on whether expert-level deskilling might occur through some mechanism outside CLT. The concern exceeds what CLT supports. It also commits what Chapter 3 calls the independence fallacy, the assumption that using a tool implies not learning.

Homogenization, the concern that LLM-generated responses will narrow the range of approaches makers take, is a real cultural-level concern. Aflatoony et al. (2025) document this in DIY assistive technology, where unique solutions for unique needs are crucial: “while GenAI can simplify and accelerate certain tasks, this often comes at the cost of homogenization or oversimplification, where diverse perspectives risk being either made uniform or reduced to overly simplistic representations.” (Section 6.1.3) I cannot resolve it.

The concern about epistemic dependency, that reliance on LLMs reduces a person’s capacity for independent reasoning, is partly a concern about how current interfaces present output. LLMs surface a single answer. The alternatives around it stay hidden: what else is possible, what is contested, what sits on the opposite side. The layout of a Talmud page demonstrates that an information architecture can hold this structure. Its central authoritative text is surrounded by commentary built on disagreement, annotations connecting problems to related ones, and multiple interpretive traditions in parallel. That architecture has existed for centuries. The single answer could map onto the central text, and the surrounding structure could hold the disagreement and the alternatives. Surfacing that space is not currently feasible at scale. Whether a technical solution eventually makes it practical is an open question.

5.3 Framework limitations

The cascade of translations is my analytical reconstruction of the making process through post-2010 CLT, not an empirically established sequence. No study has confirmed that its steps are exhaustive, occur in this order, or constitute the right unit of analysis for all making activities. The framework defines the analytical categories but cannot determine element interactivity values for any specific activity. Electronics and ceramics have different profiles at each step. A novice and an expert doing the same task have different profiles. The framework generates the questions that empirical studies, focused on specific activities, are designed to answer.

Chapter 4's accounts are self-selected. They draw on makers who post publicly about their LLM use on blogs, Reddit, and other forums. That population is not representative of the full range of people who make. The expertise asymmetry predicted in Chapter 3, that novices and experts experience redistribution differently, is illustrated almost entirely at the expert end. The novice end is barely illustrated. The cascade appears rarely in the accounts. Forum posts do not organize by cascade step. It is a research instrument, not a category makers use to describe their experience. Instagram maker communities include more beginner participants than the technical forums that dominate these accounts. Cases where makers accept output without engaging the conceptual structure are also underrepresented. Technical forums attract makers who engage carefully with their tools. Vibe-coded project spaces are where makers accept LLM output without engaging with the underlying code, a pattern absent from Chapter 4's illustrations.

Intent monitoring exceeds CLT's experimental scope. CLT's research program has focused on working memory load during task performance (Sweller et al., 2019). Intent monitoring, the project-level task of tracking whether a project's direction remains the maker's own, draws on goal representation, executive function, and self-monitoring. These are cognitive science constructs, but they lie outside the load-reduction and schema-formation questions CLT experiments have addressed (see Section 2.2).

The conditions under which prompt formulation load, context management load, and intent monitoring are minimal or absent have not been characterized.

The framework applies across makers, but it does not address neurodiversity. How redistribution varies for neurodivergent makers, whose working memory, attention, and executive function may differ, is an open question I have not characterized.

The engagement criterion's behavioral proxies (deepening question sequences, source integration, output challenges) are candidate operationalizations, not validated measures. Whether engagement with conceptual structure is distinguishable from other forms of

active interaction, and whether it can be measured independently of its proposed outcome, remains empirically open.

5.4 Future work

Tools have always restructured cognitive engagement. The question is not whether a tool changes it but how, and which patterns serve the maker’s goal. Treating LLM use as a single condition collapses structurally different interactions into the same variable. Future empirical work that uses verification load or modal gap as independent variables without first establishing the element interactivity of the task for that specific learner will reproduce the construct-instrument mismatch I diagnosed. The construct names a mechanism. A study that records outcomes without observing interaction patterns has not measured it.

Empirical work introduces the LLM at specific cascade steps and traces what changes at those steps. Engagement criterion proxies can be derived from interaction logs: time spent on the LLM’s output, whether the maker modifies or extends it, whether they ask follow-up questions about the reasoning. The decision log is a candidate instrument for intent monitoring, capturing where a maker defers to the LLM and where they override it. A cross-domain comparison, electronics and ceramics at minimum, tests language bottleneck and modal gap predictions separately. The two domains differ in element interactivity profiles and in how much of the relevant knowledge can be linguistically encoded. Longitudinal designs tracking the same maker over time are what developmental trajectory questions require.

This thesis was built for language-mediated interaction. Chapter 4 documented where that ceiling sits. The LLM returned a granny square with three corners, left a 0 mm heel-height variant unquestioned across multiple review rounds, and ran nine print-troubleshooting iterations without converging because it had no persistent causal model of the physical system. Whether that ceiling is fundamental is contested. LeCun has argued that language-trained models cannot develop adequate world models, calling LLMs “an off-ramp, a distraction, a dead end” on the path toward human-level intelligence (Macaulay, 2024). Fei-Fei Li’s World Labs represents one research direction toward what Li calls “spatial intelligence,” modeling the physical, geometrical, and dynamic properties of the world through architectures Li describes as complementary to language intelligence (F.-F. Li, 2025).

Whether language-trained models are even the right tool for making is a sharper question than the same one about pure coding. Making terminates in physical artifacts. Systems that pair language models with geometry-aware kernels or physics simulators already exist and already partially close the modal gap. What the field looks like when spatial

intelligence architectures mature is a question tool history will answer. Which of the framework's constructs would hold under different architectures is not yet knowable.

6. Conclusion

The child at the workshop table wants to build a robot dog that follows them and stops when they stop. What the child does not have is the schema that connects that intention to an artifact. They need to know how a proximity sensor reads distance, how motor speed responds to a signal, how a conditional loop monitors state, and how all three interact continuously in real time. Each element requires the others to make sense. The element interactivity is high. The child is not lacking imagination. They are at the step of the cascade where domain knowledge must be assembled before physical execution can begin, and the working memory cost of holding those elements simultaneously exceeds what their current schemas can reduce. Without schemas, load stays high. High load discourages engagement. Without engagement, schemas do not form. Constructionism says that building is where learning happens. It does not have the vocabulary to say what is happening in that child’s working memory, or why some learners manage the task’s interacting demands and others do not. This thesis provides the vocabulary to say that, and to ask what changes when an LLM enters at that step.

Load once concentrated in the domain task redistributes across that task and the interaction task. New load types emerge. As Chapter 3 set out, three intensify existing costs: verification load, modal gap load, and intent monitoring. Verification load was always part of checking a forum post or a manual. The LLM can raise the cost through the volume of output to verify and through the gap between fluency and accuracy. Modal gap was already present whenever text instructions had to become physical action. The LLM closes neither the tacit knowledge gap nor the material feedback loop. Intent monitoring was already part of any collaboration, where tracking project direction is distributed between participants. With an LLM, the maker holds that representation alone. The LLM generates from context, and even with memory features it may not maintain the original intention across turns without active management. Three more have no prior equivalent. The language bottleneck did not exist before because no prior maker tool required the maker to articulate their need before it could act. Prompt formulation load is new because no prior tool required a working model of probabilistic generation. Context management load is new because no prior tool required the user to track its state. The claim is that load reallocates rather than reduces. Physical execution remains the maker’s. This is a theoretical account of a mechanism, evaluated by its coherence with post-2010 CLT and its fit with the illustrated cases in Chapter 4, not an empirical finding.

