

AI-Assisted MBSE for Agentic Software Synthesis

Better Specs Yield Better Outcomes

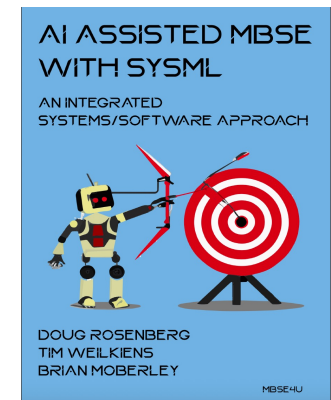
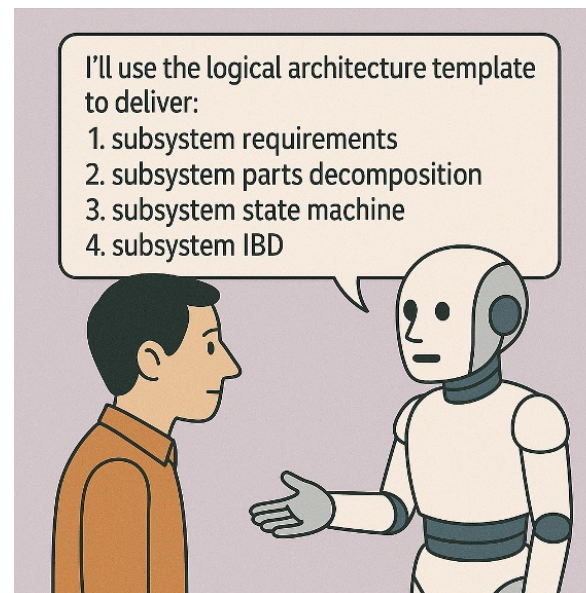
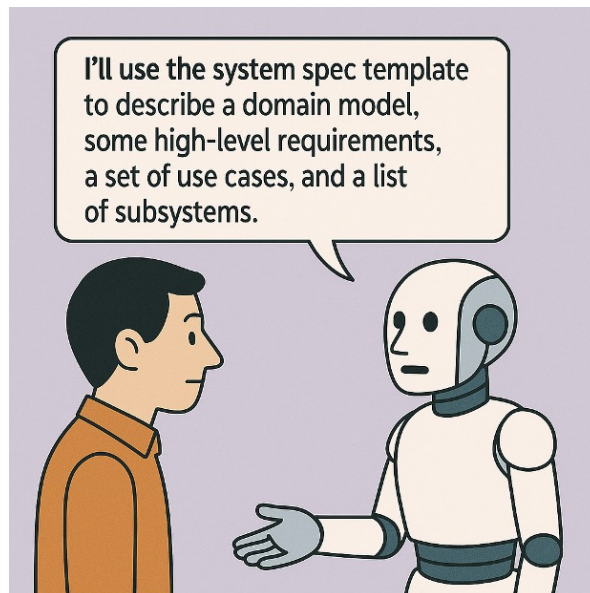
Doug Rosenberg, Parallel Agile, Inc. and Caltech CTME

Brian Moberley, Innovative Defense Technologies

Dr. Rick Hefner, Caltech CTME

We've Been Using AI to Generate SysML Specs for a couple of years

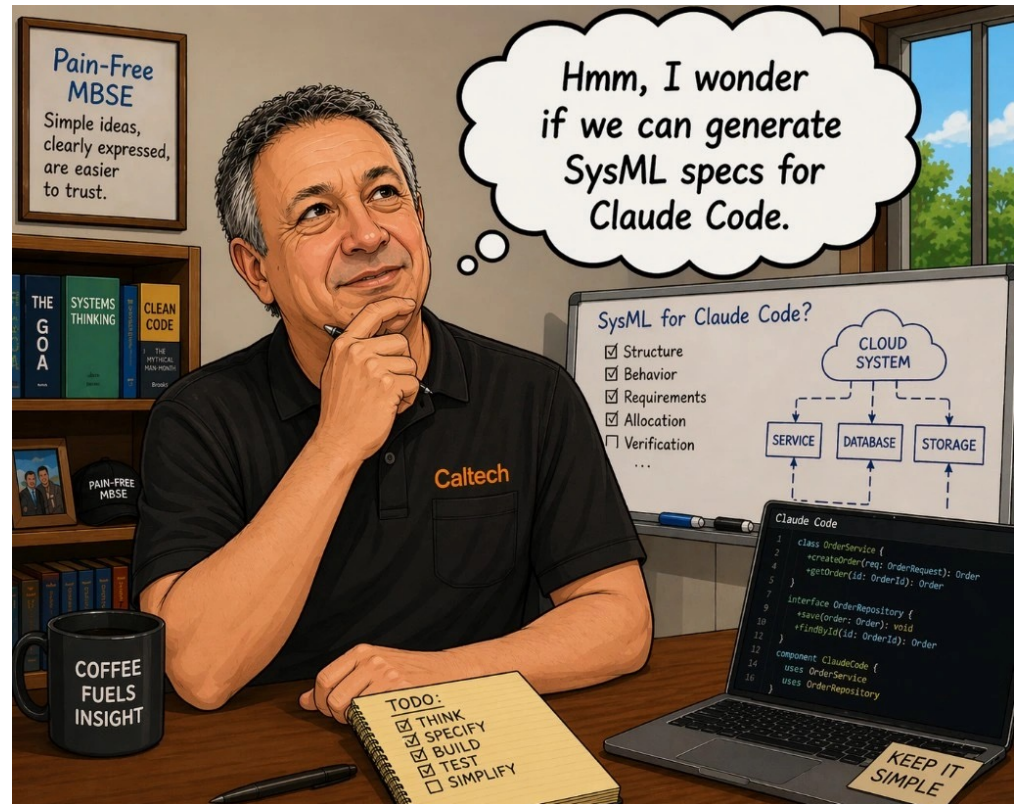
Template-driven SysML generation evolved into agentic software synthesis



Can SysML Specs Improve AI Coding?

For months, Brian had been encouraging me to try vibe coding with Claude Code

Finally, one day...



AIM Layered Specification Structure

SysML specs commonly contain

- Requirements
- Use Cases
- State Machines
- Test Cases

These are all useful to Code Generating Agents

We designed an AIM template to produce good specs

Test coverage is improved through specifications



Why Vibe Coding Often Struggles

Not too different from human programming...

