2026 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES
SANTA CLARA, CA, USA
March 2 - 5, 2026

Vayavya Labs

a silicon to systems solutions company

Tvastaa VP

(twa - us - thaw - V.P)

Agentic AI for Virtual Platform Development

D G E M S 200
D-Globalist Award 2025

nasscom
Inspire Award 2025
Delivery Excellence in Tech Services

About Vayavya Labs

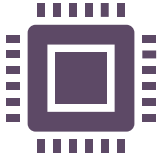
Corporate Snapshot

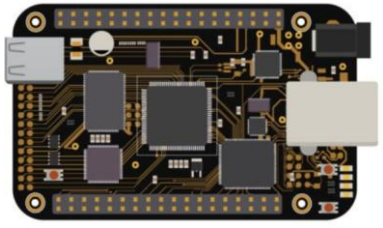
19 Years

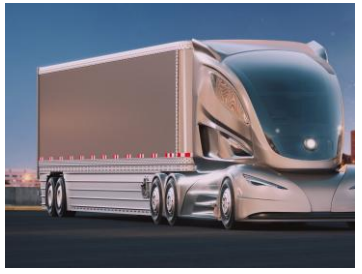
India
US
Japan
Europe

Our Offerings

“Silicon to System” Engineering Solutions







Semiconductor/ESL Offerings from Vayavya

- Performance Modeling
- HSI Consulting

- Software Driven Verification
  - Bare metal/Linux
  - Portable Stimulus
- Emulation/FPGA

- S.W. bring-up on silicon
- Productizing of S.W. stack

- End-to-end Virtual platform development
- QEMU, SystemC, C/C++ models, …

- BSP, device drivers, Protocol Stacks
- GPU, PCIe, CXL, UCIe, ETH, NVMe, …
- Linux, Android, …

Vayavya Labs

Agenda

- Introduction
- Design and Plan
- Stub Code Generation
- Code to model IP
- Impact
- Roadmap
- Case Studies

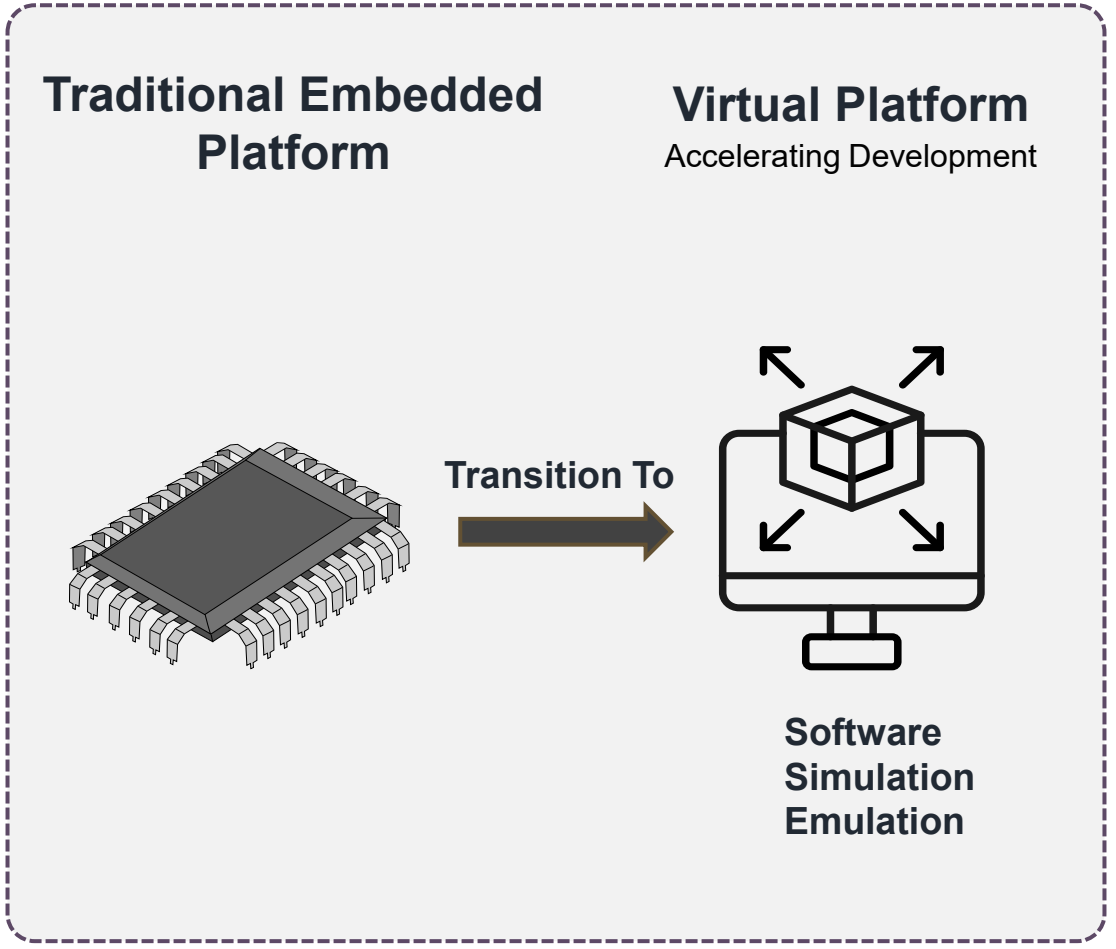
Introduction

Agentic approach through Tvastaa VP

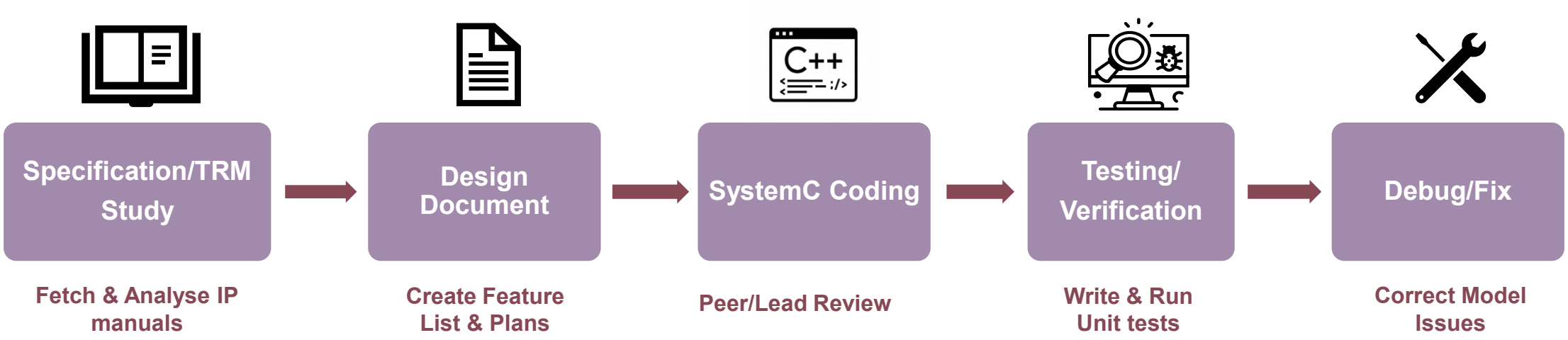
20min

Virtual Platforms

- Virtual Platforms have moved from experimentation to production
- Development and “productizing” of Virtual Platforms
  - Time & Resource consuming – often acts as a barrier to adoption
  - Needs to hit the usage “sweet spot”
  - SystemC and C/C++ modelling
    - A niche skill
    - Small developer base i.e. lack of a wider talent pool