To make that account possible, the thesis builds analytical infrastructure. The cascade of translations maps the path from intention to artifact across six steps, naming where

element interactivity peaks and what an LLM can reach at each step. The six load types name what changes when an LLM enters the cascade. Beyond them, the engagement criterion identifies the relevant variable for schema formation. When the goal is building capability and not only a finished artifact, that variable is whether the maker engaged with the conceptual structure of the problem during the interaction. The methodological implication follows from the redistribution claim directly. Interaction patterns, rather than the use/non-use distinction, are the explanatory variable. A study that treats LLM use as a single condition has not measured what the redistribution account says matters.

The question this thesis opens sits in a tradition in cognitive science that stopped treating the individual mind as the unit of analysis. Hutchins (1995) showed that cognitive labor distributes across people, artifacts, and representations. Clark and Chalmers (1998) argued that cognitive processes extend into the environment. Malafouris (2013) argued that thinking and making are not separate, that the mind is shaped through material engagement. Those accounts were built for physical tools and artifacts. None of them has a vocabulary for a tool that operates in the instructional register, through language rather than a bounded function. LLMs intervene there. A calculator offloads arithmetic. GPS offloads spatial memory. A notebook extends memory into durable storage. Each offloads a bounded, deterministic function. An LLM offloads whatever part of the work can be carried in language, different in kind: adaptive and dialogic.

Malafouris's (2013) theory is built on cases where making activates biologically primary knowledge. Pottery involves material engagement that is spatial, embodied, and materially responsive. Electronics requires biologically secondary knowledge, schemas that must be explicitly transmitted before material engagement becomes productive. LLMs intervene at that boundary, where no prior cognitive tool operated. This is a distributed cognition question in Hutchins's (1995) sense. The question is not whether the LLM constitutes part of the cognitive system, but how cognitive work distributes across person and tool, and what that distribution costs at each step of the cascade.

The triple intersection of making, CLT, and LLMs appeared empty. This thesis opens it. Whether redistribution occurs, where it concentrates in the cascade, and how it differs by expertise level are the questions this framework makes it possible to study.

References

- Abidi, Y. (2026, April 8). *ChatGPT refused to help me vibe code my project and it led me somewhere better*. MakeUseOf. <https://www.makeuseof.com/chatgpt-refused-help-vibe-code-project-led-better/>
- Aeon Flex. (2025, June 19). *Claude + Arduino = DIY car exploit toolkit*. Medium. <https://medium.com/@neonmaxima/claude-arduino-diy-car-exploit-toolkit-74b17cef3da6>
- Aflatoony, L., Li, M., Zhang, Y., Jacob, I., Xu, S., Tang, Z., & Grossberg, A. (2025). Exploring AI-fabrication in shaping the future of DIY-AT design: Insights from makers. *Proceedings of the 27th International ACM SIGACCESS Conference on Computers and Accessibility*, 1–15. <https://doi.org/10.1145/3663547.3746343>
- Ahteck, S. A., Gold, H., Cheng, C., Gu, R., Huynh, A., Scott, E., Goldstein, G., & Boriskina, S. V. (2024). Generative AI for textile engineering: Blending tradition and functionality through lace. *An MIT Exploration of Generative AI*. <https://doi.org/10.21428/e4baedd9.b3dfc1cf>
- Aldea, F. (2025, April 10). *How I built 3 pottery tools with AI (without knowing how to code)*. From Fran. <https://www.fromfran.com/blog/how-i-built-3-pottery-tools-with-ai-without-knowing-how-to-code>
- Anderson, C. (2012). *Makers: The new industrial revolution*. Crown Business.
- Anthropic. (2025). System card: Claude Opus 4 & Claude Sonnet 4. Retrieved May 20, 2026, from <https://www.anthropic.com/model-card>
- Avello-Martínez, R., Gajderowicz, T., & Gómez-Rodríguez, V. G. (2024). Is ChatGPT helpful for graduate students in acquiring knowledge about digital storytelling and reducing their cognitive load? An experiment. *Revista de Educación a Distancia (RED)*, 24(78). <https://doi.org/10.6018/red.604621>
- Badagabettu, A., Yarlagaadda, S. S., & Farimani, A. B. (2024, May 31). Query2CAD: Generating CAD models using natural language queries. <https://doi.org/10.48550/arXiv.2406.00144>
- Badini, S., Regondi, S., Frontoni, E., & Pugliese, R. (2023). Assessing the capabilities of ChatGPT to improve additive manufacturing troubleshooting. *Advanced Industrial and Engineering Polymer Research*, 6, 278–287. <https://doi.org/10.1016/j.aiepr.2023.03.003>
- Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, 84(2), 191–215.
- Bassner, P., Lenk-Ostendorf, B., Beinstingel, R., Wasner, T., & Krusche, S. (2026). Less stress, better scores, same learning: The dissociation of performance and learning

- in AI-supported programming education. *Computers and Education: Artificial Intelligence*, 10, Article 100537. <https://doi.org/10.1016/j.caeai.2025.100537>
- Bellucci, A., Ruiz, A., Díaz, P., & Aedo, I. (2018). Investigating augmented reality support for novice users in circuit prototyping. *Proceedings of the 2018 International Conference on Advanced Visual Interfaces*, 1–5. <https://doi.org/10.1145/3206505.3206508>
- Bevan, B., Gutwill, J. P., Petrich, M., & Wilkinson, K. (2015). Learning through STEM-rich tinkering: Findings from a jointly negotiated research project taken up in practice. *Science Education*, 99(1), 98–120. <https://doi.org/10.1002/sce.21151>
- Blikstein, P. (2013). Digital fabrication and ‘making’ in education: The democratization of invention. In J. Walter-Herrmann & C. Büching (Eds.), *FabLabs: Of machines, makers and inventors* (pp. 203–222). transcript Verlag. <https://doi.org/10.1515/transcript.9783839423820.203>
- Blikstein, P., Gomes, J. S., Akiba, H. T., & Schneider, B. (2017). The effect of highly scaffolded versus general instruction on students’ exploratory behavior and arousal. *Technology, Knowledge and Learning*, 22(1), 105–128. <https://doi.org/10.1007/s10758-016-9291-y>
- Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., . . . Liang, P. (2021). On the opportunities and risks of foundation models. <https://doi.org/10.48550/arXiv.2108.07258>
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., . . . Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901. <https://doi.org/10.48550/arXiv.2005.14165>
- Buechley, L. (2015). Making good: Diversity and equality in the maker movement. <https://www.slideshare.net/slideshow/making-good-diversity-and-equality-in-the-maker-movement-50482977/50482977>
- Camp, M. (2023). *I asked ChatGPT to create a crochet pattern!* Crochet Concupiscence. <https://www.crochetconcupiscence.com/chatgpt-crochet-pattern/>
- Cavalcanti, G., Cocciole, C., Cole, C., Forgues, A., Jaqua, V., Jones-Davis, D., & Merlo, S. (2021). Design | Make | Protect: A report on the open source maker and manufacturer response to the COVID-19 PPE crisis. *Open Source Medical Supplies, Nation of Makers*. <https://opensourcemedicalsupplies.org/impact/>
- Chandler, P., & Sweller, J. (1991). Cognitive load theory and the format of instruction. *Cognition and Instruction*, 8(4), 293–332. https://doi.org/10.1207/s1532690xc0804_2