High token cost from repeated rebuilds

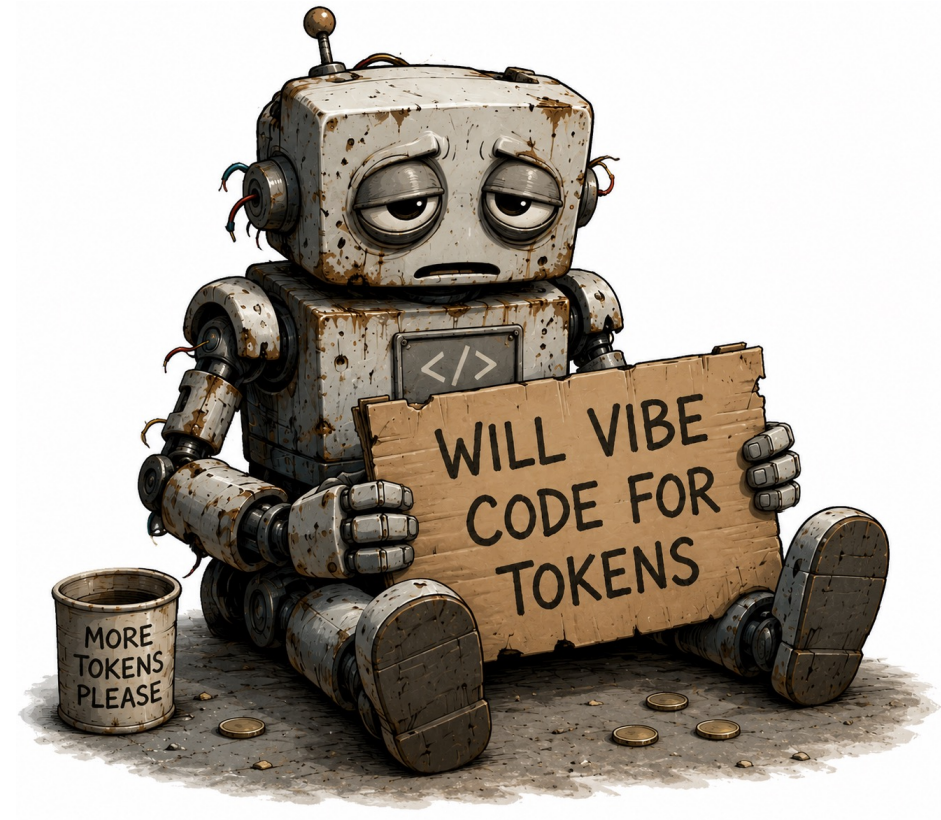
Re-coding is expensive, even with AI Agents

Vibe-coded apps are difficult to test and trust

If you have no spec, what do you test against?

AI loses context (AI Amnesia)

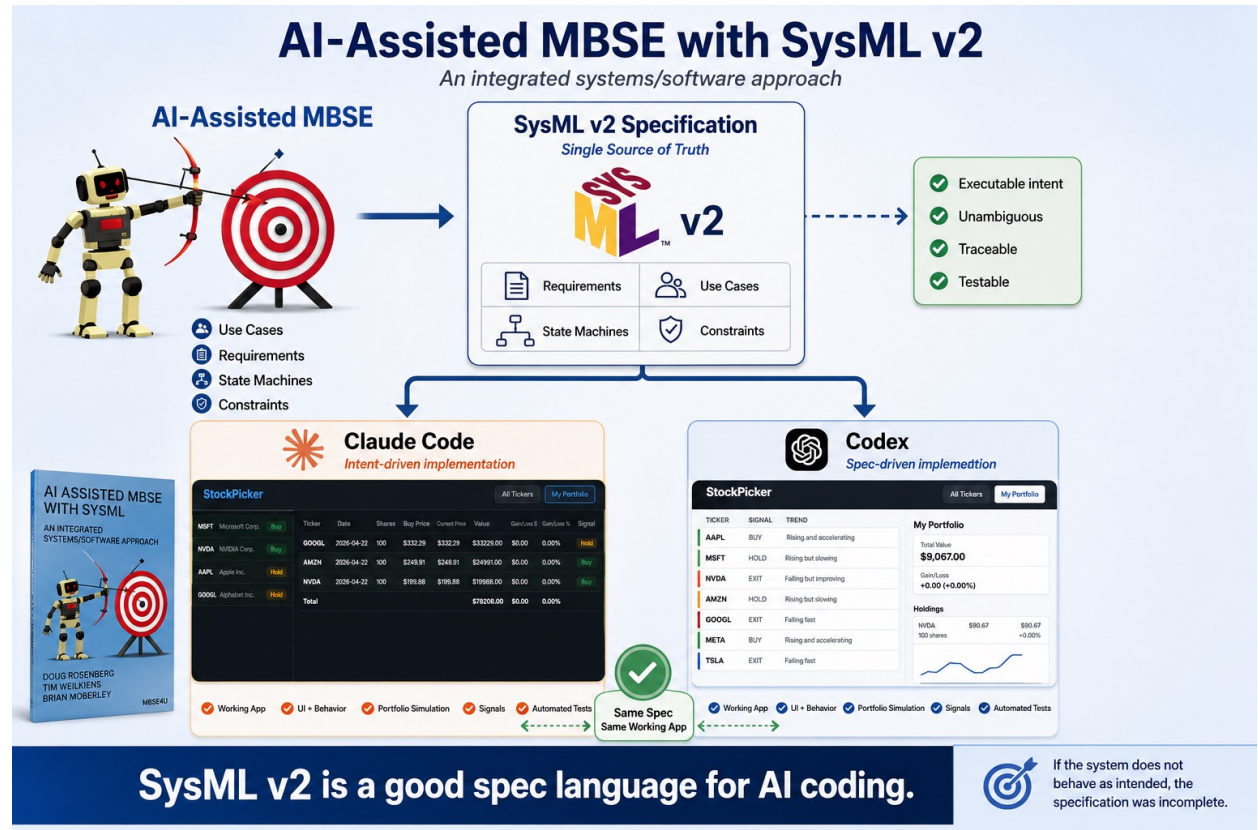
*It forgets what it had learned in the previous sessions
and un-does things that were previously working*



SysML v2 is a good language for code gen specs

We tried SysML v2 as a spec language for AI coding agents

It worked well for both Claude Code and for Codex



Lower Token Cost, Faster Progress

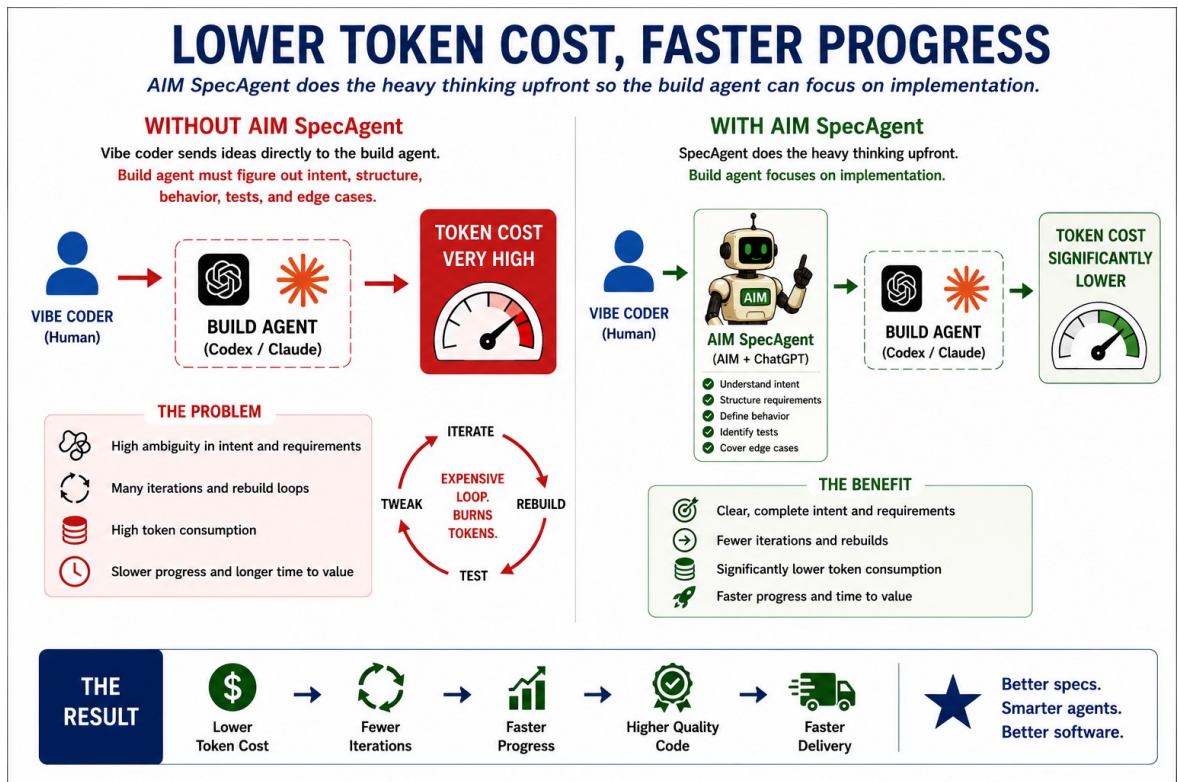
Iterating on the spec can be done without Claude Code or Codex (e.g., ChatGPT)

