The Age of AI Agents: Understanding A2A, MCP, and Agentic Experience

Artificial intelligence is evolving beyond simple chatbots into **agent-based systems** that can act autonomously on our behalf. In these systems, AI "agents" can **plan steps, use tools, and even collaborate with other agents** to achieve a goal. This new paradigm is changing how we interact with technology. To understand it, we need to explain a few key concepts: the Agent-to-Agent (A2A) protocol, the Model Context Protocol (MCP), and the idea of *agentic experience* vs *user experience*. We'll also look at what agent-based systems are in general, and then explore real-world applications in areas like education, healthcare, customer service, software development, and creative work.

Agent-Based Systems: What Are AI Agents?

AI agents are essentially AI-powered programs that can **take initiative and perform tasks** for you, rather than just respond passively to single commands. Unlike a basic chatbot (which only replies to queries), an agent has a sort of "goal-oriented" behavior: it can **decide what actions to take, in what sequence, to fulfill a given task**. One definition calls an AI agent "a self-directed autonomous application" that leverages a large language model's reasoning ability, uses tools, and draws on context from various data sources to carry out tasks . In other words, the agent has an AI "brain" (often a large language model) but also the ability to **remember context, call on external tools or data, and adjust its plan** as it works. Agents can "think" through a problem, **plan multiple steps, execute actions, then observe the results and refine their approach** – all with minimal human intervention .

Think of an AI agent like a **digital assistant** you could delegate work to, the way you might delegate tasks to a human assistant. For example, instead of you manually browsing flight websites, comparing dates, and booking a ticket, you could simply tell an AI agent, "Book me a flight to New York next Monday morning." The agent would then handle *all the steps*: searching different airlines, finding a flight that meets your criteria, and completing the booking

(possibly even putting the event on your calendar). The agent **figures out the sequence of actions** needed to fulfill your request. This ability to independently pursue goals is what makes a system "agent-based."

Early experiments like AutoGPT and others have demonstrated simple versions of this, where an AI tries to pursue a goal by breaking it into sub-tasks. Modern agent-based frameworks are making these systems more reliable and useful. However, enabling AI agents to work in the real world (and in complex organizations) requires some **common protocols** – shared "languages" and standards so that agents can safely communicate and use the tools they need. Two of these emerging standards are **A2A** and **MCP**, described below.

The Agent-to-Agent (A2A) Protocol: How AI Agents Talk to Each Other

As we deploy more AI agents, often we'll want them to **work together**. In a company, you might have one agent that handles scheduling, another that manages inventory, another that answers customer inquiries, and so on. For complex tasks, these agents need a way to **communicate and coordinate** with each other – even if they were built by different vendors or in different programming frameworks. The **Agent-to-Agent protocol (A2A)** is essentially a **standard method for agents to "talk" to other agents** and collaborate on tasks.

According to Google (who introduced A2A in 2025), this protocol allows AI agents to *"communicate with each other, securely exchange information, and coordinate actions"* across different platforms . In simpler terms, A2A is like giving every AI agent a common language or a shared phone line to **send requests and updates to one another**. An easy analogy is to imagine A2A as a **network cable or a social network connecting AI agents** – it lets them find each other and directly share work . For example, with A2A, your calendar-scheduling agent could automatically message your travel-booking agent when a flight you're on gets delayed, and the two agents could work out a new plan (rescheduling meetings, rebooking a flight, etc.) without bothering you for each detail.

The A2A protocol is **vendor-neutral and open**, meaning any company or developer can use it to make their AI agents interoperable. This is important because it breaks agents out of silos. If your customer service agent from Vendor A can call on an order-processing agent from Vendor B via A2A, it doesn't matter that they were built by different teams – they can still cooperate. Google and dozens of other tech companies (Atlassian, Salesforce, Dropbox,

and many more) have backed A2A as a standard . The protocol handles things like how agents advertise what they can do, how they **send task requests** to each other, and how they **exchange the results** of those tasks. It's also designed with security in mind (so agents only do authorized things) and supports long-running processes (agents can keep a task going and update each other over time) .

In summary, A2A enables a **multi-agent ecosystem**. It's what allows one autonomous AI agent to delegate subtasks to another agent and then integrate the results. This is a big shift from today's applications – instead of one monolithic AI trying to do everything, you can have specialized agents that **team up**, each handling what it's best at. A2A is the **communication backbone** that makes such teamwork possible.

