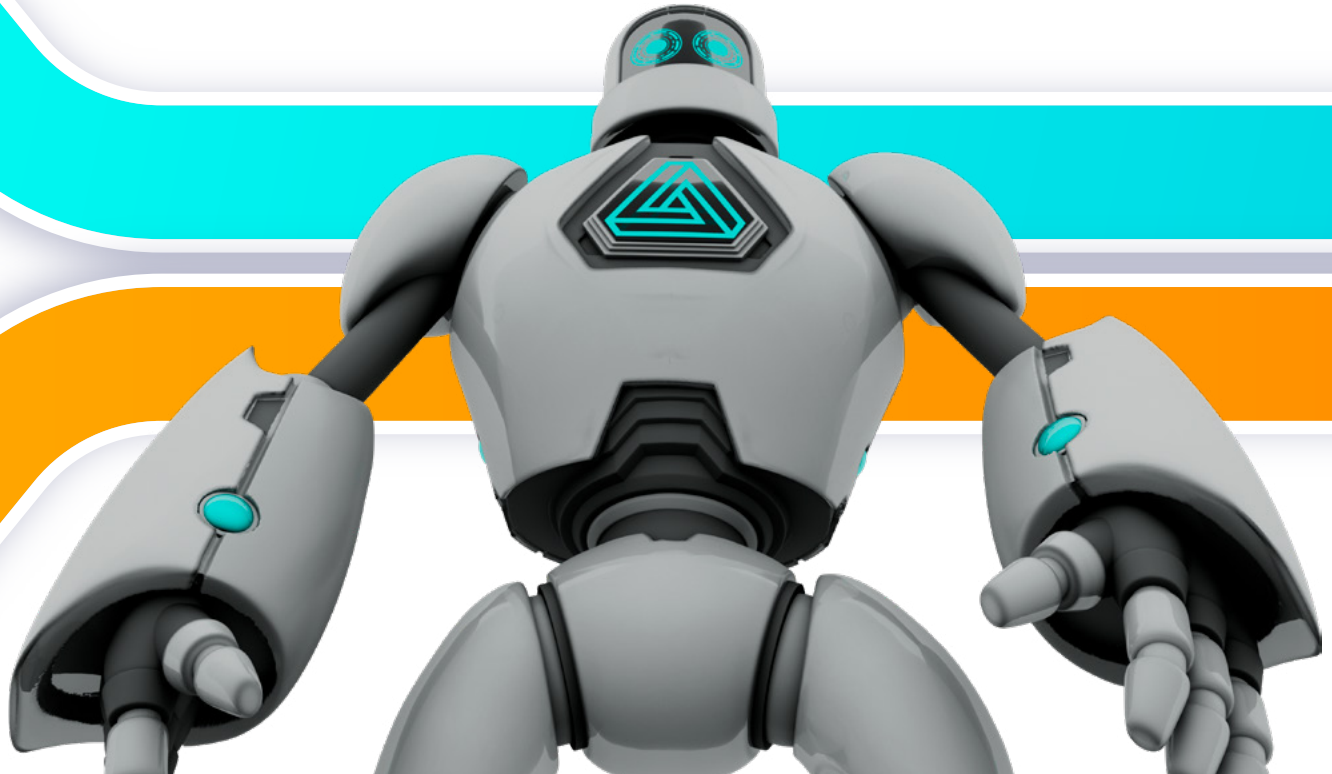
AUTOMOX

Agile + AI = IT Superpowers

How the fusion of Agile and AI delivers speed, reliability, and confidence to citizen developers in IT.



Agile is a project management and software development approach rooted in flexibility, collaboration, and achieving customer satisfaction.

Why Agile IT?

Change is the only constant and Agile gives IT the flexibility to adapt.

Before Agile

Back in the day, most IT projects utilized some variation of a waterfall development structure. This framework worked best when following a sequential process that didn't require updates or changes as tasks were predetermined at the outset of the development cycle.

But when was the last time you worked on a project sequentially where the requirements didn't undergo multiple updates? It just doesn't happen anymore.