The spec iterations don't burn tokens

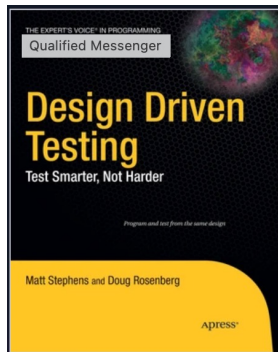
The coding agent re-writes the code a lot less than it would without the spec

Vibe coding remains very valuable, in particular, for tuning the user interface

Spec is re-generated after each coding session



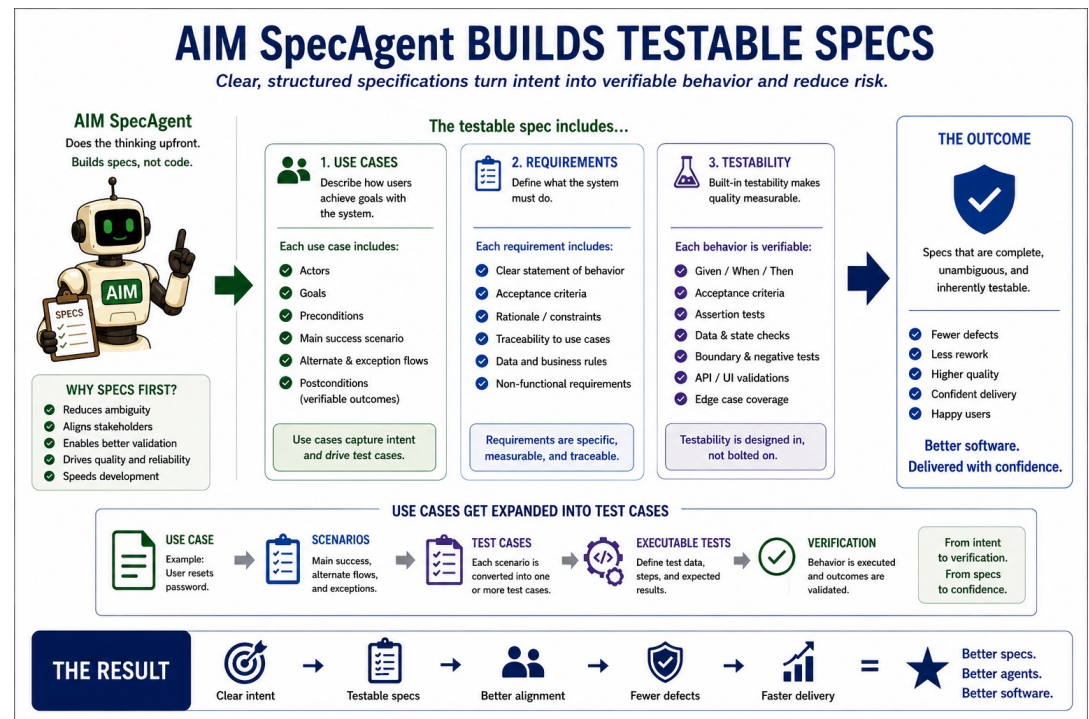
AIM Spec Agent Builds Testable Specs



We adopted test strategies from the 2003 book I wrote with Matt Stephens, notably:

- Requirement Testing
- Use Case Thread Expansion

Testing against use case threads greatly improves test coverage



The StockPicker Experiment Begins

I had an idea back when I was in college about picking stocks that were all going up

I had never used Codex and decided to try the spec that Brian and I were using with Claude Code

It worked!!!



Claude got good results from the SysML spec

Brian and I worked together with Claude

StockPicker built in 30 minutes and ran the first time

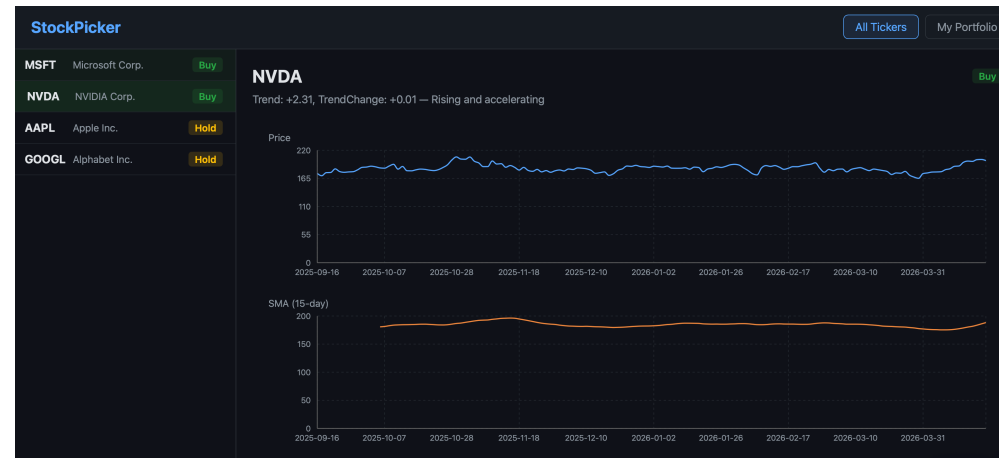
First build had a few bugs

*We had also forgotten to specify “web app” and
online data source*

*We added Design Driven Testing to the spec and built
again*

Number of tests run increased around 4X

Next build was much more reliable



StockPicker		All Tickers My Portfolio							
Ticker	Date	Shares	Buy Price	Current Price	Value	Gain/Loss \$	Gain/Loss %	Signal	
GOOGL	2026-04-22	100	\$332.29	\$332.29	\$33229.00	\$0.00	0.00%	Hold	
AMZN	2026-04-22	100	\$249.91	\$249.91	\$24991.00	\$0.00	0.00%	Buy	
NVDA	2026-04-22	100	\$199.88	\$199.88	\$19988.00	\$0.00	0.00%	Buy	
Total					\$78208.00	\$0.00	0.00%		

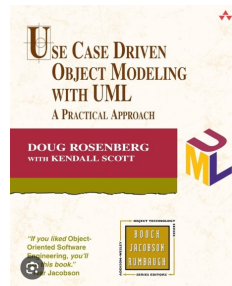
Three-Agent Collaboration

Human + AIM Spec Agent + Build Agent

I worked with both the Spec Agent (ChatGPT 5.5) and the Build Agent (Codex)