The Model Context Protocol (MCP): How AI Agents Connect to Tools and Data

While A2A lets agents talk to each other, an AI agent often also needs to **talk to external tools and databases** to be effective. Think of all the actions an AI assistant might need to do: query a database, fetch a file from Google Drive, call an external API (like checking the weather or stock prices), or use a company's internal knowledge base. Traditionally, connecting an AI system to each of these tools requires **custom integration** – a lot of ad-hoc coding and wiring up APIs. This is slow and doesn't scale well if you have many tools. What if there were a single *standard way* to plug any tool or data source into an AI agent? That's the idea of the **Model Context Protocol (MCP)**.

MCP is an **open standard (introduced by Anthropic in late 2024)** that provides a universal interface between AI agents (or AI models) and external data/tools . One article described MCP this way: *"AI agents promise to transform how work gets done — but without a shared language, they struggle to operate across tools and systems. Enter Model Context Protocol (MCP), a new open-source standard…for solving this challenge."* In essence, MCP defines a **common format for tools to describe themselves to AI, and for AI to invoke those tools**. It's like a **"universal adapter"** or a **USB-C port** for AI models . Instead of having to custom-code how an agent interacts with each different service (one for Slack, a different one for GitHub, another for your database, etc.), any service that has an MCP interface can be accessed in the **same standardized way**.

Here's how it works in practice: An AI agent can query an **MCP server**, asking for a particular operation or piece of data. The MCP server acts as a bridge to

the actual tool or database. The agent's request and the tool's response are exchanged in a structured format (often JSON) that all MCP-compatible systems understand . For example, suppose an AI agent needs to get customer information from a CRM database and also send an email. Without MCP, the developer would have to integrate the CRM API and an email API separately and handle auth, data formatting, etc. With MCP, the CRM and email services (if they support MCP) would each have a little "adapter" (an MCP server) that knows how to talk to them. The AI agent just sends a standard MCP request like *"fetch customer record 123"* or *"send email to X with body Y"*, and the MCP layer handles routing that to the right service and returning the result.

The **advantage** is a huge increase in flexibility and scalability. As one expert noted, *"Without MCP… every time an agent needs to do something in the world — whether fetching a file, querying a database, or invoking an API — developers would have to wire up a custom integration… tedious and not scalable."* MCP removes that bottleneck. Tools **self-describe their inputs/outputs** to the agent, so if an agent knows the MCP protocol, it can plug into any new tool that also speaks MCP without additional coding. This means AI agents can have access to a **wide ecosystem of tools and real-time data** easily, which in turn means they can perform much more complex tasks reliably. Anthropic initially provided MCP connectors for things like Google Drive, Slack, GitHub, databases, etc., and the community quickly expanded with hundreds more connectors . Even OpenAI announced support for MCP in its products, showing how it's becoming a cross-industry standard .

In short, MCP **"feeds the mind" of AI agents with the context and capabilities they need** from external sources . If we continue our assistant analogy, MCP is like giving your AI assistant a toolbox full of standardized instruments – one tool to read any database, one to use any web service, one to control any software – as opposed to the assistant having to build a new tool from scratch for each job. With A2A and MCP together, an agent can both **access information/tools (via MCP)** and **coordinate with other agents (via A2A)**. These protocols are complementary and together enable much more powerful "autonomous" behavior in complex environments .

Agentic Experience vs. User Experience

So what does all this mean for **users**? Traditionally, when we use software, we talk about *user experience (UX)* – basically, how a human interacts with an interface to achieve their goal. In a classic user experience, **the user does all**

the driving: you click menus, fill out forms, follow steps in an app. The software may assist you, but *you* are in control of each action. This can be thought of as the user manually navigating an "obstacle course" of interface steps to get something done .

With agent-based systems, a new paradigm is emerging, sometimes called *agentic experience* (AX) or *agentive UX*. In an **agentic experience**, the user's role shifts from performing every step to **setting goals and supervising**, while the AI agent handles the detailed execution. As design leader John Maeda describes it, *"Rather than forcing users to learn and master complex interfaces, agents allow a user to directly express their intent and have the system handle the complexity behind the scenes."* In other words, you just tell the AI what outcome you want, and **the agent figures out the "how."** This represents a fundamental shift from the traditional user experience (UX) to what we might call an agent or *agentic* experience .