Agile to the Rescue

Today, requirements change rapidly and unpredictably. Agile was built to embrace the bumps, hairpin turns, and surprises in the road.

While the Agile framework can trace its roots to the early seventies, it took its official shape in 2001 when software developers sought a framework focused on delivering high-quality work in small, manageable increments – or sprints.

The developers wanted to promote:

1. **Adaptive planning**
2. **Evolutionary development**
3. **Early delivery**
4. **Rapid, flexible responses to unforeseen changes**

The result of their collaboration is the **Agile Manifesto**.

History of the Agile Manifesto

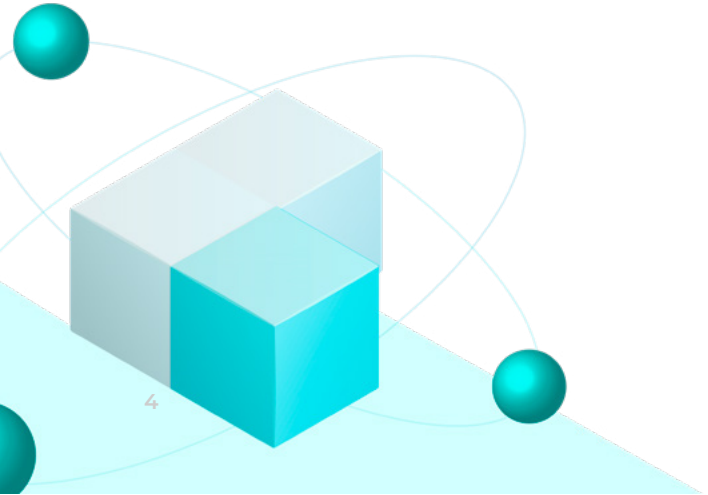
The Agile Manifesto

Prioritize individuals and interactions over processes and tools

Choose working software over comprehensive documentation

Collaborate with customers instead of just negotiating contracts

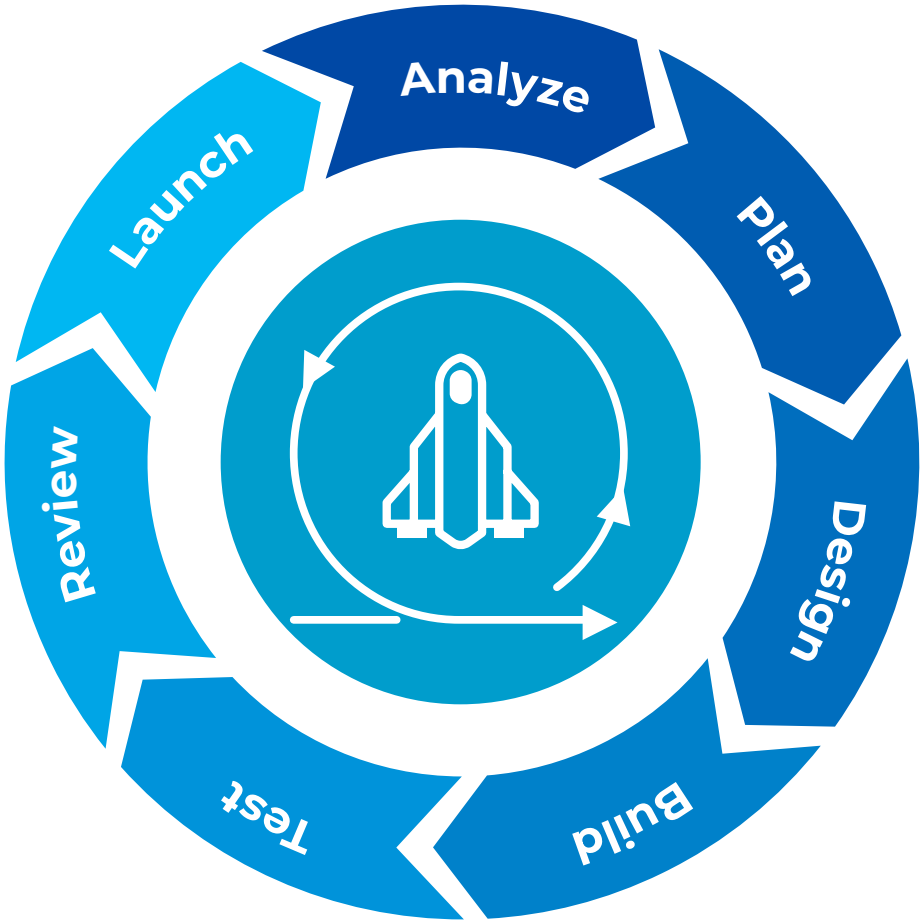
Respond to change in real time instead of just following the initial plan