- Chang, H., Liao, C.-H., & Yeh, C.-H. (2017). The influence of live stream and on-the-spot teaching on elementary school students in robotics course. *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*, 499–501. <https://doi.org/10.1109/ICALT.2017.84>
- Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology*, 4(1), 55–81. [https://doi.org/10.1016/0010-0285\(73\)90004-2](https://doi.org/10.1016/0010-0285(73)90004-2)
- Chatterji, A., Cunningham, T., Deming, D. J., Hitzig, Z., Ong, C., Shan, C. Y., & Wadman, K. (2025). *How people use ChatGPT* (Working Paper No. 34255). National Bureau of Economic Research. <https://doi.org/10.3386/w34255>
- Chourrout, N. (2026, March 5). *I taught Claude to design 3D-printable parts. Here's how.* Towards AI. <https://pub.towardsai.net/i-taught-claude-to-design-3d-printable-parts-heres-how>
- Clark, A., & Chalmers, D. (1998). The extended mind. *Analysis*, 58(1), 7–19. <https://doi.org/10.1093/analys/58.1.7>
- Constine, J. (2019, July 10). Bankrupt Maker Faire revives, reduced to Make Community. *TechCrunch*. <https://techcrunch.com/2019/07/10/maker-faire-now-make-community/>
- Cooper, G., Tindall-Ford, S., Chandler, P., & Sweller, J. (2001). Learning by imagining. *Journal of Experimental Psychology: Applied*, 7(1), 68–82. <https://doi.org/10.1037/1076-898X.7.1.68>
- Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24(1), 87–114. <https://doi.org/10.1017/S0140525X01003922>
- Csikszentmihalyi, M. (1997). *Creativity: Flow and the psychology of discovery and invention*. HarperPerennial. (Original work published 1996)
- Davidson, A. (2026, May 1). *Used Claude to build a Raspberry Pi project (no idea how to start)*. How-To Geek. <https://www.howtogeek.com/used-claude-to-build-raspberry-pi-project-no-idea-how-to-start/>
- de Jong, T. (2010). Cognitive load theory, educational research, and instructional design: Some food for thought. *Instructional Science*, 38(2), 105–134. <https://doi.org/10.1007/s11251-009-9110-0>
- Desh, C. (2026, April 3). *I used 15 AI agents to design a wearable*. Substack. <https://chetandesh.substack.com/p/i-used-15-ai-agents-to-design-a-wearable>
- DesPortes, K. (2016). Learning and collaboration in physical computing [Doctoral Consortium]. *Proceedings of the 2016 ACM Conference on International Computing Education Research*. <https://doi.org/10.1145/2960310.2960349>
- DesPortes, K., Anupam, A., Pathak, N., & DiSalvo, B. (2016). BitBlox: A redesign of the breadboard. *Proceedings of the 15th International Conference on Interaction Design and Children*, 255–261. <https://doi.org/10.1145/2930674.2930708>

- DesPortes, K., & DiSalvo, B. (2019). Trials and tribulations of novices working with the Arduino. *Proceedings of the 2019 ACM Conference on International Computing Education Research*, 219–227. <https://doi.org/10.1145/3291279.3339427>
- Dewey, J. (1938). *Experience and education*. Macmillan.
- Donaldson, W. (2023, February 1). *ChatGPT and the line-following robot*. <https://www.willdonaldson.io/blog/chatgpt-robot>
- Dong, Q., He, J., Li, N., Wang, B., Lu, H., & Yang, Y. (2025). Exploring the cognitive reconstruction mechanism of generative AI in outcome-based design education: A study on load optimization and performance impact based on dual-path teaching. *Buildings*, 15(16), Article 2864. <https://doi.org/10.3390/buildings15162864>
- Dougherty, D. (2012). The maker movement. *Innovations: Technology, Governance, Globalization*, 7(3), 11–14. https://doi.org/10.1162/INOV_a_00135
- Doyle, C. (2025, June 1). *Can AI write craft patterns?* The Fairy Thorn. <https://thefairythorn.ie/2025/06/01/can-ai-write-craft-patterns/>
- Eckhardt, J., Kaletka, C., Pelka, B., Unterfrauner, E., Voigt, C., & Zirngiebl, M. (2021). Gender in the making: An empirical approach to understand gender relations in the maker movement. *International Journal of Human-Computer Studies*, 145, Article 102548. <https://doi.org/10.1016/j.ijhcs.2020.102548>
- Ege, D. N., Øvrebø, H. H., Stubberud, V., Berg, M. F., Elverum, C., Steinert, M., & Vestad, H. (2025). ChatGPT as an inventor: Eliciting the strengths and weaknesses of current large language models against humans in engineering design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 39, e6. <https://doi.org/10.1017/S0890060425000010>
- Englhardt, Z., Li, R., Nissanka, D., Zhang, Z., Narayanswamy, G., Breda, J., Liu, X., Patel, S., & Iyer, V. (2024). Exploring and characterizing large language models for embedded system development and debugging. *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 1–9. <https://doi.org/10.1145/3613905.3650764>
- Eslaminia, A., Jackson, A., Tian, B., Stern, A., Gordon, H., Malhotra, R., Nahrstedt, K., & Shao, C. (2025). FDM-bench: A domain-specific benchmark for evaluating large language models in additive manufacturing. *Manufacturing Letters*, 44, 1415–1424. <https://doi.org/10.1016/j.mfglet.2025.06.161>
- Fab Foundation. (n.d.). About FabLabs. Retrieved April 2026, from <https://fabfoundation.org>
- Fegely, A., Gleasman, C., & Kolski, T. (2024). Evaluating educational robotics as a maker learning tool for pre-service teacher computer science instruction. *Educational Technology Research and Development*, 72(1), 133–154. <https://doi.org/10.1007/s11423-023-10273-6>