Another way to look at it: In a conventional digital experience, the software provides the tools and information, but *the user* must do all the thinking, planning, and deciding. By contrast, an agentic experience turns the software into more of a **collaborator or even a mentor** that carries some of that load. One UX expert noted that in an agentic UX, *"a user is continuously supported by a built-in agent that pulls real weight, transforming every digital experience from a solo endeavor to a partner-based journey."* Instead of the user alone navigating the app, you have an AI partner working alongside you or on your behalf. For example, if you're using an agentic writing app, you might simply say, "Help me draft a report about climate change's impact on agriculture," and the AI agent will research, outline, and maybe even start writing a draft for you. You're still in charge – you review the draft, you guide the agent if needed – but the **experience** is more about you supervising a capable assistant rather than clicking every menu yourself.

This agentic approach can make complex tasks feel **much simpler for the user**. It aligns with the idea of *"simplicity through automation"*: the complexity is still there, but it's under the hood, managed by the agent. However, it also introduces new considerations. Trust becomes a big factor – users have to trust the agent to act correctly on their behalf . Design-wise, we have to think about how the agent communicates what it's doing, how the user can correct or guide it, and how to ensure the agent's actions align with the user's true intent and ethical boundaries . In essence, the focus expands from just designing a great user-facing interface, to also designing the *agent's behavior and decision-*

making in a way that creates a positive overall experience. This is a new frontier for product designers and engineers: we're no longer just crafting buttons and screens, we're crafting **agent behaviors** and interactions that might largely be invisible to the user except for the end result.

To make this concrete, consider the difference between a classic travel-booking website and an agentic travel planner. In the classic UX, you (the user) would go through multiple websites or forms: search flights, compare options, book one; search hotels, filter and book; mark your calendar; etc. In an agentic experience, you might simply chat with an AI: *"I need to plan a one-week vacation to a quiet beach destination in July, with a hotel that has a pool and no all-inclusive packages. Once I pick an option, book the flights and hotel, block my work calendar, and arrange pet boarding."* Then the **AI agents handle it** – one agent might search travel info, another handles booking transactions, they might use A2A to coordinate and MCP to pull data from travel sites – and they come back with, say, three itinerary options for you. You choose one, and the agents execute all the bookings and confirmations. Your experience as a user is just expressing your goal and making high-level decisions, rather than wrestling with five different apps. This illustrates how *agentic UX* can dramatically streamline complex tasks. It's more **goal-oriented and conversational** vs. tool-oriented and manual.

Real-World Use Cases and Applications Across Industries

The move toward agent-based, agentic systems isn't just theoretical – it's already underway in many industries. Below, we explore how these concepts (AI agents + A2A + MCP, enabling an agentic experience) are being applied in practice. The common theme is that **AI agents are expanding what technology can do by taking on complex, multi-step tasks** in a variety of domains. This can **augment human effort**, automate drudgery, and provide more personalized or intelligent services. We will look at examples in education, healthcare, customer service, software development, and creative work (and mention others). In each case, imagine how having autonomous yet collaborative agents (with access to the right data and tools) changes the game.

Education: Personalized Tutors and Smart Automation

In education, AI agents are helping both learners and educators by providing more personalized support and automating routine tasks. For example, an AI tutoring agent can **monitor a student's performance** on assignments in real time and detect if the student is struggling on a topic. It can then *proactively* offer help – perhaps by providing extra practice problems or alerting the teacher with a recommendation for intervention . This kind of early, individualized support was hard to achieve in traditional classrooms, but an agent can watch each student's progress continuously without getting tired.

At the same time, other agents can take care of **administrative burdens** in education. Schools are using agents to schedule study sessions or meetings, send out reminder notifications to students, track attendance, and even handle course registration paperwork . By autonomously dealing with these repetitive processes, the AI frees up teachers and staff to focus on teaching and mentoring rather than form-filling. Education agents can also create **adaptive learning pathways** for students: for instance, adjusting the pace or difficulty of material based on the learner's needs. If a student masters a concept quickly, the agent can accelerate them to the next topic; if they are having trouble, the agent can provide remedial content or a different explanation style . Each student, in effect, could have a personal AI tutor/assistant that tailors the learning experience – something that isn't feasible with one teacher for many students. This **hyper-personalization** is a promising use of agentic AI in education, aiming to keep every student appropriately challenged and supported. In short, agent-based systems in education are making learning more individualized and efficient, while reducing the workload on educators for mundane tasks.