Traditional SystemC Workflow: Key Pain Points

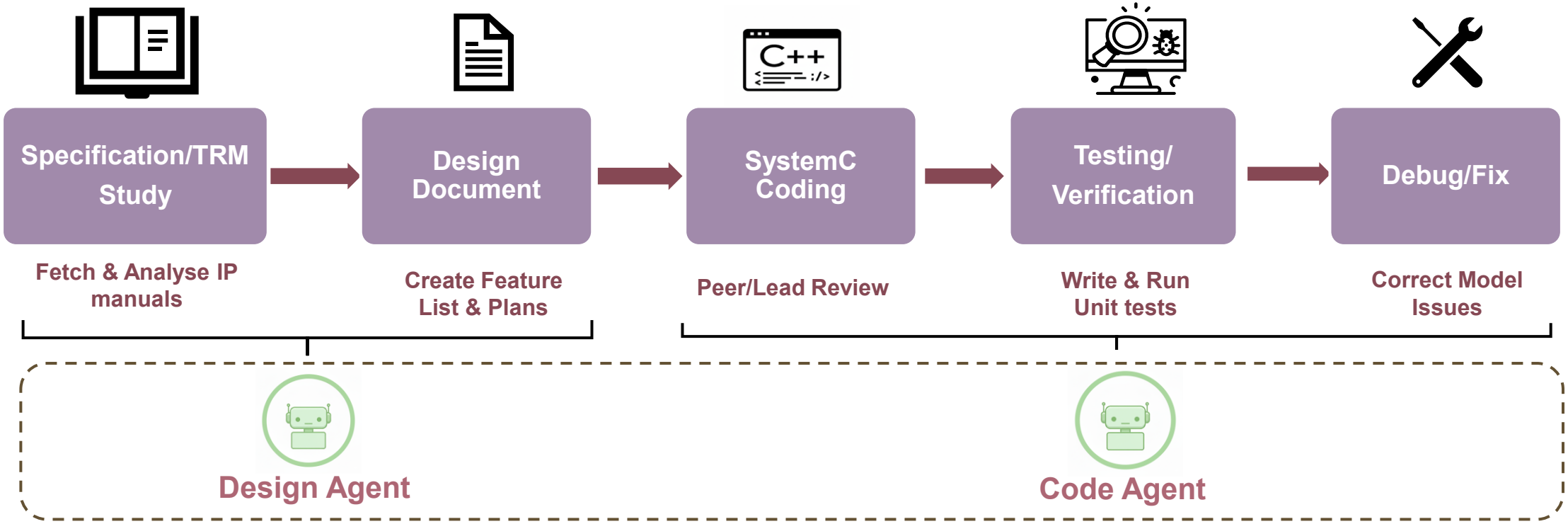


- Complete Spec understanding is a slow process
- Requires deep IP knowledge and SystemC expertise
- Skilled SystemC engineers are scarce

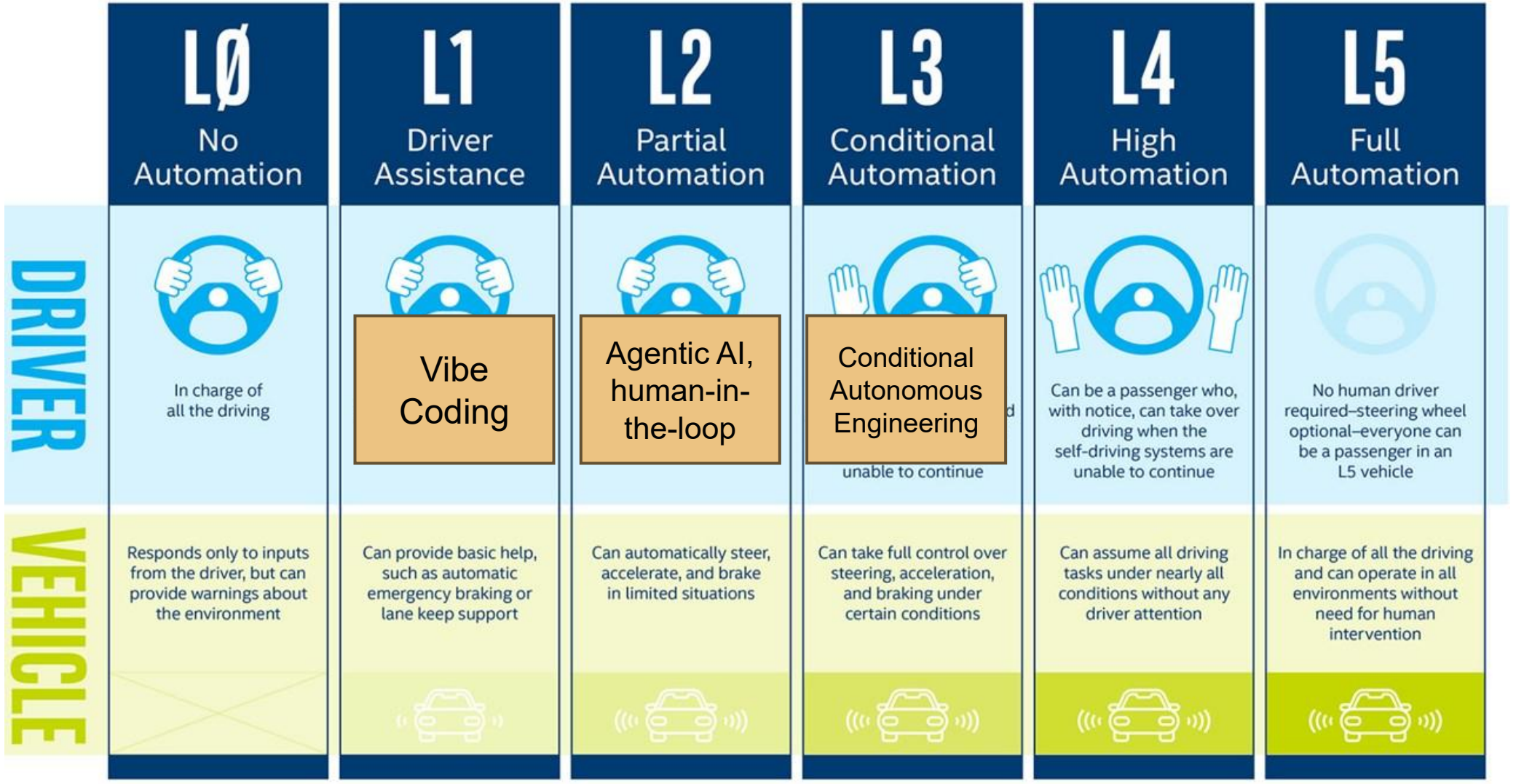
- High manual effort - coding, testing and debugging
- Hard to maintain design intent and code consistency
- Long cycle from spec to usable virtual prototype

Agentic AI approach addresses these challenges comfortably

How Agentic AI Approach Helps…

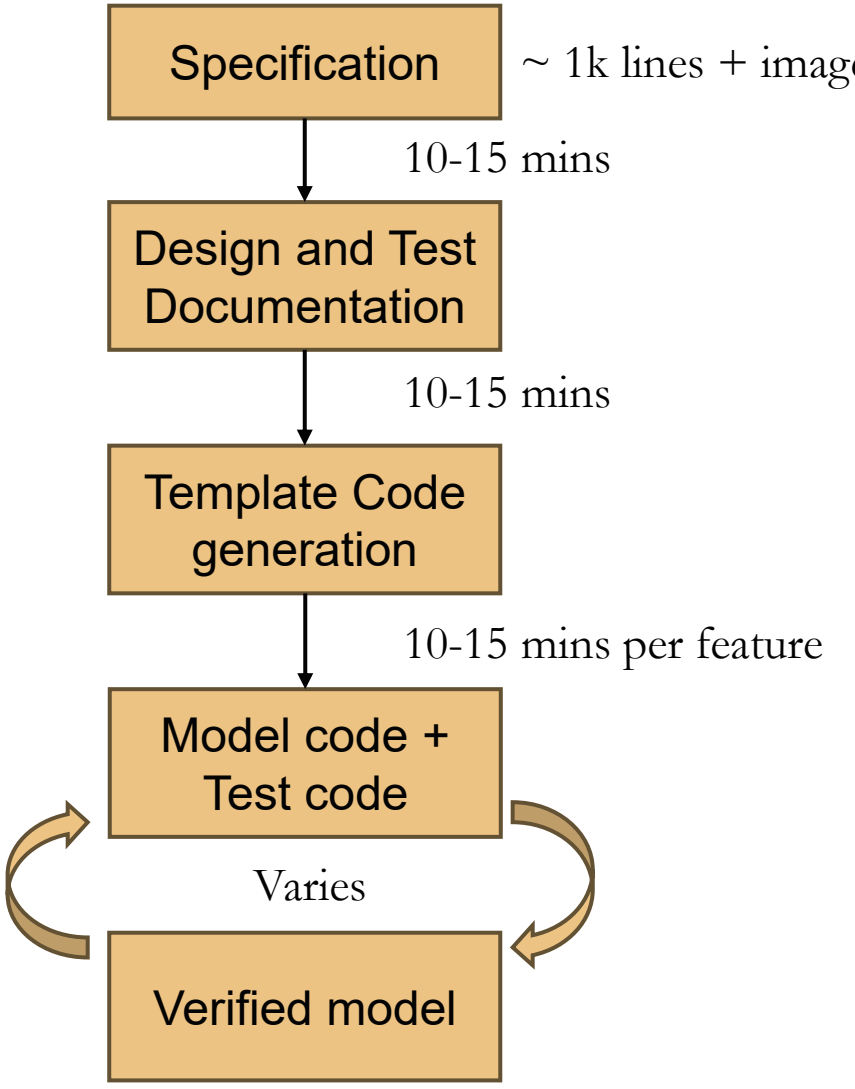


- Faster spec-to-model turnaround
- Reduced dependence on SystemC human experts
- Reduced design and documentation efforts
- Consistent code quality across engineers
- Automated test generation and debugging
- Faster iteration and prototype availability

