MARTINELLI
Keep IT simple

Spec-driven Development Workshop

https://martinelli.ch/sdd/

Simon Martinelli

About Me

- 30 years in software engineering
- 25 years with Java
- Self-employed since 2009
- Teaching at two Universities
- Co-lead Berne, JUG Switzerland

Java Champions

vaadin Champion

Oracle ACE Pro

Introduction

Quotes by Martin Fowler

"I think the appearance of LLMs will change software development to a similar degree as the change from **assembler** to the first **high-level programming languages**.

The further development of languages and frameworks increased our abstraction level and productivity, but didn't have that kind of impact on the nature of programming.

"LLMs are making that degree of impact like high-level languages had versus the assembler.

The distinction is that LLMs are not just **raising the level of abstraction**, but also forcing us to consider what it means to program with **non-deterministic tools**."

What is Vibe Coding?

I just see things, say things, run things, and copy-paste things, and it mostly works.
- Andrej Karpathy, OpenAI Co-founder, former Tesla AI lead, February 2025

- A **programming approach** where you
  - Use natural language to describe what you want
  - Let AI (LLMs) write all the code
  - Accept code without full understanding
  - Focus on the goal, not the code itself

Is Vibe Coding Any Good?

Good

- Prototypes
- Hobby projects
- Quick experiments
- Non-programmers building simple tools

Risky

- Production code
- Security-critical apps
- Complex systems
- Business applications

Tools, Tools, Tools

- **Chats**
  - Claude.ai
  - ChatGPT
  - Gemini
  - Mistral (EU)
- **IDE integrated chats**
  - JetBrains AI Assistant
  - Copilot
  - Devoxx Genie
- **Spec-driven Development**
  - Amazon Kiro
  - GitHub Spec-Kit
  - Tessl
- **Agentic AI**
  - Windsurf
  - Copilot Coding Agent
  - JetBrains Junie
  - Claude Code
  - Codex
  - Gemini CLI
  - Mistral Vibe CLI
- **Online AI Development**
  - Replit
  - Bolt
  - Lovable

Caution: AI Is Not a Compiler

- **AI** code generation **isn't a compiler** - it's an assistant!
- Many developers expect AI to work like a compiler:
  - Precise input → perfect output
  - But that's not how it works

- AI is **not deterministic** and makes mistakes

- **You are responsible for the output!**

AI Native Development

- **Spec-Centric Development**
  - Clear intent/specs guide AI to generate meaningful code
- **Context-Aware Development**
  - AI agents understand full codebase context
- **Agent Experience (AX)**
  - Autonomous AI tasks (bug fixes, PRs, tests) enhancing developer throughput

https://ainativedev.io/

Spec-driven Development

- **Start with a spec**, not code
- Spec is the contract and **source of truth** for tools and AI
- Leads to less guesswork, fewer surprises, **better code**

- **Process**
  - AI Unified Process: https://aiup.dev

- **Tools**
  - Amazon Kiro: https://kiro.dev
  - GitHub Spec Kit: https://github.com/github/spec-kit

Claude Code

- **Terminal-based** AI code assistant
- Integrated in Claude App and https://claude.ai
- **Plugins** for JetBrains tools and Visual Studio Code
- GitHub and GitLab **integration**
- **Features**
  - Skills
  - MCP
  - Subagents

- Documentation: https://docs.claude.com/en/home

Claude Code Security Model

- Claude Code provides built-in sandboxing, but it is **not full isolation**
- **Key mechanisms**
  - Filesystem restrictions (limit accessible folders)
  - Network controls (allow / deny outbound access)
  - Permission system (asks before critical actions)
  - Configurable policies per project
- **Important**
  - Sandbox can be relaxed or bypassed with approval
  - It is a soft boundary, not a hard security layer
- **Takeaway**
  - Good default protection, but not enough for untrusted execution

From Sandbox to Real Isolation

- Stronger security requires **external isolation**
- **Option 1: Docker Container**
  - Run agent inside container
  - Limit volumes and secrets
  - Control network access
- **Option 2: Docker Sandbox (MicroVM)**
  - Full isolation from host
  - Separate filesystem and runtime
  - No access to local machine
  - Environment can be reset anytime
- **Benefits**
  - Safe autonomous agent execution
  - Prevents data leaks to host system
  - Enables multi-agent setups securely

Hotel Reservation System

Vision EN

- We are the proud owners of a small, ten-room hotel.