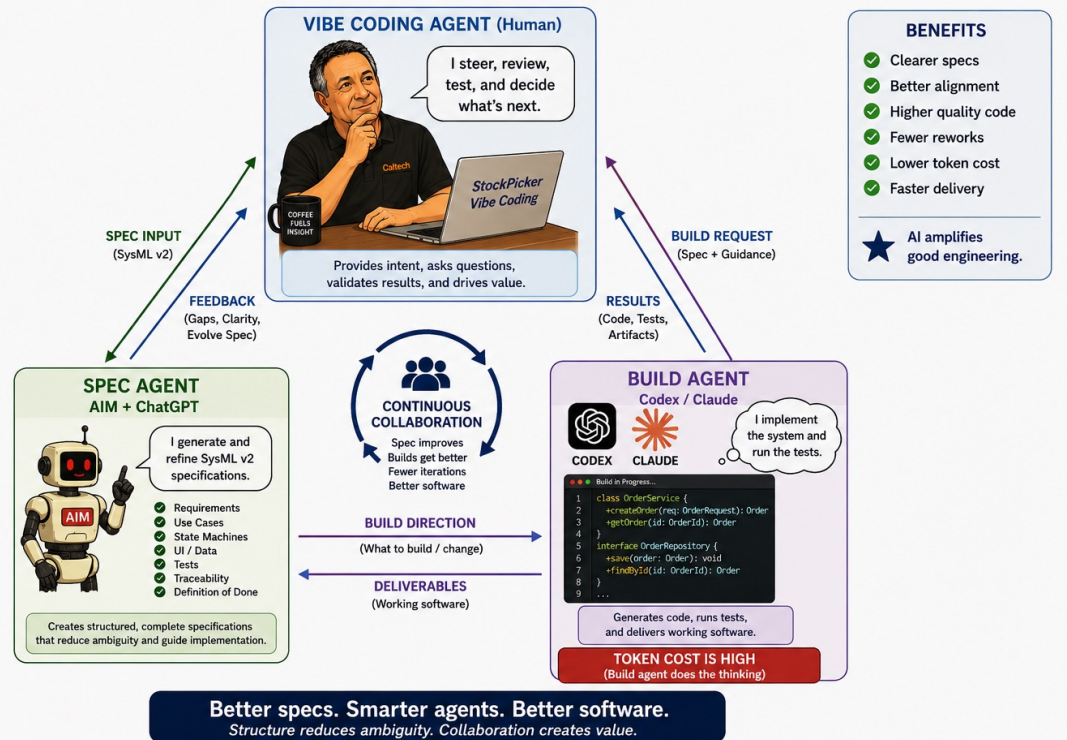
Disclaimer:

My first book featured a UML model for a portfolio management system so I was not completely inexperienced



THE STOCKPICKER DEVELOPMENT TEAM

Three agents. One goal: better specs, better software.



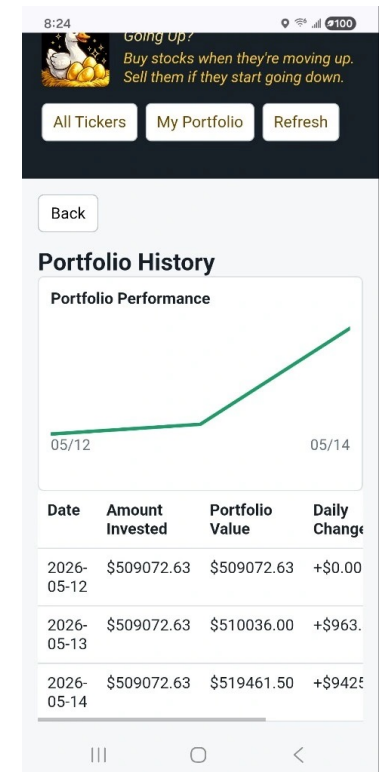
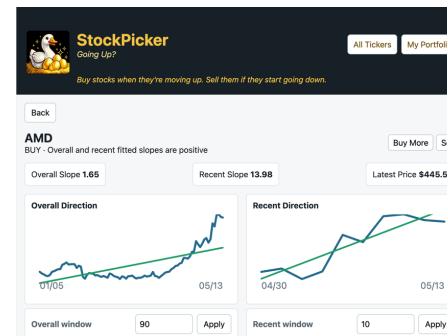
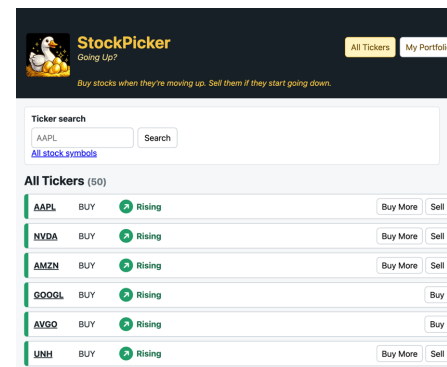
StockPicker – one human developer, and not much time spent

Combined Spec-Driven Development and Vibe Coding – real application; real iterations

About 3 calendar weeks, part time (30 or 40 hours)

StockPicker can:

- Access Stock Data from an online feed
- Build a Simulated Portfolio
- Examine trends for individual stocks
- Track Portfolio Performance over time



Better Specs Lead to Better Outcomes

Build testable specs, get testable code

Bugs are spec errors

"If the system does not behave as intended, the specification was incomplete"



Spec gives a clear set of Requirements to test against

```
# StockPicker 3 - Consolidated Requirements Baseline
=====
SYSTEM DESCRIPTION
=====

StockPicker is a portfolio research and decision-support system for simulated portfolio evaluation.

The current version:
- uses daily-close market data,
- analyzes directional stock behavior,
- supports simulated portfolio construction,
- and helps users evaluate investment strategies over time.

StockPicker does not:
- connect to brokerage accounts,
- execute real financial transactions,
- or provide financial advice.

All portfolio activity in the current version is simulated.

Users remain responsible for their own investment decisions.

=====
FUNCTIONAL REQUIREMENTS
=====

# Platform and Architecture Requirements

ARCH_001 - Web Application
StockPicker shall operate as a browser-based web application.

ARCH_002 - Graphical User Interface
The system shall provide an interactive graphical user interface and shall not rely on command-line
interaction for normal operation.

ARCH_003 - Online Market Data Sources
The system shall obtain market data from online data services rather than from manually maintained
```

```
# Transaction History Requirements

HIST_001 - Transaction History Semantics
The visible history table shall represent simulated portfolio-event history rather than automatic
valuation history.

HIST_002 - Event Types
History shall include:
- Buy events
- Buy More events
- Sell events

HIST_003 - Buy Event History Row
A buy event shall create a history row showing:
- date
- event
- amount invested
- portfolio value after event
- snapshot delta
- cumulative profit/loss

HIST_004 - Sell Event History Row
A sell event shall create a history row showing:
- date
- event
- amount sold
- portfolio value after event
- snapshot delta
- cumulative profit/loss

HIST_005 - Automatic Valuation Rows Hidden
Automatic valuation rows created by startup refresh or market-data loading shall not appear in visible
history.

HIST_006 - No Refresh Events
Visible history shall not display refresh events.

HIST_007 - Cash-Flow Separation
Purchases and sales shall be treated as cash-flow events rather than investment gains or losses.

HIST_008 - Cumulative Profit/Loss Formula
Cumulative profit/loss shall be calculated using:
- active portfolio value
- cumulative amount sold
- cumulative amount invested

HIST_009 - Snapshot Delta Calculation
Snapshot delta shall represent the change in cumulative profit/loss relative to the previous displayed
history row.
```