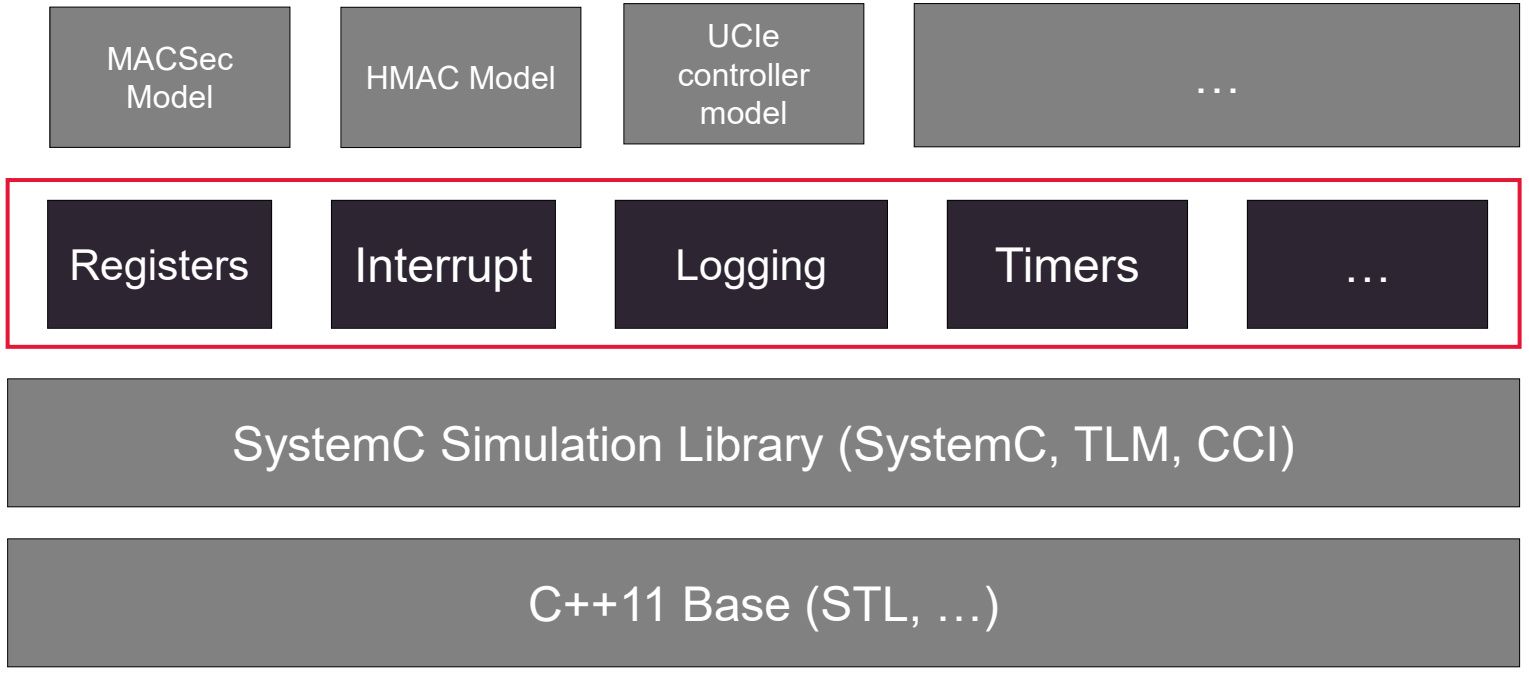


Our Demo today: HMAC

- SHA-2 hash-based authentication code generator
- Two modes: SHA-2 | HMAC based on SHA-2
- Multiple digest sizes supported (for both modes): SHA-2 256/384/512 hashing algorithm
- Configurable key length up to 1024-bit secret key for HMAC mode
- Support for context switching (via saving and restoring) across multiple message streams
- 32 x 32-bit message FIFO buffer



CSML: A Permissively licensed Modeling Library



- Fixes the missing features in SystemC for real-life peripheral models
- Vendor neutral
  - Supports Accellera Reference simulator
  - Supports commercial SystemC simulators
- Permissively licensed

“Tvastaa VP” – Our Agentic AI Tool

Before we see what Tvastaa VP is…

Vayavya Labs's Journey with LLMs

Genesis: Why We Looked at LLMs

- Despite good resource pool, we faced:
  - Growing customer demand across multiple SoCs/IPs
  - Long spec-to-model cycles
- Scaling talent linearly was not sustainable
- Started exploring how LLM could be leveraged for modeling workflows

From Exploration to Structured Adoption

- Began applying LLMs to SystemC modelling when they gained traction
- Two years of internal experimentation
- Presented work at DVCon U.S. 2024

DVCon U.S. 2024: Proof of Execution

- LLM-assisted SystemC Modeling with Qux constructs
- Prompt-driven, single-agent workflow
- ~20% productivity improvement

DVCon U.S. 2024 - Program Grid
7 March 2024

TIME (PST)	Cascade	Donner	Siskiyou
9:00 – 10:30	Data-Driven Design, Verification, Validation, and Signoff Case Studies of RISC-V SoCs (SIEMENS)	Emulation Moves Into 4-State Logic and Real Number Modeling (cadence)	**SystemC Model Code Generation using Large Language Models (Vayavya Labs)**
10:30 – 11:00		Coffee Break (Bayshore Foyer)	
11:00 – 12:30	Data-Driven Design, Verification, Validation, and Signoff Case Studies of RISC-V SoCs (continued) (SIEMENS)	FPGA Prototyping for Large Multi-Die / Multi-Core Designs (SYNOPSYS)	Advanced UCIe-based Chiplets verification from IP to SoC (cadence)
12:30 – 13:30		Lunch Sponsored by: SYNOPSYS (Sierra)	
13:30 – 15:00	Expanding role of Static Signoff in Verification Coverage (REAL INTENT)	RISC-V Core Verification: A New Normal in Verification Techniques (BREKER)	Automating the Integration Workflow with IP Centric Design (PERFORCE)

What changed in 2025: The Agentic Leap

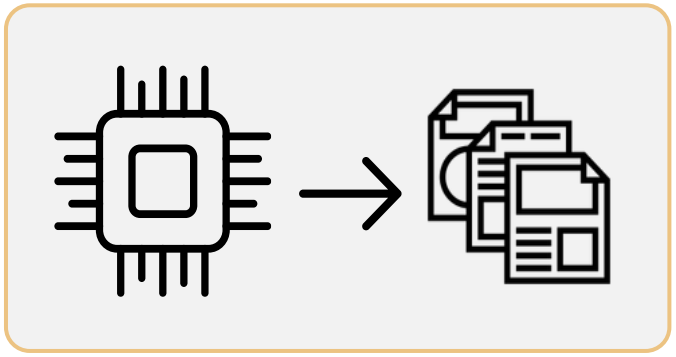
- Shift from manual modelling to Agent-driven tasks
- Domain specific Agentic workflows for Virtual Platforms (Spec ingestion, model planning, design docs, code and test generation)
- Covers the complete modelling lifecycle:
  HW IP Specifications → Verified SystemC TLM Model
- ~80% automation across lifecycle
- Human role moves from coding to supervision and approval
- Zero manual SystemC authoring

Tvastaa Agentic Platform Family

Tvastaa Janus

Code & System Compliance Automation

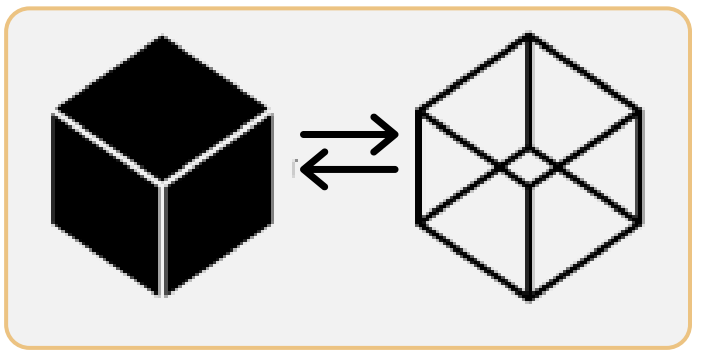
- Automates safety and security compliance workflows
- Identifies, triages, and tracks violations
- Aligns with real development and review cycles
- Reduces manual compliance effort



Tvastaa Driver

Device Driver & HSI Intelligence

- Intelligent device driver generation
- Translates HW specs into OS-ready software artifacts
- Automates HSI workflows
- Reduces integration surprises