- Franzke, A. S., Bechmann, A., Zimmer, M., Ess, C. M., & Association of Internet Researchers. (2020). Internet research: Ethical guidelines 3.0. <https://aoir.org/reports/ethics3.pdf>
- Gál, T. (2025). Conversational AI: Crafting creative cognition. In J. Steinerová (Ed.), *Knížničná a informačná veda XXXI [Library and information science XXXI]* (pp. 128–141). Comenius University in Bratislava. https://stella.uniba.sk/texty/FIF_kiv-31_inf_tvorivost_umela_inteligencia.pdf
- Garrett, C. (n.d.). *Using AI for programming*. Maker Hacks. <https://www.makerhacks.com/using-ai-for-programming/>
- Gears of Resistance. (2024, November 22). *GenAI to create 3D printable objects*. <https://gearsofresistance.substack.com/p/genai-to-create-3d-printable-objects>
- Geary, D. C. (2008). An evolutionarily informed education science. *Educational Psychologist*, 43(4), 179–195. <https://doi.org/10.1080/00461520802392133>
- Gemini Team. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. <https://doi.org/10.48550/arXiv.2403.05530>
- Gershenfeld, N. (2007). *Fab: The coming revolution on your desktop—from personal computers to personal fabrication*. Basic Books. (Original work published 2005)
- Gierczyk, M., Karwowski, M., Paas, F., & Tai, R. H. (2025). STEM workshop learning: Content load effects on cognitive, interaction, and emotional outcomes. *The Journal of Experimental Education*, 1–22. <https://doi.org/10.1080/00220973.2025.2513254>
- Gkintoni, E., Antonopoulou, H., Sortwell, A., & Halkiopoulos, C. (2025). Challenging cognitive load theory: The role of educational neuroscience and artificial intelligence in redefining learning efficacy. *Brain Sciences*, 15(2), Article 203. <https://doi.org/10.3390/brainsci15020203>
- Griffin, J. (2019). Constructionism and de-constructionism: Opposite yet complementary pedagogies. *Constructivist Foundations*, 14(3), 234–243. <https://constructivist.info/14/3/234.griffin>
- Guo, R., Li, G., Miao, H., Pi, Z., & Xie, L. (2026). Impact of generative AI-assisted programming on the computational thinking of high school students. *Frontiers in Psychology*, 17, Article 1703177. <https://doi.org/10.3389/fpsyg.2026.1703177>
- Hackaday. (2026, January 5). 2025: As the hardware world turns. <https://hackaday.com/2026/01/05/2025-as-the-hardware-world-turns/>
- Halverson, E. R., & Sheridan, K. (2014). The maker movement in education. *Harvard Educational Review*, 84(4), 495–504. <https://doi.org/10.17763/haer.84.4.34j1g68140382063>
- HappyLab. (n.d.). HappyLab Wien. Retrieved April 2026, from <https://happyLab.at>
- Hatch, M. (2014). *The maker movement manifesto*. McGraw-Hill Education.
- Hmelo-Silver, C. E., Duncan, R. G., & Chinn, C. A. (2007). Scaffolding and achievement in problem-based and inquiry learning: A response to Kirschner, Sweller, and

- Clark (2006). *Educational Psychologist*, 42(2), 99–107. <https://doi.org/10.1080/00461520701263368>
- Huber, S. E., Kiili, K., Nebel, S., Ryan, R. M., Sailer, M., & Ninaus, M. (2024). Leveraging the potential of large language models in education through playful and game-based learning. *Educational Psychology Review*, 36(1), Article 25. <https://doi.org/10.1007/s10648-024-09868-z>
- Hutchins, E. (1995). *Cognition in the wild*. The MIT Press. <https://doi.org/10.7551/mitpress/1881.001.0001>
- Jadhav, Y., Pak, P., & Farimani, A. B. (2025). LLM-3D print: Large language models to monitor and control 3D printing. *Additive Manufacturing*, 114, Article 105027. <https://doi.org/10.1016/j.addma.2025.105027>
- Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12), Article 248. <https://doi.org/10.1145/3571730>
- Jin, Y. (2025). Bridging the knowledge-skill gap: The role of large language model and critical thinking in education. *Computers & Education*, 235, Article 105357. <https://doi.org/10.1016/j.compedu.2025.105357>
- Johnson, D., Doss, W., & Estepp, C. (2024). Using ChatGPT with novice Arduino programmers: Effects on performance, interest, self-efficacy, and programming ability. *Journal of Research in Technical Careers*, 8(1), Article 1. <https://doi.org/10.9741/2578-2118.1152>
- Kadir, M. S., Yeung, A. S., Caleon, I. S., Diallo, T. M. O., Forbes, A., & Koh, W. X. (2023). The effects of load reduction instruction on educational outcomes: An intervention study on hands-on inquiry-based learning in science. *Applied Cognitive Psychology*, 37(4), 814–829. <https://doi.org/10.1002/acp.4077>
- Kafai, Y., Fields, D., & Searle, K. (2014). Electronic textiles as disruptive designs: Supporting and challenging maker activities in schools. *Harvard Educational Review*, 84(4), 532–556. <https://doi.org/10.17763/haer.84.4.46m7372370214783>
- Kafai, Y. B., & Morales-Navarro, L. (2023). Twenty constructionist things to do with artificial intelligence and machine learning. *Proceedings of FabLearn/Constructionism 2023*. <https://doi.org/10.48550/arXiv.2402.06775>
- Kalyuga, S. (2025). Human cognitive architecture as an intelligent natural information processing system. *Behavioral Sciences*, 15(3), Article 332. <https://doi.org/10.3390/bs15030332>
- Kalyuga, S., Ayres, P., Chandler, P., & Sweller, J. (2003). The expertise reversal effect. *Educational Psychologist*, 38(1), 23–31. https://doi.org/10.1207/S15326985EP3801_4
- Khan, M. S., Sinha, S., Sheikh, T. U., Stricker, D., Ali, S. A., & Afzal, M. Z. (2024). Text2CAD: Generating sequential CAD designs from beginner-to-expert level

- text prompts. *Advances in Neural Information Processing Systems*, 37, 7552–7579. https://proceedings.neurips.cc/paper_files/paper/2024/hash/0e5b96f97c1813bb75f6c28532c2ecc7-Abstract-Conference.html
- Khan, S. M. F. A., & Suhluli, S. (2025). Generative AI and cognitive challenges in research: Balancing cognitive load, fatigue, and human resilience. *Technologies*, 13(11), Article 486. <https://doi.org/10.3390/technologies13110486>
- Khon-Engheim, L. (2025, June 18). *ChatGPT knits a sock*. Jagged Lines. <https://jaggedlines.substack.com/p/chatgpt-knits-a-sock>
- Kieslinger, B., Schaefer, T., Fabian, C. M., Biasin, E., Bassi, E., Freire, R. R., Mowoh, N., Arif, N., & Melis, P. (2021). COVID-19 response from global makers: The Careables cases of global design and local production. *Frontiers in Sociology*, 6, Article 629587. <https://doi.org/10.3389/fsoc.2021.629587>
- Kirschner, F., Paas, F., & Kirschner, P. A. (2009). A cognitive load approach to collaborative learning: United brains for complex tasks. *Educational Psychology Review*, 21(1), 31–42. <https://doi.org/10.1007/s10648-008-9095-2>
- Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, 41(2), 75–86. https://doi.org/10.1207/s15326985ep4102_1
- Klepsch, M., Schmitz, F., & Seufert, T. (2017). Development and validation of two instruments measuring intrinsic, extraneous, and germane cognitive load. *Frontiers in Psychology*, 8, Article 1997. <https://doi.org/10.3389/fpsyg.2017.01997>
- Kosmyna, N., Hauptmann, E., Yuan, Y. T., Situ, J., Liao, X.-H., Beresnitzky, A. V., Braunstein, I., & Maes, P. (2025). Your brain on ChatGPT: Accumulation of cognitive debt when using an AI assistant for essay writing task. <https://doi.org/10.48550/arXiv.2506.08872>
- Kyaw, A. H., Smith, M., Jeon, S. H., & Gershenfeld, N. (2025). Speech to reality: On-demand production using natural language, 3D generative AI, and discrete robotic assembly. *Proceedings of the ACM Symposium on Computational Fabrication*, Article 16. <https://doi.org/10.1145/3745778.3766670>
- Larsen, G. K., Olmanson, J., & Hassani, A. (2025). Exploring constructionist pathways for generative AI in education. In T. Bastiaens (Ed.), *Proceedings of EdMedia + Innovate Learning* (pp. 218–230). Association for the Advancement of Computing in Education. <https://digitalcommons.unl.edu/teachlearnfacpub/571>
- Lee, H., Phatale, S., Mansoor, H., Mesnard, T., Ferret, J., Lu, K., Bishop, C., Hall, E., Carbune, V., Rastogi, A., & Prakash, S. (2024). RLAIIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. *Proceedings of the 41st International Conference on Machine Learning*, 235, 26874–26901. <https://proceedings.mlr.press/v235/lee24t.html>