Testing Use Case Threads is very effective

```
TEST_006 - Requirement Assertion Tests
The implementation shall include assertion tests that verify each testable functional requirement.

TEST_007 - Use Case Thread Tests
The implementation shall include use case thread tests covering:
- basic paths
- alternate paths
- exception paths

TEST_008 - Postcondition Verification
Use case thread tests shall verify stated postconditions.

TEST_009 - Coverage Verification
```

```
=====
USE CASE NARRATIVES
=====

# StockPicker 3 - Use Case Narratives

## Use Case: UC_001 - View Ticker Recommendations

Actor: User

Description:
The user reviews ticker recommendations on the AllTickers page.

Basic Path:
1. System displays the AllTickers page.
2. System loads available ticker data.
3. System computes ticker recommendations using directional trend analysis.
4. System displays each ticker with its recommendation.
5. System sorts tickers by recommendation priority:
   - Buy first
   - Hold second
   - Exit last
6. System visually distinguishes recommendation categories.
7. User reviews the prioritized ticker list.
8. User selects a ticker.
9. System displays the Ticker Detail page.

Alternate Paths:

2A - No tickers available
1. System detects that no tickers are available.
2. System displays an empty-state message.
3. Use case ends.

4A - Market data unavailable for one or more tickers
1. System marks the affected ticker as unavailable.
2. System does not present unavailable data as a valid recommendation.
3. Rejoin Basic Path at Step 5.

Exception Paths:

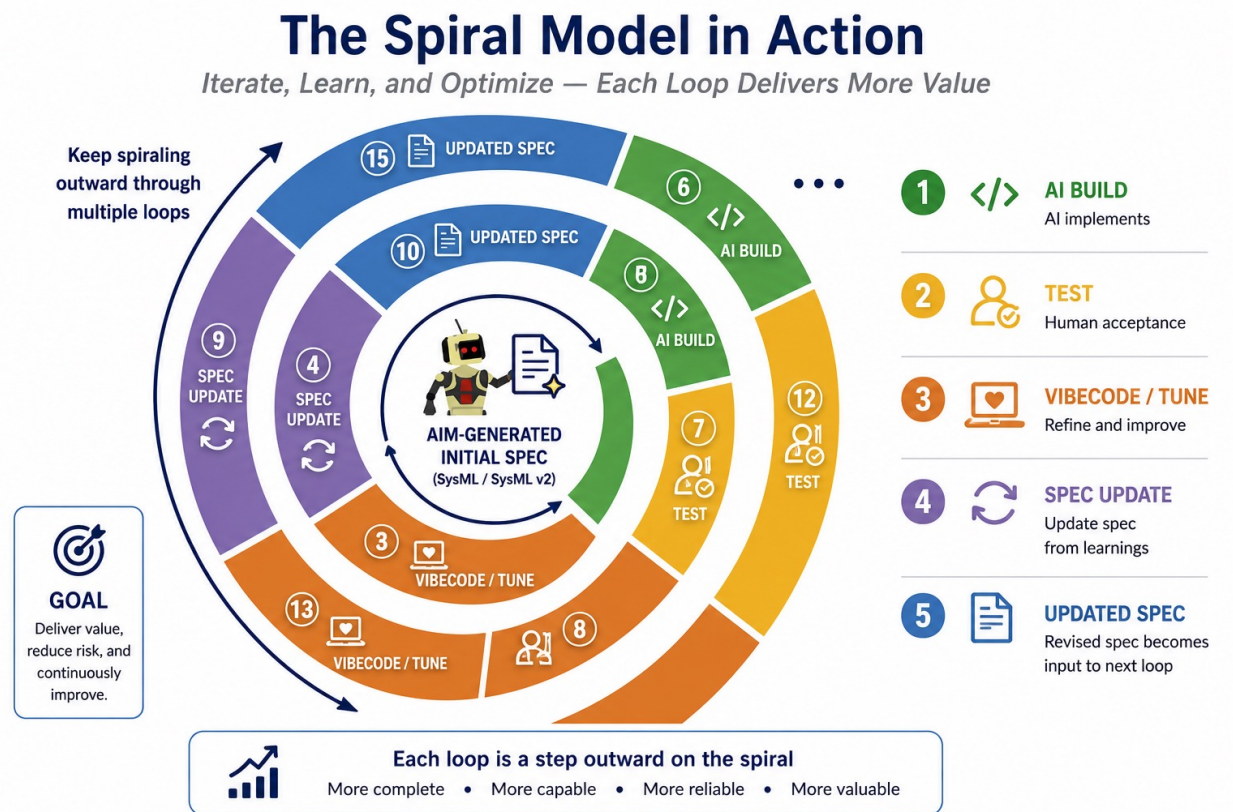
2E - AllTickers data cannot be loaded
1. System displays a recoverable error message.
2. System does not display misleading recommendations.
3. Use case ends.

Postconditions:
- Valid ticker recommendations are visible.
- Tickers are sorted Buy, Hold, Exit.
- Unavailable ticker data is clearly marked.
- Users can navigate to Ticker Detail
```

The Spiral Model Compressed by AI

A trip around the spiral used to take weeks, months, or years

Updating the specification after each trip around the spiral enables the agentic process to scale

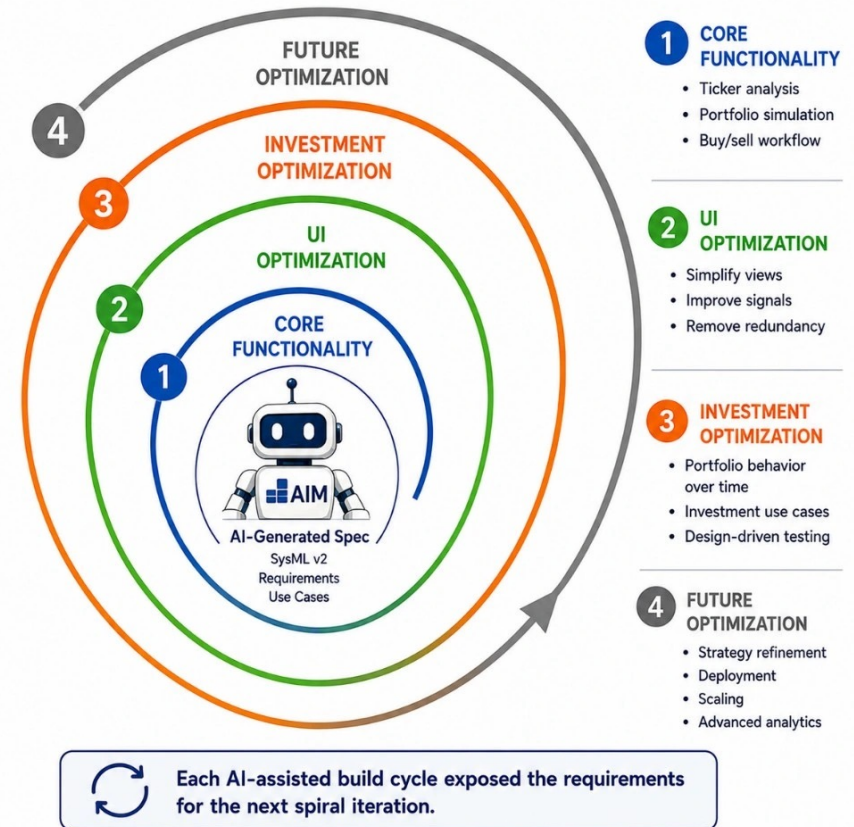


The Spiral Model Accelerated

Iteration time reduced from weeks/months **to about an hour** for Spec-Build-Tweak-Update Spec

Updating the spec at the end of each iteration is the key

STOCKPICKER 2.0 • SPIRAL DEVELOPMENT JOURNEY



One more trip around the spiral - StockPicker Test 4

Server-Side Portfolio Storage

Account Management

Multiple Portfolios per Account

```
# StockPicker Test 4 Addendum
## Accounts, Server-Side Persistence, Multiple Portfolios, and
Persistent Data Model
```

```
=====
=====
OVERVIEW
=====
=====
```

This addendum extends the StockPicker application from a single-browser-session prototype into a persistent multi-portfolio application with authenticated user accounts.

Primary goals:

- Eliminate accidental portfolio/history loss
- Move persistence responsibility from browser local storage to the server
- Support multiple named portfolios per account
- Preserve experimental and comparative investment strategies
- Improve long-term usability and trustworthiness
- Establish persistence architecture for future subscription deployment

Spec Agent generates new Requirements and Use Cases

MULTIPLE PORTFOLIO MANAGEMENT

REQ-410 – Multiple Portfolios

An authenticated user shall be able to own one or more named portfolios.