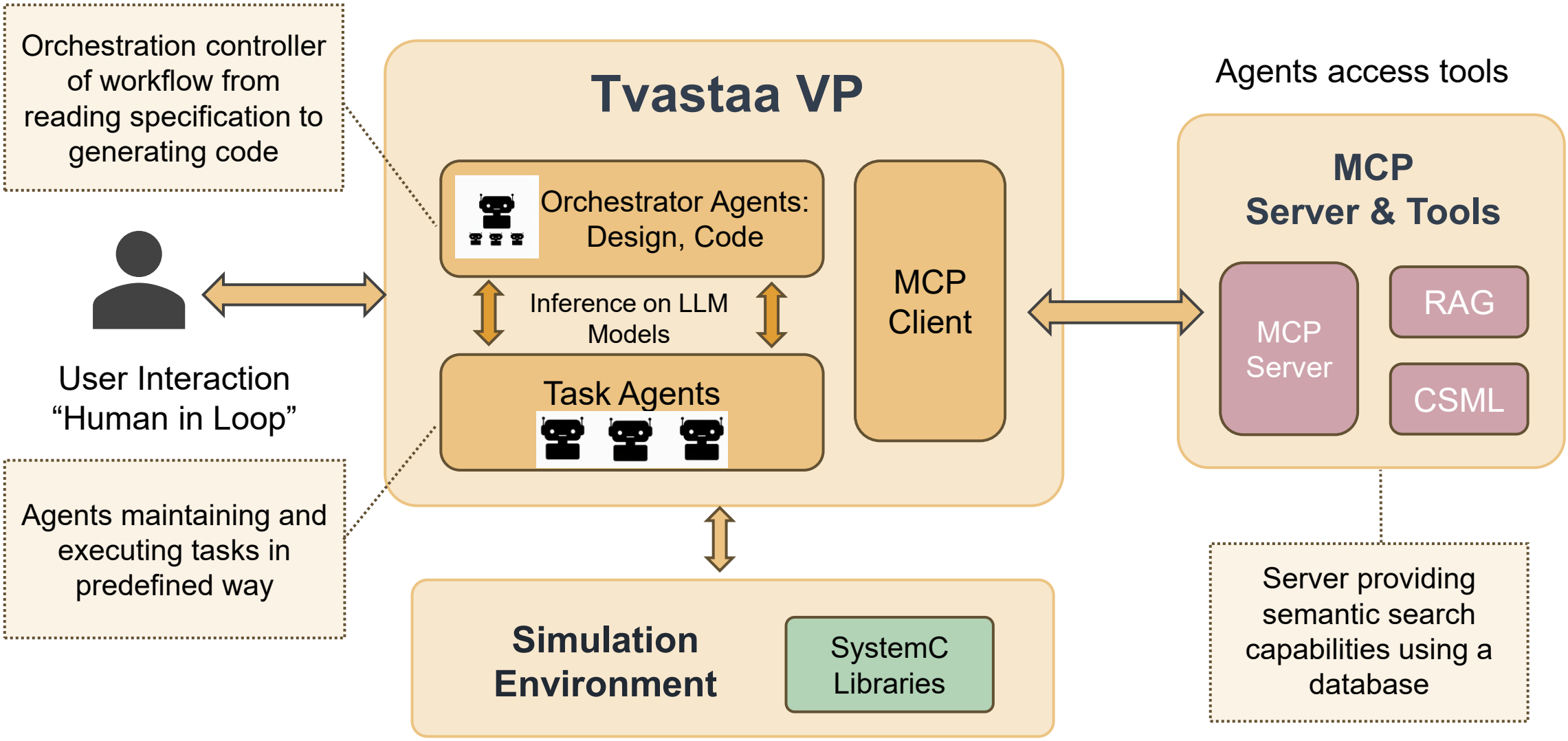
Tvastaa VP

Digital Twins for IPs & SoCs

- Generates digital twins for IPs and SoCs
- Enables early software development
- Accelerates pre-silicon validation
- Reduces dependency on physical hardware

Vayavya Labs

Tvastaa VP High-Level



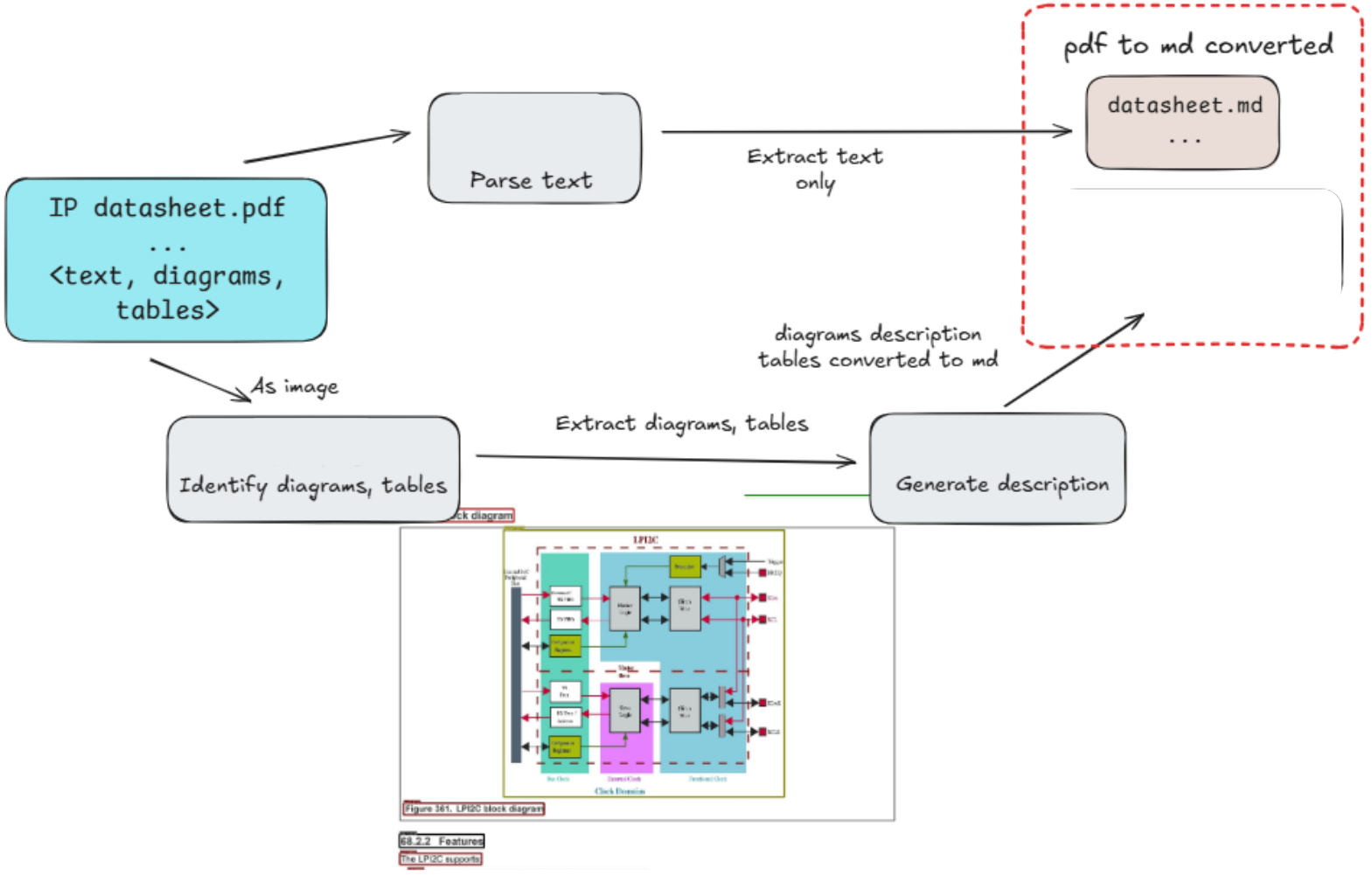
Pre-Processing

Prepare Input Data

40min

Preprocessing – “X” to Markdown

- LLMs work with text (tokens), all relevant information has to be converted to text
- IP Information is typically present as
  - PDF Datasheets, Technical Reference manuals
  - IP-XACT XML
  - XLS files
  - Text and diagrammatic representation
- Pre-processing ingests all data and generates markdown output