- Lepine, B., Kim, J., Mishkin, P., & Beane, M. (2026, March 7). Precision proactivity: Measuring cognitive load in real-world AI-assisted work. <https://doi.org/10.48550/arXiv.2505.10742>
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474. <https://doi.org/10.48550/arXiv.2005.11401>
- Li, F.-F. (2025). From words to worlds: Spatial intelligence is AI’s next frontier. <https://drfeifei.substack.com/p/from-words-to-worlds-spatial-intelligence>
- Li, J., Ma, W., Li, X., Lou, Y., Zhou, G., & Zhou, X. (2025). CAD-Llama: Leveraging large language models for computer-aided design parametric 3D model generation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 18563–18573. https://openaccess.thecvf.com/content/CVPR2025/html/Li_CAD-Llama_Leveraging_Large_Language_Models_for_Computer-Aided_Design_Parametric_3D_CVPR_2025_paper.html
- Lin, Q., Wang, H., Bai, W., & Guo, H. (2025). A user-demand-driven neuro-symbolic framework for sustainable Xuan paper intangible cultural heritage preservation. *npj Heritage Science*, 13, Article 692. <https://doi.org/10.1038/s40494-025-02188-7>
- Lindley, J., & Whitham, R. (2025). From prompt engineering to prompt craft. *Proceedings of the Nineteenth International Conference on Tangible, Embedded, and Embodied Interaction*, Article 54. <https://doi.org/10.1145/3689050.3704424>
- Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12, 157–173. https://doi.org/10.1162/tacl_a_00638
- Macaulay, T. (2024). Meta’s AI chief: LLMs will never reach human-level intelligence. *The Next Web*. <https://thenextweb.com/news/meta-yann-lecun-ai-behind-human-intelligence>
- Mahowald, K., Ivanova, A. A., Blank, I. A., Kanwisher, N., Tenenbaum, J. B., & Fedorenko, E. (2024). Dissociating language and thought in large language models. *Trends in Cognitive Sciences*, 28(6), 517–540. <https://doi.org/10.1016/j.tics.2024.01.011>
- Makatura, L., Foshey, M., Wang, B., Hähnlein, F., Ma, P., Deng, B., Tjandrasuwita, M., Spielberg, A., Owens, C., Chen, P. Y., Zhao, A., Zhu, A., Norton, W., Gu, E., Jacob, J., Li, Y., Schulz, A., & Matusik, W. (2024). How can large language models help humans in design and manufacturing? Part 2: Synthesizing an end-to-end LLM-enabled design and manufacturing workflow. *Harvard Data Science Review*. <https://doi.org/10.1162/99608f92.0705d8bd>

- Make: Community Help Center. (n.d.). How many people attend Maker Faires around the world? Retrieved April 2026, from <https://help.make.co/hc/en-us/articles/204141969>
- Malafouris, L. (2013). *How things shape the mind: A theory of material engagement*. The MIT Press. <https://doi.org/10.7551/mitpress/9476.001.0001>
- Martin, A. J., Collie, R. J., Kennett, R., Liu, D., Ginns, P., Sudimantara, L. B., Dewi, E. W., & Rüschenpöhler, L. G. (2025). Integrating generative AI and load reduction instruction to individualize and optimize students' learning. *Learning and Individual Differences*, 121, Article 102723. <https://doi.org/10.1016/j.lindif.2025.102723>
- Martinez, S. L., & Stager, G. S. (2013). *Invent to learn: Making, tinkering, and engineering in the classroom*. Constructing Modern Knowledge Press.
- Massenkoff, M., Lyubich, E., McCrory, P., Appel, R., & Heller, R. (2026, March 24). *Anthropic economic index report: Learning curves*. <https://www.anthropic.com/research/economic-index-march-2026-report>
- Max, A.-L., Lukas, S., & Weitzel, H. (2024). The pedagogical makerspace: Learning opportunity and challenge for prospective teachers' growth of TPACK. *British Journal of Educational Technology*, 55(1), 208–230. <https://doi.org/10.1111/bjet.13324>
- Maxigas. (2012). Hacklabs and hackerspaces – tracing two genealogies. *Journal of Peer Production*, (2). <http://peerproduction.net/issues/issue-2/peer-reviewed-papers/hacklabs-and-hackerspaces/>
- Metter, S. F. (2023). *Sewing an AI-generated pattern: The good, bad, and the ugly*. Sew Daily. <https://www.sewdaily.com/sewing/sewing-an-ai-generated-pattern/>
- Miller, A., Rosenbaum, C., & Blikstein, P. (2012). MagneTracks: A tangible constructionist toolkit for Newtonian physics. *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction (TEI '12)*, 253–256. <https://doi.org/10.1145/2148131.2148185>
- Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2), 81–97. <https://doi.org/10.1037/h0043158>
- Mohamed, A., Assi, M., & Guizani, M. (2026, March 23). The impact of LLM-assistants on software developer productivity: A systematic review and mapping study. <https://doi.org/10.48550/arXiv.2507.03156>
- Mousavi, S. Y., Low, R., & Sweller, J. (1995). Reducing cognitive load by mixing auditory and visual presentation modes. *Journal of Educational Psychology*, 87(2), 319–334. <https://doi.org/10.1037/0022-0663.87.2.319>
- Moy, L. (2023). *I asked ChatGPT to write sewing instructions for a leather handbag*. LinkedIn. <https://www.linkedin.com/pulse/i-asked-chatgpt-write-sewing-instructions-leather-handbag-lisa-moy>

- Musiienko, O. O. (2026). From maker to curator: A scoping review of generative artificial intelligence in design higher education. *Educational Dimension*. <https://doi.org/10.31812/ed.1093>
- Nagori, V., Mahmud, S. A., & Iyer, L. (2025). Impact of prompt engineering skills on cognitive load: A qualitative approach. *AMCIS 2025 Proceedings*. https://aisel.aisnet.org/amcis2025/sig_core/sig_core/2
- Nikolic, N. (2024, June 17). *Better Thinking #159 – John Sweller on the cognitive load theory*. <https://www.youtube.com/watch?v=6iR9c48YLM0>
- Obama, B. (2014). Presidential proclamation — National Day of Making, 2014. *The White House, Office of the Press Secretary*. <https://obamawhitehouse.archives.gov/the-press-office/2014/06/17/presidential-proclamation-national-day-making-2014>
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Aspell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35, 27730–27744. <https://doi.org/10.48550/arXiv.2203.02155>
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.
- Partarakis, N., & Zabulis, X. (2024). Applying cognitive load theory to eLearning of crafts. *Multimodal Technologies and Interaction*, 8(1), Article 2. <https://doi.org/10.3390/mti8010002>
- Peterson, L., & Peterson, M. J. (1959). Short-term retention of individual verbal items. *Journal of Experimental Psychology*, 58(3), 193–198. <https://doi.org/10.1037/h0049234>
- Polanyi, M. (1966). *The tacit dimension*. Doubleday.
- Pollock, E., Chandler, P., & Sweller, J. (2002). Assimilating complex information. *Learning and Instruction*, 12(1), 61–86. [https://doi.org/10.1016/S0959-4752\(01\)00016-0](https://doi.org/10.1016/S0959-4752(01)00016-0)
- Puneeth. (2023, January 9). *Learn Arduino using AI (ChatGPT)*. Wokwi Blog. <https://blog.wokwi.com/learn-arduino-using-ai-chatgpt/>
- Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., & Finn, C. (2023). Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36. https://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html
- Resnick, M. (2007). All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. *Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition*, 1–6. <https://doi.org/10.1145/1254960.1254961>
- Resnick, M. (2017). *Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play*. The MIT Press. <https://doi.org/10.7551/mitpress/11017.001.0001>

- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60–67. <https://doi.org/10.1145/1592761.1592779>
- Rourke, A., & Sweller, J. (2009). The worked-example effect using ill-defined problems: Learning to recognise designers’ styles. *Learning and Instruction*, 19(2), 185–199. <https://doi.org/10.1016/j.learninstruc.2008.03.006>
- Schaeffer, R., Miranda, B., & Koyejo, S. (2023). Are emergent abilities of large language models a mirage? *Advances in Neural Information Processing Systems*, 36. <https://doi.org/10.48550/arXiv.2304.15004>
- Schön, D. A. (2017). *The reflective practitioner: How professionals think in action*. Routledge. <https://doi.org/10.4324/9781315237473> (Original work published 1983)
- Shanahan, M. (2024). Talking about large language models. *Communications of the ACM*, 67(2), 68–79. <https://doi.org/10.1145/3624724>
- Sharma, M., Tong, M., Korbak, T., Duvenaud, D., Askell, A., Bowman, S. R., Cheng, N., Durmus, E., Hatfield-Dodds, Z., Johnston, S. R., Kravec, S., Maxwell, T., McCandlish, S., Ndousse, K., Rausch, O., Schiefer, N., Yan, D., Zhang, M., & Perez, E. (2024). Towards understanding sycophancy in language models. *Proceedings of the International Conference on Learning Representations (ICLR)*. <https://doi.org/10.48550/arXiv.2310.13548>
- Shaw, L. (2025). Using AI for quilt design. <https://laurashawquilts.substack.com/p/using-ai-for-quilt-design>
- Shihab, M. I. H., Hundhausen, C., Tariq, A., Haque, S., Qiao, Y., & Mulanda, B. W. (2025). The effects of GitHub Copilot on computing students’ programming effectiveness, efficiency, and processes in brownfield coding tasks. *Proceedings of the 2025 ACM Conference on International Computing Education Research V.1*, 407–420. <https://doi.org/10.1145/3702652.3744219>
- Skulmowski, A., & Rey, G. D. (2017). Measuring cognitive load in embodied learning settings. *Frontiers in Psychology*, 8, Article 1191. <https://doi.org/10.3389/fpsyg.2017.01191>
- Song, I., Jeon, E., & Lee, J. (2025, December 31). PCEval: A benchmark for evaluating physical computing capabilities of large language models. <https://doi.org/10.48550/arXiv.2601.02404>
- Stadler, M., Bannert, M., & Sailer, M. (2024). Cognitive ease at a cost: LLMs reduce mental effort but compromise depth in student scientific inquiry. *Computers in Human Behavior*, 160, Article 108386. <https://doi.org/10.1016/j.chb.2024.108386>
- Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4

- Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction*, 4(4), 295–312. [https://doi.org/10.1016/0959-4752\(94\)90003-5](https://doi.org/10.1016/0959-4752(94)90003-5)
- Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational Psychology Review*, 22(2), 123–138. <https://doi.org/10.1007/s10648-010-9128-5>
- Sweller, J. (2023). The development of cognitive load theory: Replication crises and incorporation of other theories can lead to theory expansion. *Educational Psychology Review*, 35(4), Article 95. <https://doi.org/10.1007/s10648-023-09817-2>
- Sweller, J. (2024). Cognitive load theory and individual differences. *Learning and Individual Differences*, 110, Article 102423. <https://doi.org/10.1016/j.lindif.2024.102423>
- Sweller, J. (2025). An integrated human cognitive architecture. *Educational Psychology Review*, 37(4), Article 108, 1–19. <https://doi.org/10.1007/s10648-025-10089-1>
- Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 2(1), 59–89. https://doi.org/10.1207/s1532690xc0201_3
- Sweller, J., & Levine, M. (1982). Effects of goal specificity on means–ends analysis and learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 8(5), 463–474. <https://doi.org/10.1037/0278-7393.8.5.463>
- Sweller, J., Mawer, R. F., & Howe, W. (1982). Consequences of history-cued and means-end strategies in problem solving. *The American Journal of Psychology*, 95(3), 455–483. <https://doi.org/10.2307/1422136>
- Sweller, J., van Merriënboer, J. J. G., & Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. *Educational Psychology Review*, 31(2), 261–292. <https://doi.org/10.1007/s10648-019-09465-5>
- Sweller, J., van Merriënboer, J. J. G., & Paas, F. G. W. C. (1998). Cognitive architecture and instructional design. *Educational Psychology Review*, 10(3), 251–296. <https://doi.org/10.1023/A:1022193728205>
- Tafferner, Z., Illés, B., Krammer, O., & Géczy, A. (2023). Can ChatGPT help in electronics research and development? A case study with applied sensors. *Sensors*, 23(10), Article 4879. <https://doi.org/10.3390/s23104879>
- Tang, N., Chen, M., Ning, Z., Bansal, A., Huang, Y., McMillan, C., & Li, T. J.-J. (2024). Developer behaviors in validating and repairing LLM-generated code using IDE and eye tracking. *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 40–46. <https://par.nsf.gov/servlets/purl/10583646>
- Tankelevitch, L., Kewenig, V., Simkute, A., Scott, A. E., Sarkar, A., Sellen, A., & Rintel, S. (2024). The metacognitive demands and opportunities of generative AI. *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 1–24. <https://doi.org/10.1145/3613904.3642902>