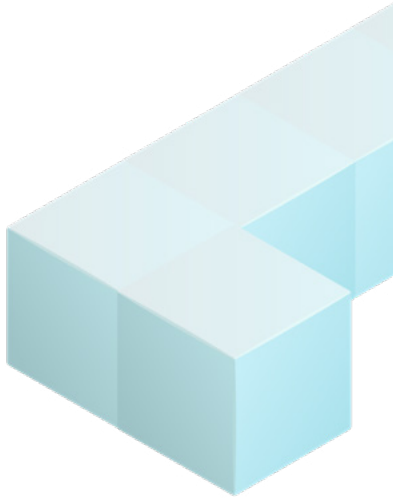


So, now you know Agile is a powerful framework to solve problems. But does it work well given today's IT automation boom? Um, yes! – especially when you see IT automation as a way to automate how you solve problems.

With the advent of low-code, no-code, and AI-assisted solutions, more people are automating IT tasks than ever before. These developers are often called 'citizen' developers.

Citizen developers don't always have a background in software development or a deep understanding of current development practices. So, if you're a citizen developer, understanding the Agile framework can help you level the playing field.





AI is a great partner to Agile.

AI democratizes rules-based automation at scale to multiply the impact of citizen developers.

Artificial intelligence large language models excel at code composition.

AI large language models (LLMs) are designed to leverage machine learning techniques to understand and follow human language patterns.

Most modern computer programming languages are defined as high-level programming languages – they're made up of common words so they're easily understood when read by humans.

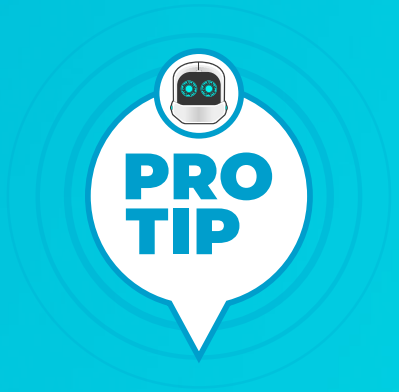
If you think about it, computer programming was created so humans could tell computers what to do. Today, LLMs mimic what a human would say when telling computers what to do. But both humans and AI LLMs thrive on specificity. The intersection of these technologies allows LLMs to excel at code composition.

Define the problem for AI in 4 Steps

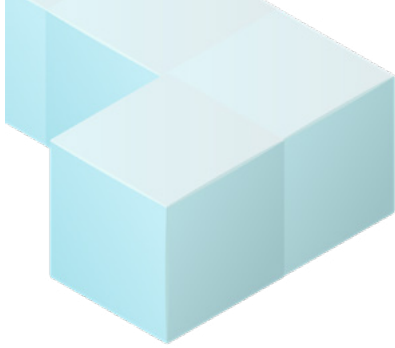
The first step in the problem-solving process is problem specification. An undefined problem has no solution. Without knowing what the problem is, how can you solve it?

Defining your problem can be simple.

1. **What's the problem?**
2. **How would you explain the problem to someone else?**
3. **What automations can help solve the problem?**
4. **What tools will you use to solve the problem?**



Create a document answering these questions or write them out with an old-school pencil.



Once you've defined the problem, you're ready to prompt AI. You'll want to be as clear as possible. How you communicate to AI tools matters. The quality of your input dictates the relevance of the output. This is **prompt engineering** – the strategic crafting of instructions or queries to feed to AI systems.