- We need a system to manage reservations, guest check-in/check-out, and room cleaning.

- Actors
  - Guest (Online Booking)
  - Receptionist (On-site Management)
  - Manager (Reporting and Administration)
  - Housekeeping (Room Service)

Vision DE

- Wir sind stolze Besitzer eines kleinen Hotels mit zehn Zimmern

- Wir benötigen ein System zur Verwaltung von Reservierungen, Gästen mit Check-in/Check-out und Zimmerreinigung erstellen.

- Akteure
  - Gast (Online-Buchung)
  - Rezeptionist (Vor-Ort-Verwaltung)
  - Manager (Berichte und Verwaltung)
  - Housekeeping (Zimmerservice)

0. Set the Stage

1. **Clone the project**
   https://github.com/simasch/hrs
   - main: PostgreSQL with Testcontainers (requires Docker)
   - h2: H2 (no Docker required)

2. **Install the Claude Code Plugin**
   http://github.com/martinellich/aiup-marketplace

3. Run **`./mvnw spring-boot:test-run`**

1. From Vision to Requirements

1. Create the vision markdown file **`docs/vision.md`**

2. Run **`/requirements`**

3. Review **docs/requirements.md** and make adjustments if needed

Example Requirements EN

ID	Description	Prio	Status	Size
FR-001	Guests can search online for available rooms	High	Open	M
FR-002	Guests can reserve rooms online	High	Open	L
FR-003	The system automatically generates booking confirmation emails	High	Open	S
FR-004	Receptionists can check in walk-in guests	High	Open	M
FR-005	Receptionists can check out guests and generate invoices	High	Open	L
FR-006	The system manages room status (available, occupied, cleaning, maintenance)	High	Open	M
FR-007	Guests can cancel reservations up to 24 hours before arrival	Medium	Open	S
FR-008	Managers can generate occupancy reports	Medium	Open	XL
FR-009	The system sends reminder emails before arrival	Low	Open	S
FR-010	Housekeeping can update room status	High	Open	S

Example Requirements DE

ID	Beschreibung	Prio	Status	Grösse
FR-001	Gäste können online nach verfügbaren Zimmern suchen	Hoch	Offen	M
FR-002	Gäste können Zimmer online reservieren	Hoch	Offen	L
FR-003	System generiert automatisch Buchungsbestätigung per E-Mail	Hoch	Offen	S
FR-004	Rezeptionist kann Walk-in Gäste einchecken	Hoch	Offen	M
FR-005	Rezeptionist kann Gäste auschecken und Rechnung erstellen	Hoch	Offen	L
FR-006	System verwaltet Zimmerstatus (frei, belegt, reinigung, wartung)	Hoch	Offen	M
FR-007	Gäste können Reservierungen bis 24h vor Ankunft stornieren	Mittel	Offen	S
FR-008	Manager kann Berichte über Auslastung generieren	Mittel	Offen	XL
FR-009	System versendet Erinnerungs-E-Mails vor Ankunft	Niedrig	Offen	S
FR-010	Housekeeping kann Zimmerstatus aktualisieren	Hoch	Offen	S