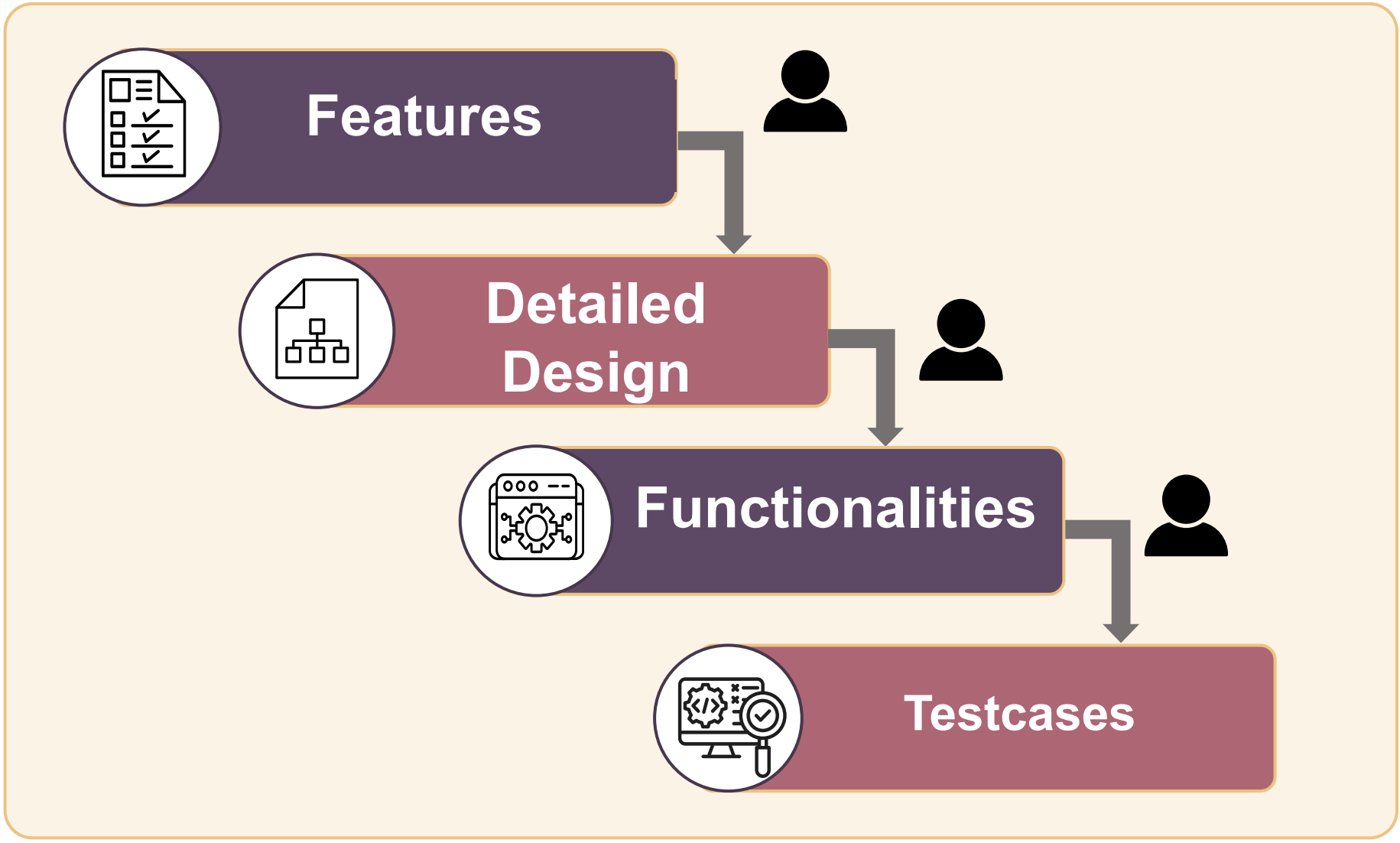


Design

Plan for IP functionality implementation

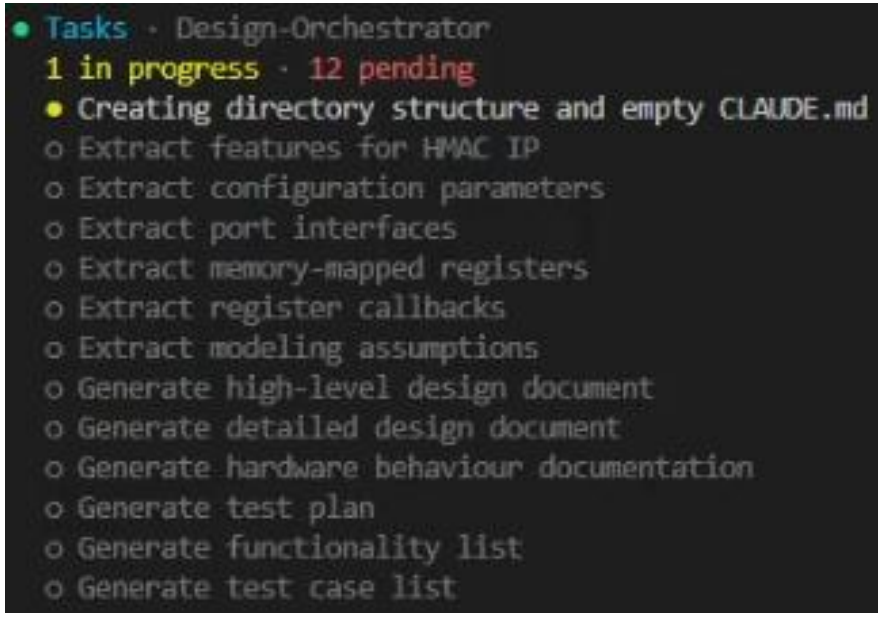
40min

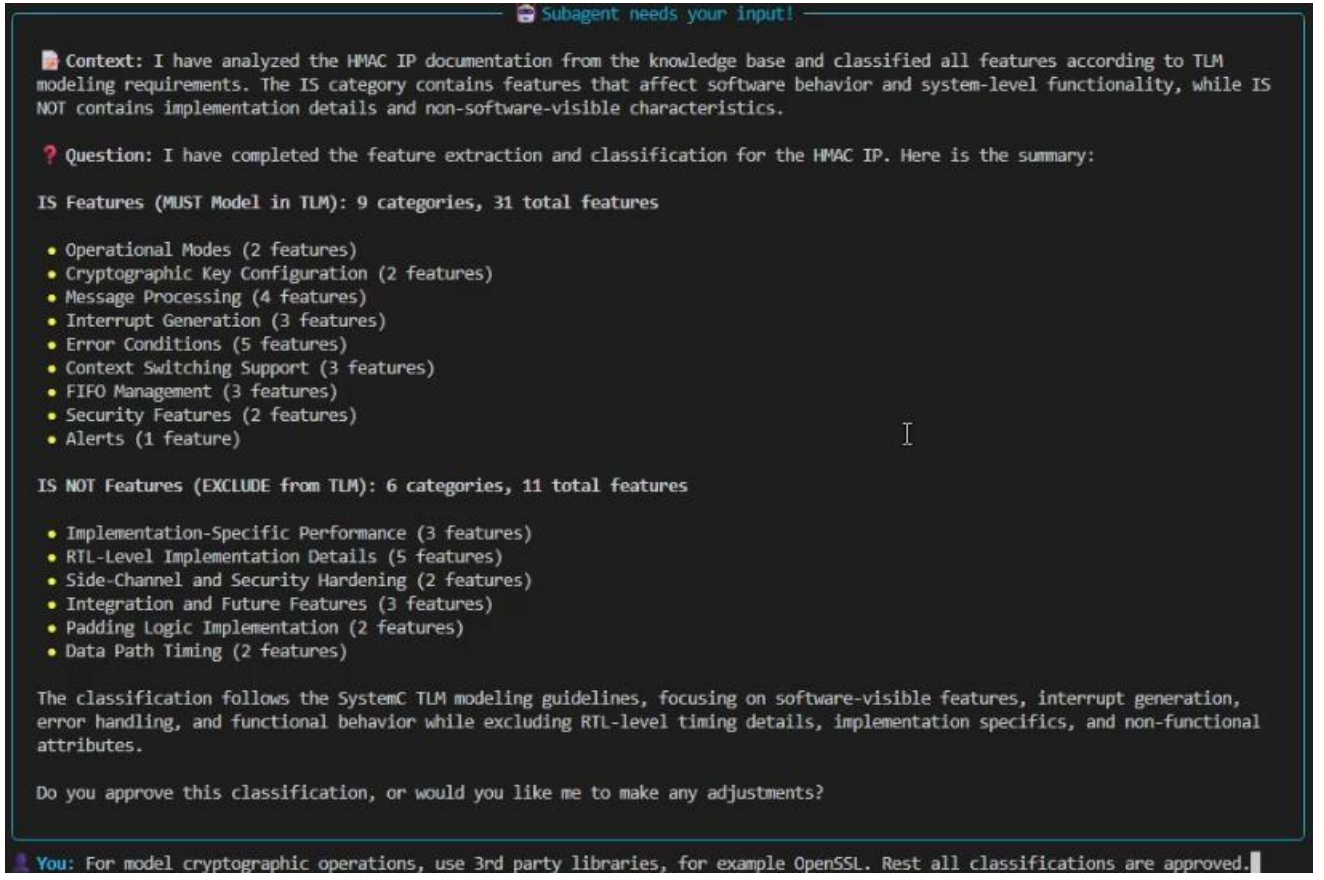
Tvastaa VP – Plan and Design



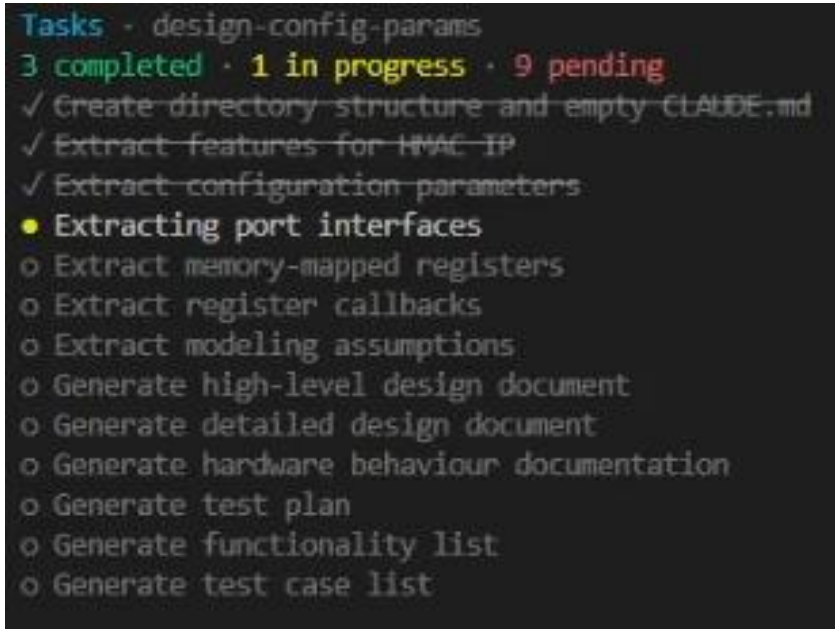
Demo – Hash-based Message Authentication Code (HMAC)