REQ-411 – Create Portfolio

The system shall allow the user to create a new named portfolio.

REQ-412 – Rename Portfolio

The system shall allow the user to rename an existing portfolio.

REQ-413 – Delete Portfolio

The system shall allow the user to delete a portfolio.

Deletion shall require explicit user confirmation before permanent removal or archival.

REQ-414 – Active Portfolio Selection

The user shall be able to select one active portfolio at a time.

REQ-415 – Portfolio Switching

Switching portfolios shall update:

- positions
- transaction history

USE CASE: UC-417 SWITCH ACTIVE PORTFOLIO

Actor:

Authenticated User

Description:

The user switches from one active portfolio to another active portfolio.

Basic Path:

1. User selects a different portfolio.
2. System validates ownership.
3. System saves pending changes to the current portfolio.
4. System retrieves selected portfolio data.
5. System restores selected portfolio state.
6. System updates active portfolio reference.
7. System refreshes:
 - charts
 - positions
 - transaction history
 - performance history
 - portfolio settings
8. System displays updated dashboard.

Exception Paths:

From plan to build in under an hour

This is a fairly large jump, so I'm going to make Test 4 usable in slices. I'm starting with a server API that can own accounts, sessions, and portfolios, using server-side JSON persistence locally so we avoid adding npm dependencies. The cPanel Python deployment can map the same API shape onto Python's built-in persistence later.

✍ Created 1 file, edited 1 file

The first server-side account/persistence API is in place locally: register/login/session cookies, portfolio CRUD, active portfolio, import/export, password change/reset stub, and ownership checks. I'm going to add focused domain/API tests now before wiring the UI, so failures stay small and readable.



Build Agent implements the new Requirements and Use Cases



StockPicker

Going Up?

Buy stocks when they're moving up. Sell them if they start going down.

[All Tickers](#) [My Portfolio](#) [Account](#) [Log Out](#)

Account

Signed in as doug@parallelagile.com

Active Portfolio

Portfolio

5 yellow tickers

Current portfolio: 5 green tickers

Create Portfolio

Name

Green 100-share baselin

Description

Optional

Manage Active Portfolio

New name

5 green tickers

Backup and Migration

Import JSON

Paste exported portfolio JSON here



StockPicker

Going Up?

Buy stocks when they're moving up. Sell them if they start going down.

[All Tickers](#) [My Portfolio](#) [Account](#) [Log Out](#)

My Portfolio (5 green tickers)

[Switch Portfolios](#) [Portfolio History](#)

Total Value \$115742.61 Gain/Loss +\$0.00 Gain/Loss % +0.00%

Ticker	Purchase Date	Purchase Price	Current Price	Shares	Value	\$ G/L	% G/L	Actions
AMZN	2026-05-27	\$269.56	\$269.56	100	\$26956.00	+\$0.00	+0.00%	Buy More Sell
AAPL	2026-05-27	\$310.77	\$310.77	100	\$31077.00	+\$0.00	+0.00%	Buy More Sell
ORCL	2026-05-27	\$191.26	\$191.26	100	\$19126.00	+\$0.00	+0.00%	Buy More Sell



StockPicker

Going Up?

Buy stocks when they're moving up. Sell them if they start going down.

[All Tickers](#) [My Portfolio](#) [Account](#) [Log Out](#)

My Portfolio (5 yellow tickers)

[Switch Portfolios](#) [Portfolio History](#)

Total Value \$234922.50 Gain/Loss +\$0.00 Gain/Loss % +0.00%

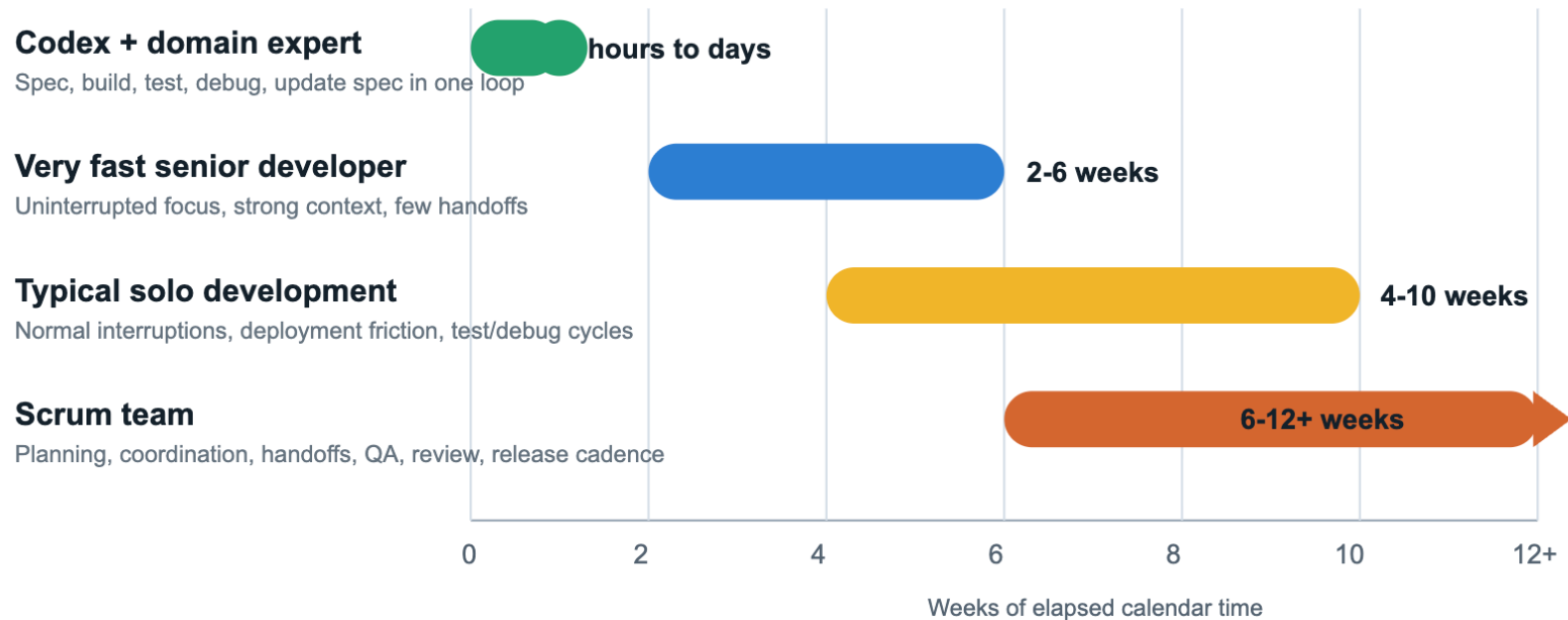
Ticker	Purchase Date	Purchase Price	Current Price	Shares	Value	\$ G/L	% G/L	Actions
AVGO	2026-05-27	\$420.20	\$420.20	100	\$42020.50	+\$0.00	+0.00%	Buy More Sell
UNH	2026-05-27	\$381.39	\$381.39	100	\$38139.00	+\$0.00	+0.00%	Buy More Sell
GOOGL	2026-05-	\$391.80	\$391.80	100	\$39180.00	+\$0.00	+0.00%	Buy More

It literally builds in a few minutes when given the spec!

Time Savings is Enormous

Cycle-Time Compression

Estimated elapsed calendar time for the StockPicker Test 4 feature set



AI Coding Agent boosts Productivity – without burning many tokens

What Did Codex Do?