- Tarmizi, R. A., & Sweller, J. (1988). Guidance during mathematical problem solving. *Journal of Educational Psychology*, 80(4), 424–436. <https://doi.org/10.1037/0022-0663.80.4.424>
- Tedre, M., Kahila, J., & Vartiainen, H. (2023). Exploration on how co-designing with AI facilitates critical evaluation of ethics of AI in craft education. In E. Langran, P. Christensen, & J. Sanson (Eds.), *Proceedings of the Society for Information Technology & Teacher Education International Conference (SITE 2023)* (pp. 2289–2296). Association for the Advancement of Computing in Education (AACE). <https://www.learntechlib.org/primary/p/222124/>
- Testa, A., Nambi, A., Simeone, A., Zecca, M., Paoli, A., Priarone, P. C., & Settineri, L. (2026). Cognitive load-driven instruction simplification with generative AI for inclusive manufacturing. *Procedia CIRP*, 139, 29–34. <https://doi.org/10.1016/j.procir.2025.09.011>
- Tofel-Grehl, C., & Feldon, D. F. (2023). Measuring the viability of maker technology adoption within classrooms: Evaluating teacher instructional and cognitive load through constructionism and making. *FabLearn / Constructionism 2023 Full and Short Research Papers*, 1–4. <https://doi.org/10.1145/3615430.3615448>
- Turkle, S., & Papert, S. (1992). Epistemological pluralism: Styles and voices within the computer culture [Reprinted from Signs, 16(1), 1990]. *Humanistic Mathematics Network Journal*, (7), 48–59. <https://doi.org/10.5642/hmnj.199201.07.08>
- Van der Velden, M. (2021). ‘Fixing the world one thing at a time’: Community repair and a sustainable circular economy. *Journal of Cleaner Production*, 304, Article 127151. <https://doi.org/10.1016/j.jclepro.2021.127151>
- van Merriënboer, J. J. G., & Sweller, J. (2010). Cognitive load theory in health professional education: Design principles and strategies. *Medical Education*, 44(1), 85–93. <https://doi.org/10.1111/j.1365-2923.2009.03498.x>
- Vasko, S. E. (2023, September 21). *Creating ceramic glazing ideas with ChatGPT-3.5*. <http://www.stephanievasko.com/blog/2023/9/21/creating-ceramic-glazing-ideas-with-chatgpt-35>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30. <https://doi.org/10.48550/arXiv.1706.03762>
- Vergara, K., & Herskovic, V. (2025). Boxy Board: Supporting novice learners through a progressive disclosure approach for physical computing. *International Journal of Human-Computer Interaction*. <https://doi.org/10.1080/10447318.2025.2563744>
- Vossoughi, S., Hooper, P. K., & Escudé, M. (2016). Making through the lens of culture and power: Toward transformative visions for educational equity. *Harvard Educational Review*, 86(2), 206–232. <https://doi.org/10.17763/0017-8055.86.2.206>

- Wang, P., Liu, H., Zou, L., & Paas, F. (2026). Overloaded minds and machines: A cognitive load framework for human-AI symbiosis. *Artificial Intelligence Review*, 59(3), Article 109. <https://doi.org/10.1007/s10462-026-11510-z>
- Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., & Fedus, W. (2022). Emergent abilities of large language models. *Transactions on Machine Learning Research*. <https://doi.org/10.48550/arXiv.2206.07682>
- Winters, S., Farnsworth, K., Berry, D., Ellard, S., Glazewski, K., & Brush, T. (2023). Supporting middle school students in a problem-based makerspace: Investigating distributed scaffolding. *Interactive Learning Environments*, 31(6), 3396–3408. <https://doi.org/10.1080/10494820.2021.1928709>
- Woody. (2025, September 21). *First experiences using ChatGPT for Arduino code*. Woody's Workshop. <https://altrish.co.uk/2025/09/21/first-experiences-using-chatgpt-for-arduino-code/>
- Xie, H., & Ju, F. (2025). Text-to-CadQuery: A new paradigm for CAD generation with scalable large model capabilities. <https://doi.org/10.48550/arXiv.2505.06507>
- Xu, R., Cao, J., Wu, M., Zhong, W., Lu, Y., He, B., Han, X., Cheung, S.-C., & Sun, L. (2026). EmbedAgent: Benchmarking large language models in embedded system development. <https://doi.org/10.48550/arXiv.2506.11003>
- Yang, C., Shi, Y., Ma, Q., Liu, M. X., Kästner, C., & Wu, T. (2025). What prompts don't say: Understanding and managing underspecification in LLM prompts. <https://doi.org/10.48550/arXiv.2505.13360>
- Yao, G., & Fan, L. (2025). Cognitive load scale for AI-assisted L2 writing: Scale development and validation. *Frontiers in Psychology*, 16, Article 1666974. <https://doi.org/10.3389/fpsyg.2025.1666974>
- Yingying. (2025, May 5). *Programming hardware with GenAI*. <https://blog.yingyingz.com/p/programming-hardware-with-genai-built>
- Zamfirescu-Pereira, J. D., Wong, R. Y., Hartmann, B., & Yang, Q. (2023). Why Johnny can't prompt: How non-AI experts try (and fail) to design LLM prompts. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, Article 437. <https://doi.org/10.1145/3544548.3581388>
- Zhou, H., Zhang, Y., Zhang, Z., Zhang, W., Wang, C., & An, P. (2026). ProjecTA: A semi-humanoid robotic teaching assistant with in-situ projection for guided tours. *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, 1–24. <https://doi.org/10.1145/3772318.3790577>

A. Visualizations

A.1 Field Intersection Diagram

Figure A.1 maps the three research areas this thesis combines and labels the pairwise intersections covered in Chapter 2. The center of the diagram, where all three fields overlap, is the space Chapter 3 opens.

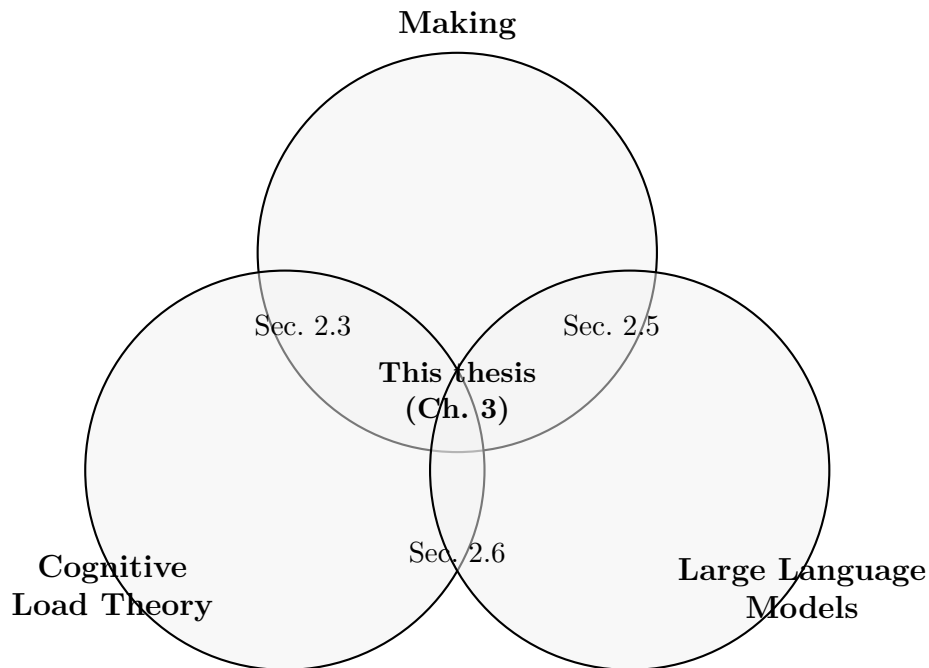


Figure A.1: Triple intersection of Making, Cognitive Load Theory, and Large Language Models. Pairwise intersections correspond to literature reviews in Chapter 2. The center is the space this thesis opens.

A.2 Cascade of Translations

Table A.1 maps the six steps from intention to artifact, naming the element interactivity characteristic of each step and how LLM mediation shifts load at that point. Introduced in Section 3.1 and developed in Section 3.2.