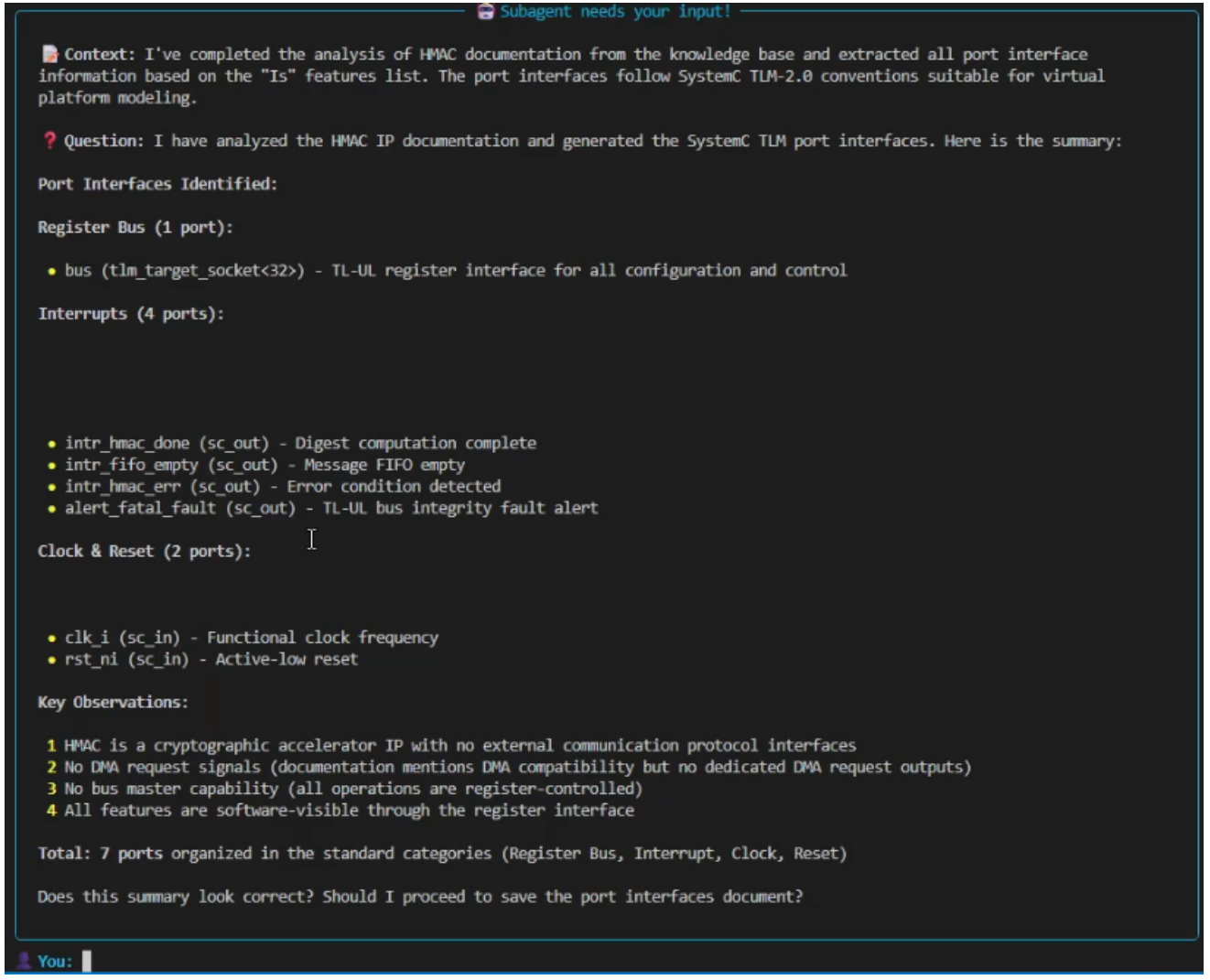
Design phase – Snapshots (1/2)





Design phase – Snapshots (2/2)





