GitHub Copilot Memory: From Black Box to Memory Bank

Dr. Matthias Liebeck

Agentic Coding Summit #1

27.05.2026

About Dr. Matthias Liebeck

- Since 2019: Senior Solution Architect at anyMOTION GmbH, Düsseldorf
- .NET / C# / Azure, GitHub Copilot
- 2009 – 2015: Studies in Computer Science and Mathematics
- 2018: PhD in Computer Science / Artificial Intelligence, focus on Natural Language Processing
- Organizer: Azure Düsseldorf Meetup
- Community Speaker: GitHub Copilot, MCP, Azure OpenAI
- Newsletter: ghcp.liebeck.io

Azure Meetup Düsseldorf

Microsoft Certified: Azure Solutions Architect Expert

Microsoft Certified: DevOps Engineer Expert

Microsoft Certified: Azure Developer Associate

Microsoft Certified: Azure Administrator Associate

Microsoft Certified: Azure Data Scientist Associate

GitHub Copilot Certification Program

GitHub Copilot Newsletter

The Problem

Every Coding Session Starts From Scratch

- You explain your architecture … again
- You explain part of your business logic … again
- Knowledge built up during a session does not transfer to new sessions
- Good conversations feel like living documentation, but they disappear
- Context that took 20 minutes to build is lost in one click if a session is broken
- I start re-using old chat threads because they feel "warmed up"

GitHub Promises a Solution

GitHub Copilot Memory (Public Preview since January 2026)

- GitHub's promise: "Copilot Memory allows Copilot to build and retain a persistent, repository-level understanding of your codebase so you spend less time reexplaining context and more time shipping code.“[1]
- Backed by A/B tests showing 7% higher PR merge rates [2]
- Code reviews have 2% more positive feedback (77% vs 75%) [2]
- But almost no real-world experience reports from developers
- **GitHub's own feedback thread? Barely any concrete answers.**

👉 I decided to take a closer look and test it out

[1] https://github.blog/changelog/2026-03-04-copilot-memory-now-on-by-default-for-pro-and-pro-users-in-public-preview/
[2] https://github.blog/ai-and-ml/github-copilot/building-an-agentic-memory-system-for-github-copilot/

1. **GitHub Copilot Memory**: What it is and how it works
2. My experience using GitHub Copilot Memory
3. **The Memory Bank pattern**: What I use instead

GitHub Copilot Memory

- „Copilot Memory allows Copilot to learn about your codebase, helping Copilot cloud agent, Copilot code review, and Copilot CLI to work more effectively in a repository.” [2]
- Copilot Memory [2] :
  - “Reduces the burden of repeatedly providing the same details in your prompts.”
  - “Reduces the need for regular, manual maintenance of custom instruction files.”
  - “By building and maintaining a persistent, repository-level memory, Copilot develops its own knowledge of your codebase, adapts to your coding requirements, and increases the value it can deliver over time.”

🧠 💡 **Sounds great, right?**

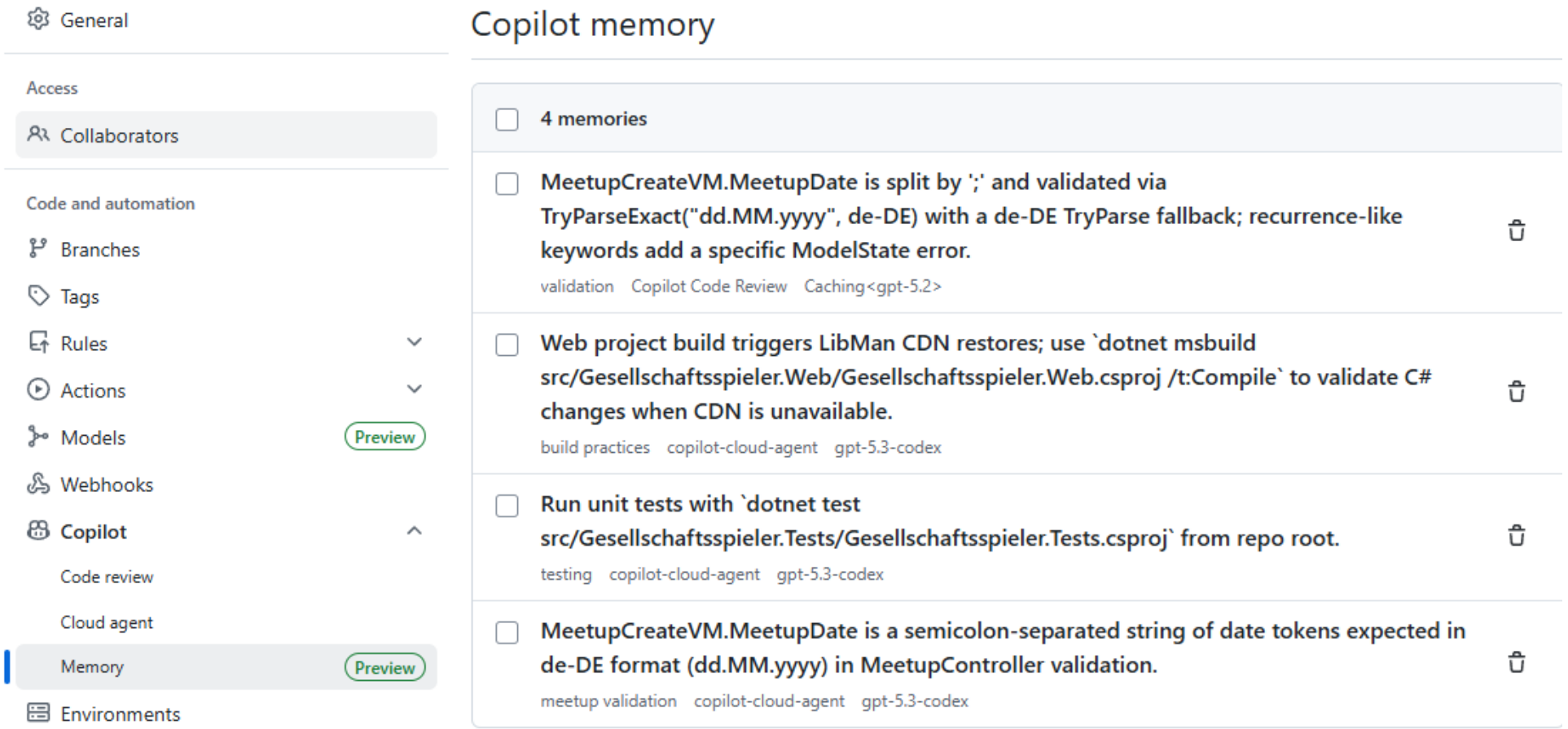
[2] https://docs.github.com/en/copilot/how-tos/use-copilot-agents/copilot-memory