From SysML-style requirements to a deployed, testable StockPicker web app

1 Interpreted The Spec

- Parsed SysML-style requirements
- Identified assumptions and gaps
- Mapped requirements into app structure and tests

2 Built The Product

- Built ticker, portfolio, and detail views
- Implemented recommendation logic and charts
- Added buy, buy-more, sell, and partial-sell flows
- Added valuation, gain/loss, and history behavior

3 Connected Real Data

- Added server-side market-data fetching
- Added cache and refresh behavior
- Added safeguards for stale or failed data

4 Added Persistence And Accounts

- Added registration, login, logout, and sessions
- Added multiple named portfolios per account
- Moved portfolio state to server-side persistence
- Added export/import backup support

5 Tested And Deployed

- Generated regression and use-case tests
- Fixed bugs from live exploratory testing
- Packaged for cPanel/GoDaddy deployment
- Debugged deployment and startup-file issues

6 Fed Learning Back

- Captured discovered behavior as requirements
- Updated spec addenda for future builds

Moving the spec activities out of the coding agent minimized token use

What Did The Spec Agent Do?

No tokens used beyond my \$20/month ChatGPT subscription.

1 Structured The Idea

- Turned product intuition into explicit requirements
- Organized intent into features and constraints
- Kept the SysML-style spec as source of truth

2 Modeled The System

- Defined app structure and system responsibilities
- Captured portfolio, position, and transaction concepts
- Added accounts, sessions, portfolios, and persistence

3 Defined Behavior

- Wrote use-case narratives and workflows
- Added preconditions, postconditions, and alternatives
- Clarified market dates, snapshots, and isolation

4 Guided Testing

- Turned expected behavior into testable requirements
- Added guidance for account and persistence tests
- Identified edge cases for regression coverage

5 Updated The Baseline

- Integrated lessons from live app testing
- Reassembled scattered addenda into a baseline
- Kept future builds aligned with what was learned

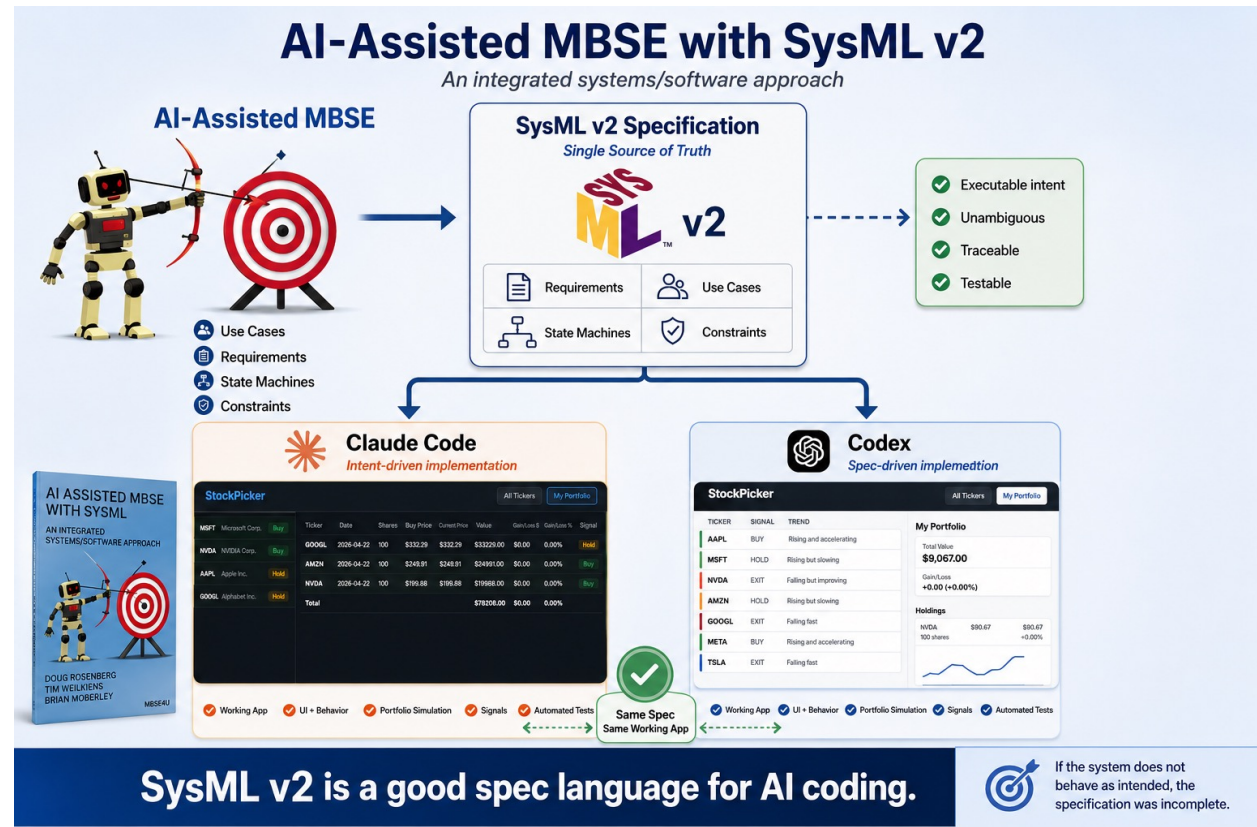
6 Reduced Build Risk

- Separated brainstorming from build instructions
- Made ambiguous behavior visible before coding
- Gave the build agent a clearer target

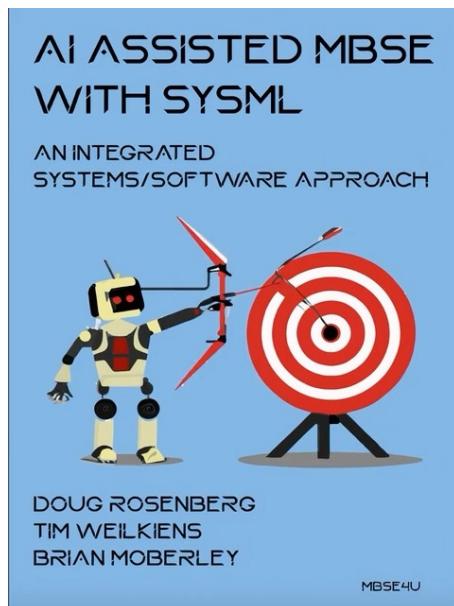
Results

Same structured specification → similar working systems

SysML v2 turns out to be a good spec language for coding agents



AIM enables rapid spec development with minimal token usage



Traditional Development:

Minimize spec changes because they are expensive

AI-Assisted MBSE:

Maximize spec refinement because it is cheap

Late Last Night – We fed the Stock Picker 8 spec back to Claude

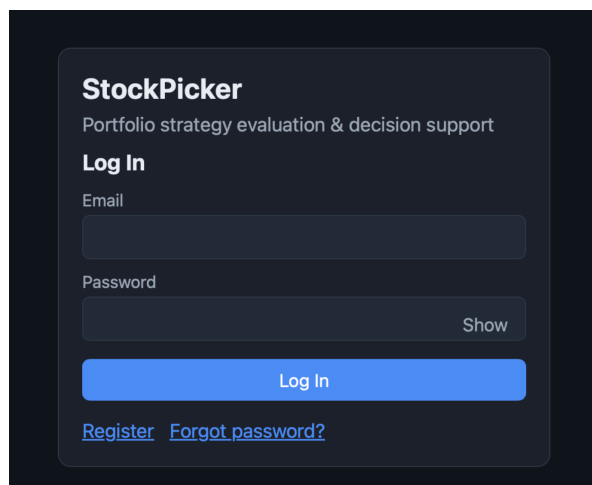
Spec is a 42-page PDF

STOCKPICKER SPECIFICATION BASELINE v8.0

TABLE OF CONTENTS

1. Overview
 - 1.1 Purpose
 - 1.2 Scope
 - 1.3 Design Philosophy
 - 1.4 Authoritative Data Model
 - 1.5 Testing Strategy
2. Glossary
3. Requirements
 - 3.1 Account Management
 - 3.2 Portfolio Management
 - 3.3 Portfolio History and Performance
 - 3.4 SQLite Persistence
 - 3.5 Security
 - 3.6 User Interface Requirements
4. Use Cases
 - 4.1 Use Case Inventory
 - 4.2 Use Case Narratives
5. State Machines
 - SM-002 Market Data Cache Lifecycle
 - SM-003 Portfolio Position Lifecycle
 - SM-004 Overall UI Flow
6. Testing Rules