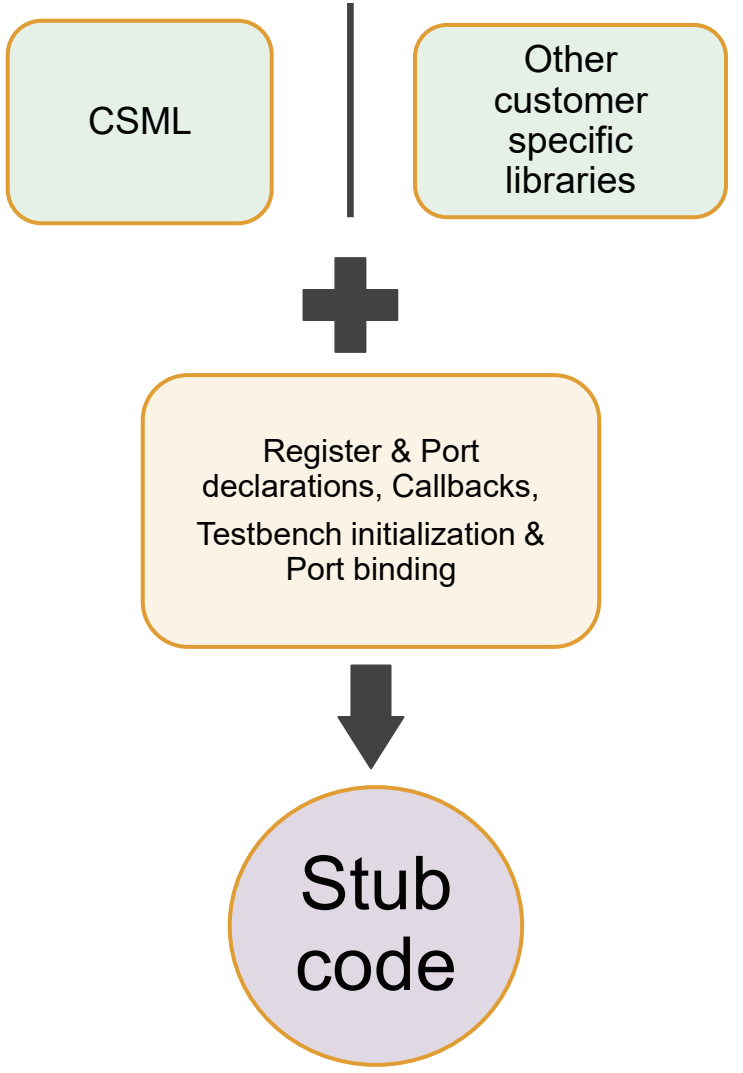
Stub/Template Code

Initial code generation

15min

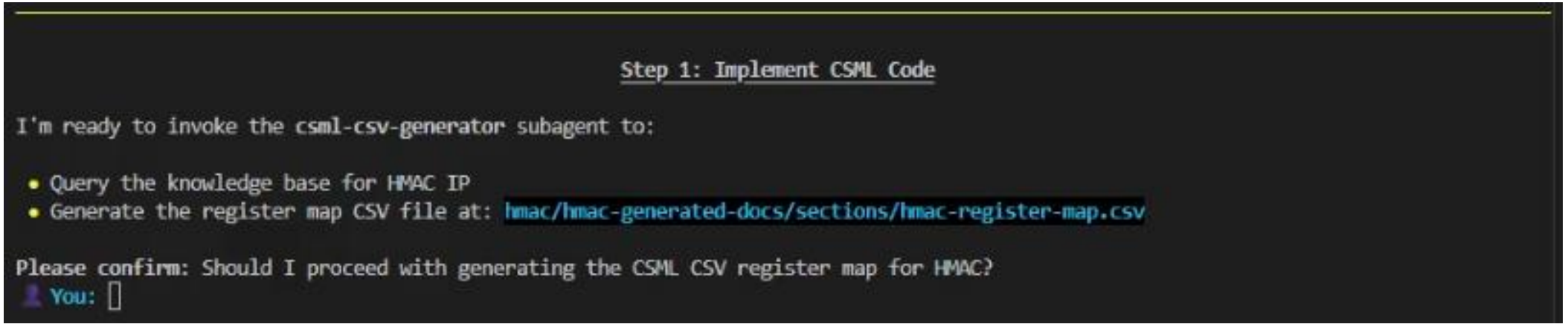
Tvastaa VP – Stub/Template Phase

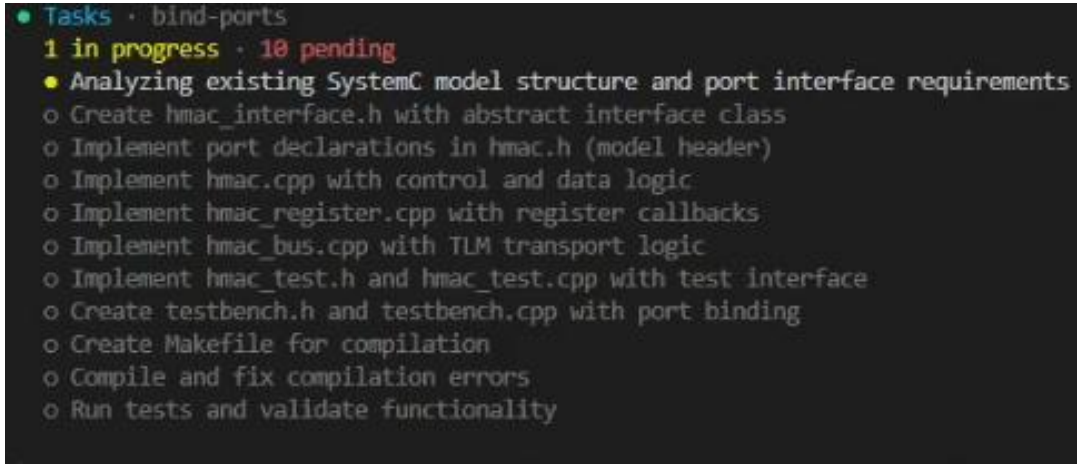
- Uses tools like CSML or any other customer specific libraries to generate initial code
- Generate initial code for given IP
  - Register declaration
  - Register callbacks
  - Port declarations
  - Initialization of testbench and port bindings
- Executes CSML tool to generate initial code



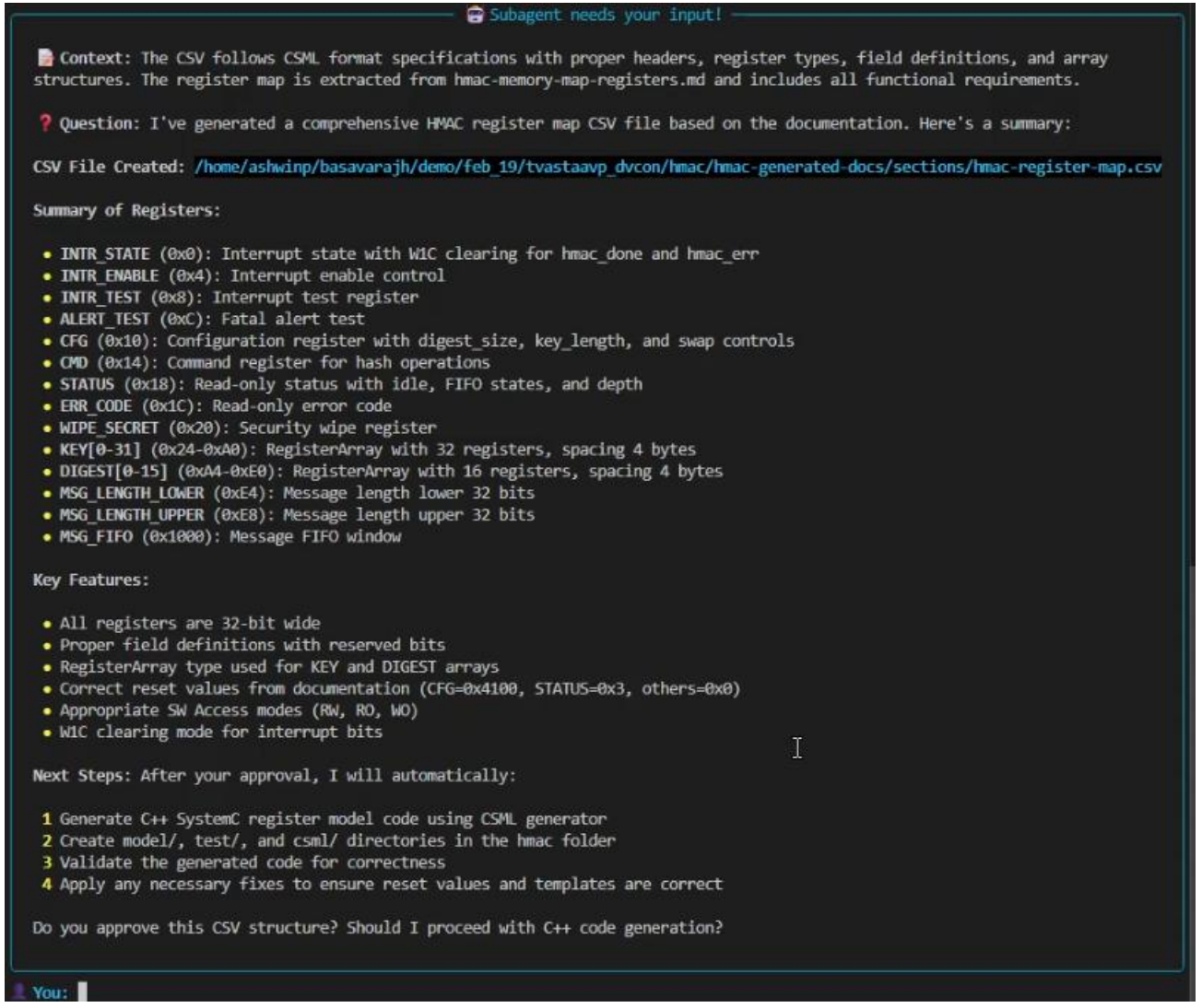
Demo – Stub Phase

Stub phase – Snapshots (1/2)





Stub phase – Snapshots (2/2)