Healthcare: Continuous Care and Intelligent Support

In healthcare, agentic AI systems are being developed to assist both medical professionals and patients. One major application is **continuous patient monitoring and proactive care**. Imagine a health agent that monitors data from a patient's wearable devices and health records 24/7. If it detects an anomaly or a risk (say, the patient's heart rate and blood pressure indicate a potential issue), the agent can automatically alert the patient and their doctor, *and even take initial actions* like scheduling an earlier appointment or suggesting a dosage adjustment (within doctors' preset guidelines). This means potential problems can be caught and addressed **earlier**, possibly preventing serious complications. Such an agent could be life-saving for chronic disease management – for example, helping diabetics manage insulin or monitoring

cardiac patients for early warning signs. It's like having a virtual nurse that never sleeps, keeping an eye on you and coordinating care.

AI agents are also supercharging **diagnostic and analytical tasks** in medicine. A doctor can have an AI agent that, with a given set of patient symptoms, history, and lab results, quickly scans medical databases and research literature to suggest likely diagnoses and treatment options . The agent might say, "Based on the data, here are the top 3 possible diagnoses to consider, and here's why, along with relevant recent research." This doesn't replace the doctor's judgment, but it **augments their decision-making** with speedy research and pattern recognition across millions of medical cases. In complex or rare cases, this can be a game-changer, ensuring no important detail is overlooked.

Furthermore, healthcare agents can handle a lot of **administrative and logistical work** that often bogs down clinics. There are agents for automating appointment scheduling (finding an open slot that fits patient and doctor schedules), sending reminders or follow-up instructions to patients, processing insurance claims and paperwork, and updating medical records. By taking over these time-consuming tasks, AI agents help reduce human error and free healthcare staff to concentrate on direct patient care . For instance, when a doctor orders a lab test, an agent could automatically ensure the lab request is entered, follow up on getting the results, and notify the patient of the next steps – all without a human clerk doing data entry. In essence, **agent-based systems in healthcare aim to provide more vigilant, personalized care and to streamline operations**. This can lead to better patient outcomes (through timely interventions and data-driven insights) and a reduction in burnout for medical professionals who spend less time on paperwork.

Customer Service: Automated Problem-Solving Agents

Customer service is an area many of us encounter, and it's being transformed by AI agents that go beyond simple chatbots. Traditionally, a customer service bot could answer frequently asked questions, and anything complex would be handed off to a human. Now, with agentic AI, we're seeing **AI agents that can resolve complex customer issues end-to-end**. For example, consider a package delivery problem: you receive a message that your package is delayed. Instead of just apologizing, an advanced customer service agent could **take initiative** to fix the situation. It might automatically check the shipping system for where the package is and why it's delayed, then figure out

a solution – perhaps it can order a replacement item to be sent by overnight mail, or offer you a discount for the inconvenience . The agent would then update the order in the database, initiate the refund or new shipment, and send you a confirmation, all without a human representative manually doing those steps .

In this scenario, the user (customer) barely has to do anything – the agent not only communicates but also **executes the solution**. A2A and MCP play a role here behind the scenes: the customer service agent might call on an inventory agent or a shipping agent via A2A to coordinate the fix, and use MCP to access the order database and shipping API for information and action. The result is faster service and happier customers, because problems are solved in minutes rather than days.

Agents in customer support can also handle things like **answering complex queries by pulling in information from various company databases**, scheduling appointments or callbacks, or even upselling and personalization (e.g., a sales agent that knows your preferences and can suggest the right product). Importantly, these agents can operate 24/7 and handle surges in customer inquiries without long wait times. Companies like banks, e-commerce retailers, and software providers are exploring agentic AI to automate a lot of customer-facing processes. By letting AI agents take care of routine or data-driven tasks (checking account status, resetting passwords, troubleshooting common device issues, etc.), human support staff are free to focus on the toughest or most sensitive cases. This combination can significantly **improve the user experience** – you get quicker help – and reduce costs. Enterprises have reported using autonomous agents for everything from *"ordering new laptops" for employees (IT support)* to assisting human service reps by pre-fetching all relevant customer info and possible solutions before the rep even picks up the phone . Overall, agent-based customer service promises more **efficient, responsive, and even proactive support** compared to the old reactive helpdesk model.

Software Development: AI Agents as Coding Co-Pilots

Software developers are also benefiting from AI agents, turning coding into a more collaborative effort between human and machine. We already have tools like GitHub Copilot that suggest code snippets, but agentic AI in software development goes further. An **AI coding agent** can take a task description (e.g., "Implement a function to calculate shipping cost based on weight and

destination") and then proceed to *write the code*, test it, and refine it in a loop. For instance, an agent might generate an initial code implementation, then automatically generate unit tests to check that code, run the tests, see some tests failed, and then modify the code to fix the bugs – all without a human intervening in that cycle . It's as if the agent is a junior developer who not only writes code but also debugs and verifies it.