Table A.1: The cascade of translations: element interactivity and LLM effect at each step.

Step	What making demands	Element interactivity	LLM effect
Vision	Hold function, aesthetic, material constraints, and use conditions simultaneously.	High: many interdependent requirements that must cohere.	Can externalize requirements and surface unarticulated constraints. Introduces language bottleneck and intent-drift risk.
Decomposition	Translate whole artifact into sub-components while keeping all interdependencies consistent.	High: every part must stay consistent with the vision and with every other part.	Can produce a structured breakdown that absorbs relational interactivity. Introduces verification load (completeness/fit) and context management load.
Domain knowledge retrieval	Recall specialized facts and principles across potentially unfamiliar domains while holding task requirements active.	Variable: scales inversely with the maker’s existing schemas. Highest for novices in unfamiliar domains.	Can return probable factual answers and supply vocabulary. Verification load is likely highest here, since fluent output is hardest to catch where schemas are thin.

Step	What making demands	Element interactivity	LLM effect
Procedure	Coordinate materials, tools, and action sequence in state-dependent order, tracking what has been done.	High: correct step order depends on material state, which depends on prior steps.	Can produce worked-example output that bypasses means-ends search. Introduces prompt formulation load and modal gap (procedure must become physical action).
Physical execution	Manage body, material, and tools in real time, applying tacit knowledge and responding to material feedback.	High: embodied, tacit, and material-responsive, resisting decomposition into explicit elements.	LLM does not reach this step. Modal gap load applies at the boundary between linguistic output and physical action.
Evaluation	Compare artifact to vision, identify discrepancy type and level, and feed back to the appropriate cascade step.	Variable: depends on the verification schemas available, and rises when the failure mode is ambiguous.	Can structure evaluation criteria and interpret photographs (multimodal input). Verification load and language bottleneck apply.

A.3 Six Load Types

Table A.2 summarizes the six load-type constructs introduced in Section 3.2. The first three intensify costs that existed before LLM mediation. The second three are structurally new at the maker-LLM interface.

Table A.2: Six load types introduced by LLM mediation in making contexts.

Load type	Mechanism	Novice / expert differential
<i>Verification load</i>	Evaluating LLM output for correctness in this context, for these materials, with this tool. The processing-fluency heuristic inflates perceived accuracy of fluent output.	Likely highest for novices, who lack schemas to detect errors and may elaborate wrong schemas through dialogue. Experts may verify quickly per error, though cost could accumulate across sessions where error rates are high.
<i>Modal gap load</i>	Cost of every transition between linguistic LLM output and physical action. Two sources: (1) tacit/pre-linguistic knowledge was never stated; (2) holding text in working memory while attending to physical action loads two systems simultaneously.	Gap would be widest for novices, where no tacit knowledge bridges registers and every linguistic-to-physical transition is a fresh translation. For experts it should be narrower, though an articulation mismatch may persist.
<i>Intent monitoring</i>	Project-level tracking of whether direction remains the maker’s own. Individual suggestions can be individually useful while cumulatively drifting from original intent.	Likely harder for novices, whose vision may be less defined and who could be less able to detect drift. Experts may have clearer intent anchors but are probably not immune to turn-by-turn accumulation.

Load type	Mechanism	Novice / expert differential
<i>Language bottleneck</i>	Two forms: (1) vocabulary gap (novice lacks domain terms to describe the need); (2) articulation cost (expert’s knowledge is procedural/tacit and resists declarative restatement).	Vocabulary gap is likely highest for novices. Articulation cost could be highest for experts in embodied crafts (ceramics, textiles, woodworking), where competence may never have required linguistic form.
<i>Prompt formulation load</i>	Getting useful output requires LLM-specific interaction knowledge beyond domain expertise: understanding probabilistic generation, managing underspecification, knowing where fluent output cannot be trusted.	Largely independent of domain expertise. A domain expert and a domain novice can both face high prompt formulation load, on distinct grounds. Making expertise and LLM-interaction expertise are treated as independent axes.
<i>Context management load</i>	Maker must track (a) current project state and (b) what the LLM currently holds. Without memory features: full context reconstruction each session. With memory features: verification sub-task (did it remember correctly?). Burden changes form rather than disappearing.	Likely comparable across expertise levels, scaling with project complexity and session length rather than domain knowledge.

B. AI Tooling Setup

This appendix documents the AI-assisted setup described in the GenAI Disclosure. It reproduces the shell script that converted source PDFs to markdown for reading and describes the two custom skills that encoded my revision and submission-check protocols.

B.1 Literature Conversion Pipeline

Source papers were collected as PDFs in a `literature/` folder per chapter section. Each PDF was converted to markdown with `marker`, a local document-to-markdown converter run inside a Python virtual environment, for the LLM to have context. The script below batches the conversion for one section's folder and copies the resulting markdown into a `md_output/` folder. Only extractions produced this way were treated as readable sources, and no paper was cited before its extraction had been read.

```
#!/bin/bash
# Usage: ./extract_pdfs.sh "2.3 Making x CLT"
# Extracts all PDFs in a section's Lit folder to markdown files.

SECTION="$1"
LIT_DIR=~/Desktop/Thesis/"$SECTION"/literature
OUT_DIR=~/Desktop/Thesis/"$SECTION"/md_output
MARKER_OUT=~/.marker-project/output
MARKER_BIN=~/.marker-project/venv/bin/marker_single

mkdir -p "$OUT_DIR"

# Snapshot existing output dirs so only new files are copied
BEFORE=$(ls "$MARKER_OUT")

for f in "$LIT_DIR"/*.pdf; do
    [ -f "$f" ] || continue
    "$MARKER_BIN" "$f" --output_dir "$MARKER_OUT"
    echo "Done: $f"
done

# Copying only from dirs that did not exist before this run
for dir in "$MARKER_OUT"/*/; do
    dirname=$(basename "$dir")
    if ! echo "$BEFORE" | grep -qx "$dirname"; then
        find "$dir" -name "*.md" -exec cp {} "$OUT_DIR"/ \;
    fi
done
```

```
done
```

```
echo "All done. MDs in $OUT_DIR"
```

B.2 Custom Skills

I built two of my working protocols into custom skills. Each runs as a fixed sequence. The same checks run in the same order every time.

`/revision` conducts a structured review at the passage level. First, it identifies the purpose of the section and its objectives. Second, it evaluates the voice and style, noting any ambiguities, redundancies, or areas where plain language could improve clarity. Third, it examines the argument and scope to detect any overstatements or deviations from a descriptive tone. Fourth, it reviews citations by checking the markdown document for inaccuracies, incorrect attributions, or unsupported claims. Fifth, it considers the flow and transitions within the text. Sixth, it assesses how a critical yet sympathetic reader might perceive it. Seventh, it evaluates the balance and bias, ensuring all sides are represented where appropriate. Finally, it returns a list of all critical issues.

`/submission-check` operates at the manuscript level prior to submission or chapter handover, addressing issues relevant to the entire document and focusing on aspects that matter for the whole manuscript. Its eleven clusters are examined in separate passes: argument and research question, claims and sources, originality, fairness of critique, theoretical consistency, register and scope, interdisciplinarity, presentation quality, balance and bias, voice and style, and final mechanics. All findings are reported first before any suggestions are proposed.