Defining a problem helps you start theorizing its solution. But if you're using AI to create code, how can you convey the problem to AI? It's a quandary.

So, let's first look at another field of study.

Accommodation Theory is a sociolinguistic concept that refers to the phenomenon of adjusting your speech or behavior to align with those with whom you're conversing. You probably do this every day and don't even notice. For instance, when discussing work with a coworker, you likely use complex, industry-specific jargon since your colleague is familiar with the topic.

When relaying the same information to a friend, you might simplify the language or use analogies that make sense within the context of your friendship.

Likewise, If you're explaining the same idea to a child, you oversimplify the idea so the kid can understand.

You make these accommodations so whoever you're talking with understands what you're saying.

What'd you do at work today?

I reviewed pull requests, programmed a new feature, got stakeholder clarification on a project, and surfaced tactics to improve our Agile process. I also worked on a few AI automations to streamline workflow.

I did some coding and attended a project update meeting. I also integrated some AI automation into a task, which was exciting.

I played with computers all day like how you play with your toys!

What makes an effective prompt?

Prompt engineering isn't just about formulating effective prompts. It's also about understanding an AI's interpretive mechanisms. Properly engineered prompts lead to more accurate, contextually relevant responses, enhancing AI capabilities and creating efficient processes and automations.

Context is key – the only information the AI has about your problem is what you give it.

Remember your first book report in grade school? Were you asked to use the 5 Ws and an H framework to convey the important points of the story?

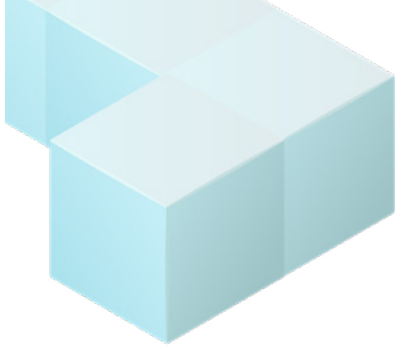
If you don't remember the framework, think of the structure of a party invitation: *Who, What, When, Where, Why, and How*

The same outline comes in handy when working on AI prompt engineering.

5W+H Prompt Checklist

The 5 Ws and an H of Prompt Engineering

- **Who:** Who does this problem affect?
- **What:** What is the problem? Use the Accommodation Theory ‘explain to others’ approach.
- **When:** When should each step occur within your script? Think: First this, then that.
- **Where:** Where does data go – and where does it flow from? For example, is it using an API, a CSV file, a text document?
- **Why:** Think of a child asking you why grass is green. Unsatisfied with vague responses the child asks, ‘Why? Why? Why?’ begging for more detail. Be specific! This will help inform your ‘how.’
- **How:** How do you want AI to approach the problem? What techniques, libraries, frameworks, software, or programming languages should it use?



Here's an example:

Who: This script is going to be used by data engineers not familiar with PowerShell.

What: Write a script to pull data from a SQL database and export it to a CSV file.

When: The script should ask the user for login credentials, then connect to a SQL database. The script should then ask the user if they want a custom SQL query, or standard SQL query, by asking for a 1 or 0. If the connection is not successful, an error should show.

Where: The data is being pulled from a remote SQL database by URL.Once it pulls the information from the database, it should write it to a CSV file.

Why: There should be an area within the code for a SQL query. This can be hard coded, and does not need to be presented to the user. A variable should be hard-coded for the SQL database connection, as the database and table will not change. Leave a section in the code for the SQL query. If the user wants a custom SQL query, it should allow the user to type in a SQL query, and override the default query.

How: The script should be written in PowerShell.

Prompt engineering and problem definition are super important when communicating with AI. By using the 5 Ws and an H framework and adhering to the Agile framework, developers can create more precise and effective AI prompts.

In the end, LLMs offer a transformative approach to code composition, facilitating efficiency and innovation within the development process. Combined with the Agile's emphasis on adaptability and customer satisfaction, these practices form an ideal structure to help you navigate a rapidly changing, ever-evolving tech landscape.