Using the Model Context Protocol, the agent can plug into development tools: it could retrieve relevant pieces of the existing codebase (to understand context or reuse functions), query documentation, or call an API to run the code in a sandbox. There might be multiple agents in play: one agent breaks down a big feature into subtasks, another specializes in writing code, another in testing. Thanks to A2A, these agents can coordinate – for example, a "planner" agent could delegate "WriteFunctionX" to a "coder" agent and "TestFunctionX" to a "tester" agent, and then integrate the results. All this can dramatically **speed up the development process**. Routine code (like boilerplate or repetitive functions) can be handled by agents, so human developers spend more time on design, architecture, or tricky logic. Early adopters have agents generating substantial portions of code and test suites automatically, which *"accelerates the development process, reduces coding errors, and frees up developers to focus on more complex challenges."*

Another use is in code maintenance and DevOps. Imagine an AI agent that scans a code repository for potential bugs or security vulnerabilities, fixes them, and opens a pull request with explanations. Or an agent that can set up your continuous integration pipeline by talking to cloud APIs (again using MCP to standardize those actions). Companies like Replit and Sourcegraph have started integrating something like MCP to let AI agents retrieve code from repositories and understand project context , so that when they generate code, it fits the project's style and requirements. It's not hard to envision a near future where a developer can say, "AI, please create a simple mobile app that does X," and a team of backend agent, frontend agent, and testing agent (all communicating via A2A) collaborate to produce a first draft of the app. The human developer would then just refine and review it. This moves coding toward a higher level of abstraction, where humans define goals and AI handles more of the grunt work of implementation. It doesn't replace programmers, but it **augments their productivity** – much like power tools extend the abilities of a craftsman.

Creative Work: Generating Content and Ideas with Agents

Creative industries – such as content writing, design, marketing, and entertainment – are also exploring agentic AI to push the boundaries of what one person can create. Generative AI (like GPT-4, DALL-E, etc.) is already used for tasks like drafting copy or creating images, but agent-based systems can take this further by **orchestrating entire creative workflows**. For example, consider content marketing: an AI agent could be given the goal "produce a monthly newsletter summarizing the latest trends in renewable energy." To accomplish this, the agent could use MCP to gather data – maybe pulling statistics from a market research database and recent news articles – then use a language model to **write a draft article**. Another agent (or the same agent with a tool) could generate accompanying graphics or charts. Yet another could even schedule the content to be posted online. This is not far-fetched – businesses are already looking into adopting autonomous agents for content generation tasks . The human's role shifts to reviewing the newsletter and giving high-level direction ("focus more on solar energy this month, please") rather than manually researching and writing everything from scratch.

In more purely creative endeavors like storytelling, design, or video game development, multiple AI agents could collaborate as specialized "artists." Because A2A and similar frameworks are **modality-agnostic (not limited to text)** – they can handle images, audio, and other data formats as well – you could have, say, an **AI screenwriting agent** and an **AI illustration agent** work together on a comic book. The writing agent drafts the story and dialogue, while the illustration agent creates panel artwork for each scene, communicating back and forth to keep the text and visuals consistent. Similarly, in game development, an AI level-design agent could generate a level's layout, then ping an AI music agent to compose background music fitting the scene. These agents can exchange information (like the mood or tempo needed for music) via A2A in a standardized way. The result is a coordinated creative output that would normally require a whole team of people.

Even when there's just one agent involved, the agentic approach means creative software can do much more on its own. For instance, Photoshop or another design tool might have an "AI design assistant" agent that you can instruct in natural language: *"Make the sky more vibrant and add some birds in the distance."* The agent understands your intent, uses image-editing tools to execute the changes (through MCP connectors to the Photoshop functions), maybe consults an image-generation model to create the birds, and then shows you the updated image. You didn't have to manually select filters or draw the birds; the agent did it for you, following your high-level guidance. This can

dramatically speed up the creative iteration process. Creators can spend more time on the conceptual and editorial decisions, and less on the tedious production steps.