Claude took 30 minutes to build it – including Account Management



StockPicker
Portfolio strategy evaluation & decision support

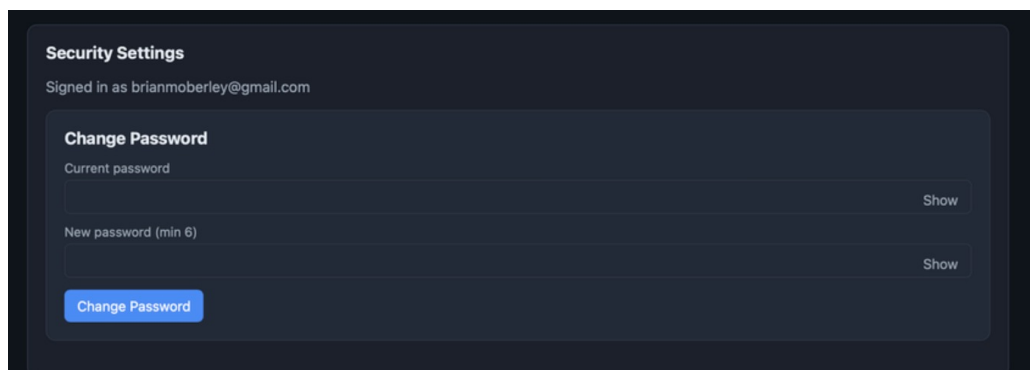
Log In

Email

Password
 Show

[Log In](#)

[Register](#) [Forgot password?](#)



Security Settings
Signed in as brianmoberley@gmail.com

Change Password

Current password
 Show

New password (min 6)
 Show

[Change Password](#)

Full disclosure: Actually, it took 32 minutes and 2 seconds to build and test

Including enhancements to Portfolio Management, Transactions, etc.

Active Portfolio

My Portfolio

Switch: My Portfolio

+ New

Rename

Delete

Portfolio Totals

PORTFOLIO VALUE

\$0.00

NET INVESTED CAPITAL

\$0.00

GAIN / LOSS

\$0.00

GAIN / LOSS %

+0.00%

Performance Index Graph

Strategy Return: +0.00%

Not enough transaction and market-data history yet to display the Performance Index.

Holdings

Ordered by triage priority

No active holdings. Buy a stock to begin.

Buy a Stock

Ticker

Shares

AAPL — Apple Inc.

Shares

Buy

Before you trade: Simulation and decision-support tool only. Not investment, financial, or tax advice. All investment decisions are your responsibility.

Transaction History					
Date	Ticker	Action	Shares	Price	Amount
2026-06-10	GOOGL	Buy	15	\$371.15	\$5,567.25
2026-06-10	AAPL	Buy	100	\$361.19	\$36,119.00

Total cost to build from spec \$13. Many tokens saved by spec agent.

```

• The full specification is implemented and tested. 97/97 tests pass.
```

```

Settings   Status   Config   Usage   Stats

Session

Total cost:           $13.14
Total duration (API): 29m 13s
Total duration (wall): 32m 2s
Total code changes:    4346 lines added, 24 lines removed
Usage by model:
  claude-haiku-4-5:    472 input, 14 output, 0 cache read, 0 cache write ($0.0005)
  claude-opus-4-8:     10.9k input, 135.7k output, 15.4m cache read, 320.1k cache write ($13.14)

Current session
██████████████████████████████████████████████████████████████████████████ 10% used
Resets 1am (America/Chicago)

Current week (all models)
██████████████████████████████████████████████████████████████████████████ 1% used
Resets Jun 13 at 6am (America/Chicago)

Current week (Sonnet only)
██████████████████████████████████████████████████████████████████████████ 0% used

What's contributing to your limits usage?
Approximate, based on local sessions on this machine – does not include other devices or claude.ai

Last 24h · these are independent characteristics of your usage, not a breakdown

54% of your usage was at >150k context
Longer sessions are more expensive even when cached. /compact mid-task, /clear
when switching to new tasks.

Skills, subagents, plugins, and MCP servers
No attribution data yet · accumulates as you use Claude
```

Public Courses from Caltech CMTE



Center for Technology & Management Education

[Educators](#)
[Why Us](#)
[Contact](#)

[Individuals](#)
[Organizations](#)

Pain-Free MBSE for Enterprise Teams

Practical Approaches in Model-Based Methods

Model-Based Systems Engineering (MBSE) should accelerate delivery and improve decisions—not add friction. Pain-Free MBSE for Enterprise Teams is a practice-driven program that helps organizations apply MBSE in a lean, pragmatic way, focusing modeling effort only where it delivers clear value.

Rather than emphasizing SysML notation in isolation, this course addresses the modeling decisions that disproportionately increase effort while delivering little benefit. Participants learn how to reduce over-modeling, avoid tool-driven workflows, and align system models with real-world software and hardware development practices.

Start Date	July 18
Time	Sat 8:00 AM - 2:00 PM Pacific Time
Duration	24 Hours
Format	LIVE-ONLINE
Program Type	Open-Enrollment/Public
Certificate Type	Short Course
CEUS	2.4
Program Number	5480726
Fees	\$2,550

[SEE FULL COURSE INFO](#)

[REGISTER](#)

[CONTACT ADVISOR](#)



Center for Technology & Management Education

[Educators](#)
[Why Us](#)
[Contact](#)

[Individuals](#)
[Organizations](#)

AI-Assisted MBSE

Generative AI-Driven Models in Systems Engineering

The *AI-Assisted Model-Based Systems Engineering (AIM)* Workshop is a compelling enterprise training program at the intersection of AI and digital transformation. Designed for engineering organizations, this 24-hour course equips teams to apply generative AI tools within MBSE workflows, accelerating model creation, improving analysis, and preparing systems engineers for the next generation of intelligent design.

Start Date	June 13
Time	Sat 8:00 AM - 3:00 PM Pacific Time
Duration	24 Hours
Format	LIVE-ONLINE
Program Type	Open-Enrollment/Public
Certificate Type	Certificate
CEUS	2.4
Program Number	5380626
Fees	\$1,950

[SEE FULL COURSE INFO](#)

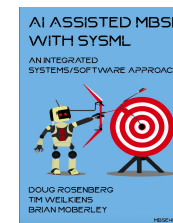
[REGISTER](#)

[CONTACT ADVISOR](#)



<https://ctme.caltech.edu/pain-free-mbse.html>

Four 6-hour sessions,
starts Saturday, July 18



<https://ctme.caltech.edu/ai-assisted-mbse.html>

Four 6-hour sessions,
starts Saturday, June 13



Caltech

©Caltech

Rosenberg | Pain-Free MBSE

<https://ctme.caltech.edu>

31