2. Derive Entity Model

1. Run **`/entity_model`**

2. Review **`docs/entity_model.md`** and make adjustments if needed

3. Create Use Cases Diagram

1. Run **`/use_case_diagram`**

2. Review **`docs/use_cases.puml`** and make adjustments if needed

4. Create Database Migrations

1. Run **`/aiup-vaadin-jooq:4_database_migration`**

2. Review the files generated in **`src/main/resources/db/migrations`** and make adjustments if needed

3. Run **`./mvnw compile`**

4. Use Case Specification

1. Run **`/use_case_spec UC-###`**
   ### = the number of the use case you want to generate the specification for

2. Review **`docs/use_cases/UC-###`** and make adjustments if needed

5. Create Database Migrations

1. Run **`/flyway_migration`**

2. Review the files generated in **`src/main/resources/db/migration`** and make adjustments if needed

3. Run **`./mvnw compile`**

6. Generate Code

1. Run **`/implement UC-###`**
   ### = the number of the use case you want to generate the code for

2. Run the application in the IDE or by running
   `./mvnw spring-boot:test-run`

3. Review the code in **`src/main/java`** and make adjustments if needed

7. Integration Test

1. Run **`/karibu_test UC-###`**
   ### = the number of the use case you want to generate the code for

2. Review the generated test in **`src/main/test`** and make adjustments if needed

3. Run the tests **`./mvnw test`**

7. E2E Test

1. Run **`/playwright_test UC-###`**
   ### = the number of the use case you want to generate the code for

2. Review the generated *IT class in **`src/main/test`** and make adjustments if needed

3. Run the tests **`./mvnw verify`**

Risks and Recommendations

Technical Risks

- **Prompt injection attacks**
  - Malicious input manipulates the agent’s behavior
- **Dependency confusion**
  - Agents may import unsafe or unnecessary libraries
- **Unsafe generated code**
  - AI may produce vulnerable or insecure implementations
- **Over-permissive configurations**
  - Too many permissions increase attack surface

Quality Risks

- **Incorrect or insecure code**
  - Agents often produce plausible but wrong or unsafe code
- **Silent logic errors**
  - Code compiles and passes basic tests but behaves incorrectly
- **Architecture drift**
  - Generated code might violate architectural rules or conventions
- **Poor test coverage**
  - Agents may generate minimal or trivial tests
- **Performance issues**
  - Agents don’t optimize unless explicitly prompted

Security and Compliance Risks

- **Data leakage**
  - Source code, data, credentials, or business logic sent to external APIs (LLM, MCP etc)
- **Insecure code patterns**
  - Hardcoded secrets, missing input validation, or unsafe serialization
- **Supply chain contamination**
  - Generated code might copy licensed or unknown-source snippets

Legal and IP Risks

- **Copyright uncertainty**
  - You may not fully own the generated code
- **License incompatibility**
  - Agents may generate code under a restrictive license
- **Auditability**
  - Hard to trace which parts of code were human vs. AI generated

Organizational and Process Risks

- **Skill erosion**
  - Developers stop understanding what they use
- **Developers stop understanding what they use**
  - Code grows faster than governance processes
- **False confidence**
  - Teams assumes “AI created it, so it’s fine”
- **Mismatch with documentation**
  - Code changes faster than specs or architecture diagrams

Mitigation Strategies

Area	Action
Governance	Constantly **review everything** that is generated
Architecture and Security	**Scaffold** the application **yourself** Use tools like ArchUnit to **enforce structure** automatically **Run static analysis** (SonarQube, OWASP Dependency Check)
Data protection	**Don’t use production data** while at development time Prefer on-prem or self-hosted models for sensitive code
Testing	Create **test case examples**
Training	**Educate developers** on AI limits and prompt engineering

Conclusion

- **Specs help** to reduce non-determinism
- AI feels like a teammate but **it's just a tool**
- It can accelerate development, but **you must**:
  - **Review**, **understand**, and **test** the output
  - **Know** your **architecture** and **domain**

Thank you!

- **Web**
  martinelli.ch
- **E-Mail**
  simon@martinelli.ch
- **Bluesky**
  @martinelli.ch
- **X/Twitter**
  @simas_ch
- **LinkedIn**
  https://linkedin.com/in/simonmartinelli