In short, agentic AI in creative work means **more firepower for artists and writers**. It's like having a versatile creative partner who can generate drafts, mockups, or variations at your request. You can explore more ideas quickly – an agent can churn out 10 versions in the time it might take you to refine 1 – and then you pick or refine the best. The technology is still maturing, and human creativity and judgment remain crucial, but these AI agents are expanding the realm of possibility. A single individual, with the help of a swarm of creative AI agents, could produce content or art that would normally require an entire team or studio. This is a great example of how **agentic experiences shift what technology can do** – the user's experience becomes more about curation and direction, while the heavy lifting (whether it's writing thousands of words or painting a detailed image) is done by the AI behind the scenes.

Beyond: Business Operations and More

The examples above are just a glimpse. Many other sectors are poised to be transformed by agent-based AI systems. In finance, for instance, firms are experimenting with AI agents that can **manage investment portfolios** automatically – monitoring market data, making buy/sell decisions according to a strategy, and adjusting to news in real time, all without human micromanagement. Such an agent acts like a tireless junior fund manager, potentially handling personalized portfolios for many clients simultaneously . In manufacturing and supply chain management, agentic AI can coordinate complex logistics: one agent might monitor inventory levels across warehouses, another watches for supply disruptions (like a delay from a supplier), and others optimize shipping routes. Together, they can **adapt the supply chain on the fly** – rerouting shipments, finding alternate suppliers, adjusting factory production schedules – to respond to changes or risks . Some companies are indeed using AI for proactive supply chain risk management and dynamic scheduling to improve efficiency and resilience .

Even government services could see AI agents streamlining processes. We might have an agent that helps citizens fill out applications for benefits or permits by conversing with them and then automatically completing and filing the right forms. Or an agent that handles routine inquiries to city hall (e.g., "When is trash pickup for my neighborhood if Monday is a holiday?") by pulling

answers from databases and websites. Salesforce, for example, calls this vision "Agentforce" in CRM: humans and AI agents working together to drive better customer (or citizen) service outcomes . The goal is that **AI agents handle the grind** so humans can focus on tasks that truly require human creativity, empathy, and critical thinking.

In conclusion, the advent of protocols like A2A and MCP, combined with the power of modern AI models, is enabling a shift toward *agentic technology*. We're moving from a world where software is a static tool and the user must direct every step, to a world where software in the form of AI agents can **collaborate, take initiative, and deliver results** with much less hand-holding. This amplifies what people can accomplish across industries: students get personalized tutoring, doctors get instant analyses, customers get immediate solutions, developers write code faster, and creators multiply their output. For undergraduate students or anyone new to this field, the key ideas to remember are:

- AI agents are **autonomous helpers** that can make decisions and use tools.
- A2A is the **language for agents to talk to other agents**, allowing teamwork among AIs.
- MCP is the **universal plug** that lets agents connect to all sorts of external data and services easily.
- Agentic experience is a new design mindset where the user focuses on goals and outcomes, while the AI handles the steps to get there (flipping the traditional user experience on its head).

It's an exciting development in computing – one that could make technology feel more like collaborating with a smart partner than using a complicated tool. As with any powerful technology, it comes with challenges (like ensuring reliability, ethics, and trust), but understanding these concepts is the first step to navigating and harnessing the agentic future of AI.

1. Anthropic. (2024). *Introducing the Model Context Protocol*. Anthropic. https://www.anthropic.com/news/mcp
2. Google Developers. (2025). *Announcing the Agent-to-Agent (A2A) Protocol*. Google. https://developers.googleblog.com/2025/03/agent-to-agent-a2a-protocol.html

3. Dynatrace. (2023). *Agentic AI: MCP, A2A, and automation's future*. Dynatrace. https://www.dynatrace.com/news/blog/agentic-ai-and-mcp/
4. UX Magazine. (2025). *What to Know About Model Context Protocol (MCP)*. UX Magazine. https://uxmag.com/articles/what-to-know-about-model-context-protocol
5. KDnuggets. (2025). *A2A vs. MCP Explained Simply*. KDnuggets. https://www.kdnuggets.com/a2a-vs-mcp
6. Maeda, J. (2025). *Simplicity and Agentic Experience (AX)*. Personal blog. https://johnmaeda.com/agentic-experience
7. Klein, A. (2024). *The Agentic Era of UX*. Design Intelligence Weekly. https://www.diweekly.com/agentic-era-of-ux
8. Salesforce. (2023). *AI Agents in Education – Benefits & Use Cases*. Salesforce. https://www.salesforce.com/blog/ai-agents-in-education/
9. Converge AI. (2025). *Top 10 Agentic AI Examples and Use Cases*. Converge AI. https://www.converge.ai/blog/top-agentic-ai-examples