GitHub Copilot Memory

- On by default for Pro/Pro+ since March 2026
- Cloud-hosted, repository-scoped
- Created automatically by Coding Agent, Code Review, and Copilot CLI
- NOT created by normal chat in VS Code or Visual Studio
- Memories are shared across agents
- Auto-expires after 28 days

- An agent works on your repo (e.g., code review reviews a PR)
- It discovers a pattern (e.g., "API versions must be synced in 3 files")
- A memory is stored with a subject, a fact, citations, and a reason
- Next session: memory is injected into the agent's prompt
- Agent validates citations against current code before using it
- If code contradicts the memory, the agent stores a corrected version

- 4 memories after several pull requests via code review



- All of them: highly specific implementation details
- None of them: architectural insights or coding conventions

What I hoped Copilot would learn:

- This project uses German date formats (dd.MM.yyyy)
- We use xUnit with FluentAssertions for testing
- Input validation follows a specific pattern across all controllers

What Copilot actually learned:

- How one specific field in one specific ViewModel is parsed

🤔 The memories are technically correct.

😟 But they are too granular to be useful for future tasks

What you CAN do:

- View memories: Repository Settings → Copilot → Memory
- Delete individual memories
- Turn the feature on/off

What you CAN'T do:

- Create memories manually
- Edit existing memories
- Choose what gets remembered
- See the code citations that each memory references
- Use it from normal chat (only Coding Agent, Code Review, CLI)
- The UX shows a flat list. No context, no citations, no way to understand why a memory was created.

- **28-day expiration**: If you work on a project every 2 months, all memories are gone when you come back.
- **No community experience reports**: GitHub's own feedback thread asks: "Did Copilot correctly learn and reuse a convention?" Almost no one has answered with concrete examples.

GitHub Copilot Memory is:

- An interesting idea with real engineering behind it
- Backed by statistically significant A/B test results
- Still in public preview

But in practice:

- It's a black box you can't control
- The memories it creates are tightly scoped
- 28 days is too short for many workflows
- There is no customization from the user's perspective

GitHub states "Agentic memory will be extended to other parts of Copilot, and for personal and organizational scopes, in future releases." [1]

We're not there yet. Let's see how the feature improves over time 🚀

[1] https://docs.github.com/en/copilot/concepts/agents/copilot-memory

What If You Could Control Your AI’s Memory?

What if your AI's memory was:

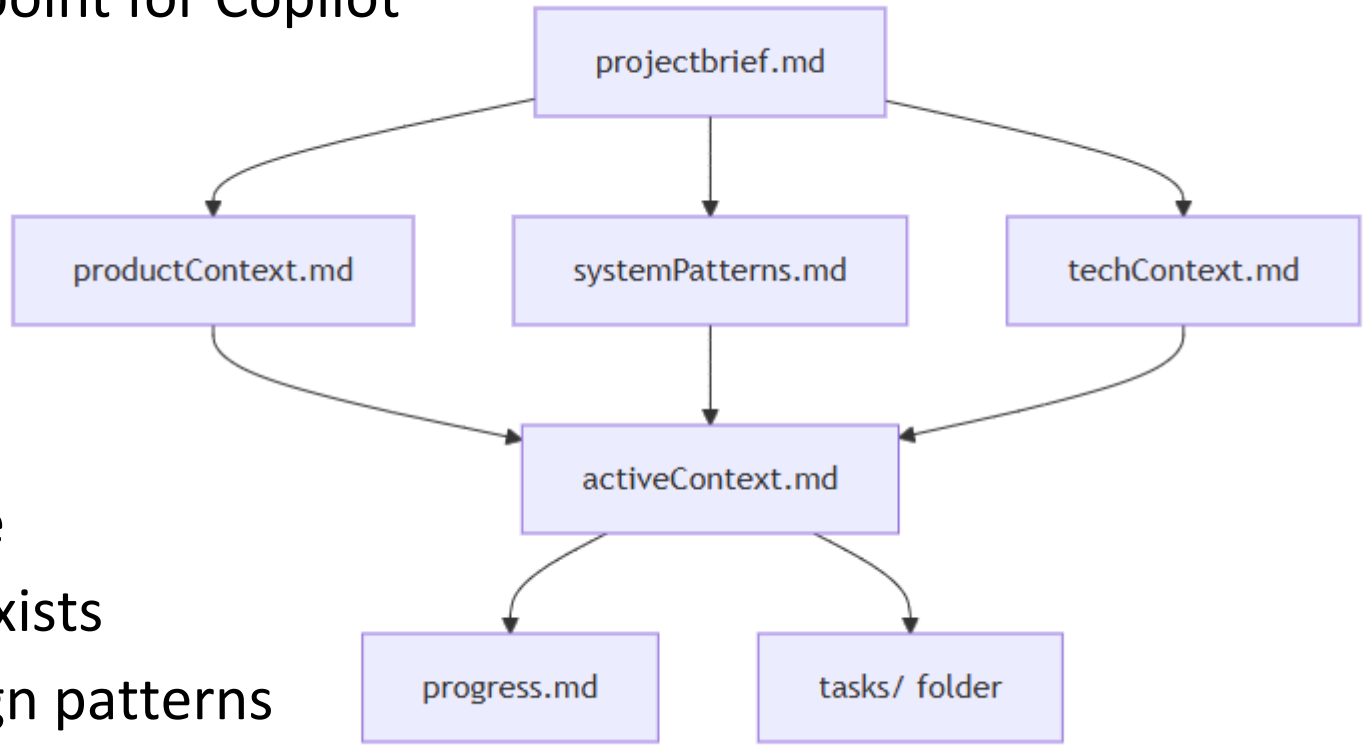
- Editable: you decide what it remembers
- Git-tracked: version-controlled, reviewable in PRs
- Shared: the whole team benefits immediately
- Permanent: no 28-day expiration
- Transparent: Markdown files you can read and understand

Memory Bank Pattern = A folder of Markdown files in your repository that serves as your AI's long-term memory.

- Your AI reads them at the start of every task
- Your AI updates them after completing work
- Works with VS Code Agent Mode and Copilot CLI (via copilot-instructions.md)
- Works with Claude Code (via CLAUDE.md)
- The pattern is tool-agnostic: it's just Markdown and Git

Available as a template in github/awesome-copilot (32k+ stars)

.github/copilot-instructions.md ← entry point for Copilot



projectbrief.md ← goals, scope, audience
productContext.md ← why this project exists
systemPatterns.md ← architecture, design patterns
techContext.md ← tech stack, constraints
activeContext.md ← current focus, open issues
progress.md ← completed work, next steps
/<feature>/ ← per-feature context

https://github.com/github/awesome-copilot/blob/main/instructions/memory-bank.instructions.md

copilot-instructions.md automatically detected by VS Code, Visual Studio, and GitHub.com

Key rules to include:

- 1. Read ALL files in /memory-bank/ before starting any work
- 2. After completing work, update activeContext.md and progress.md
- 3. When you discover a new pattern, add it to systemPatterns.md
- 4. Document architecture decisions
- 5. The Memory Bank is your only link to previous sessions

1. **Use the awesome-copilot template**
   - Copy memory-bank.instructions.md from github.com/github/awesome-copilot into your **.github/copilot-instructions.md** file
   - Create the /memory-bank/ folder
2. Let Copilot bootstrap by **prompting "Analyze this project and populate the Memory Bank.**“ in Agent Mode

Summary

Comparison of both approaches:

	Copilot Memory	Memory Bank
Storage	GitHub Cloud	Git repository
Created by	AI automatically	**Developer + AI together**
Editable	view & delete only	**full control**
shared with team	yes, via repository access	yes, via repository access
auto expiration	28 days	**never**
transparency	black box	**markdown files**
customization	not possible	**full**

Key takeaways:

1. **GitHub Copilot Memory exists, but it's a black box you can't control**. In my experience, the memories it creates are too granular to be useful.
2. **The Memory Bank pattern gives you control**. Git-tracked Markdown files that your AI reads and updates.
3. **The pattern is tool-agnostic**. Works with Copilot, Claude Code, and any AI tool that reads files. **Try it out** 🧠

Thank You!

LinkedIn

GitHub Copilot Newsletter