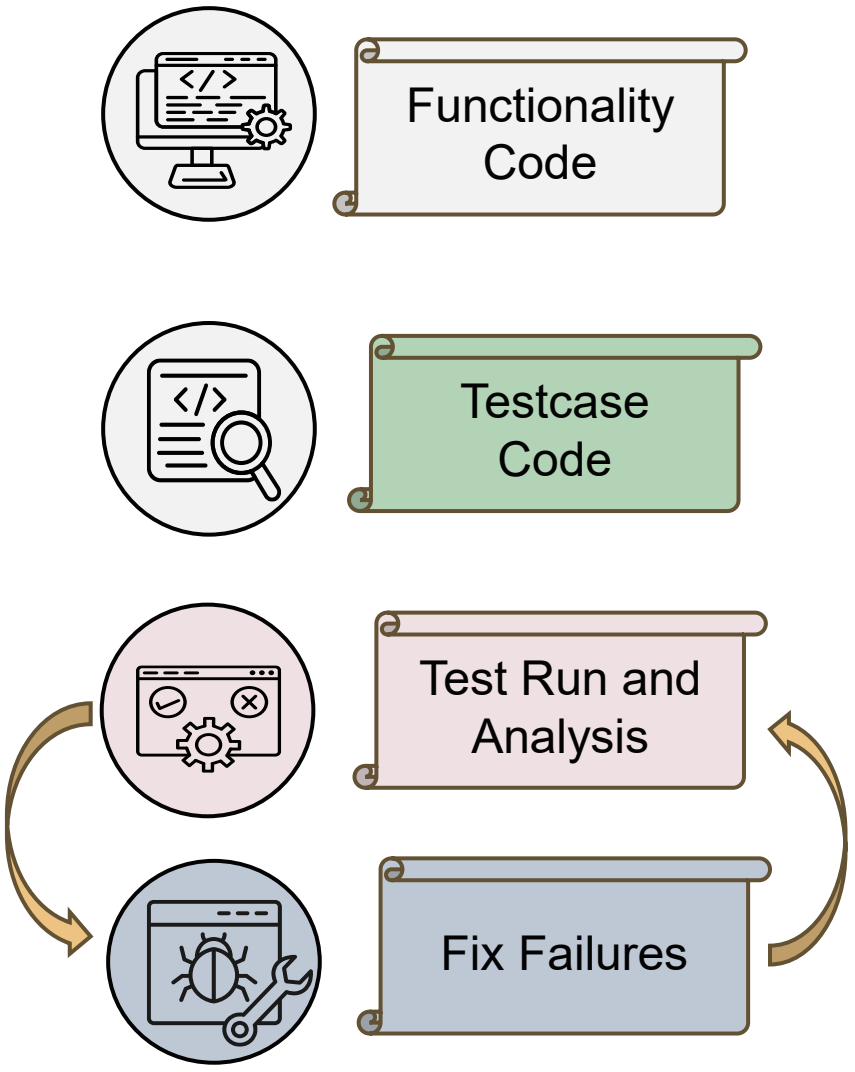


Code

Generate code for IP functionality, testcases and fix if test fails

90min

Code – Add Functionality To Stub Code



Demo – Code Phase

Code phase – Snapshots (1/2)


```
● Tasks · Functionality-Artisan
  1 completed · 1 in progress · 11 pending
  ✓ ~~Analyze existing code structure and architecture~~
  ● Searching and extracting register callback requirements from hardware behaviour JSON
  ○ Implement CFG register write callback with configuration lock
  ○ Implement CMD register write callback with command processing
  ○ Implement STATUS register read callback
  ○ Implement KEY register write callbacks with write protection
  ○ Implement DIGEST register write callbacks with write protection
  ○ Implement MSG_LENGTH register write callbacks with write protection
  ○ Implement MSG_FIFO write callback with backpressure modeling
  ○ Implement INTR_STATE, INTR_ENABLE, INTR_TEST callbacks
  ○ Implement WIPE_SECRET register write callback
  ○ Compile the project and fix any errors
  ○ Present implementation summary for approval
```

Subagent needs your input!

Context: Implemented Register-Based Transaction Interface functionality for HMAC IP with complete register callbacks, write protection, configuration lock, error detection, and TLM compliance.

Question: I have completed the SystemC TLM implementation for the HMAC IP Register-Based Transaction Interface (HMAC-FUNC-001). Below is a summary of what was implemented. Please review and approve.

Implementation Summary

Files Modified:

1. **/home/ashwinp/basavarajh/demo/feb_19/tvastaavp_dvcon/hmac/model/inc/hmac.h**
   - Added comprehensive internal state structure with:
     - Hash engine state (hmac_idle, msg_fifo_writable, operation_finalized)
     - FIFO management (fifo_depth, fifo_empty, fifo_full, 32-entry msg_fifo)
     - Message length tracking (msg_length_bits)
     - Internal key storage array (write-only KEY registers)
     - Interrupt enable tracking
   - Added helper function: set_error_and_interrupt()
2. **/home/ashwinp/basavarajh/demo/feb_19/tvastaavp_dvcon/hmac/model/src/hmac.cpp**
   - Enhanced constructor to initialize all internal state variables
   - Enhanced reset_handler() to properly reset all functionality state
3. **/home/ashwinp/basavarajh/demo/feb_19/tvastaavp_dvcon/hmac/model/src/hmac_register.cpp**
   - Implemented complete register callbacks with full hardware behavior

Register Callbacks Implemented:

1. INTR_STATE (Interrupt Status)

- W1C semantics: Write-1-to-clear for all interrupt bits
- Properly updates internal interrupt state and triggers driver updates

2. INTR_ENABLE (Interrupt Enable)

- Controls interrupt masking for all three sources
- Re-evaluates interrupt outputs when enables change

3. INTR_TEST (Interrupt Test)

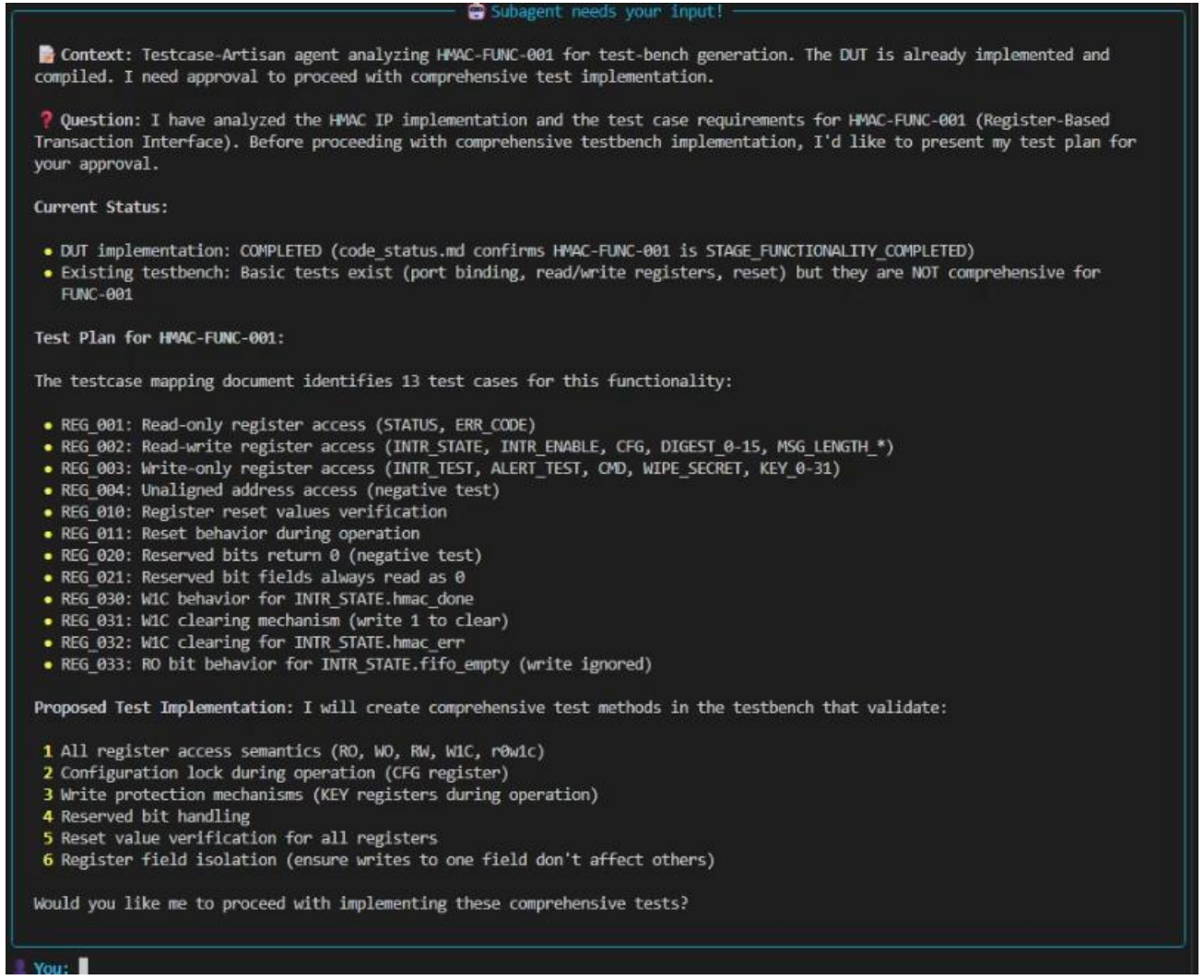
- Force-sets interrupt bits for software testing
- Respects enable masks

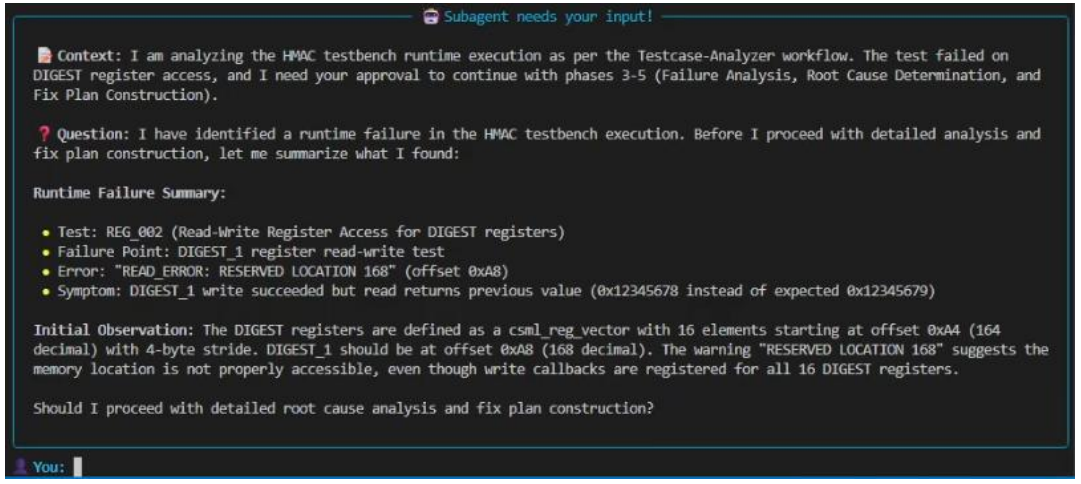
4. ALERT_TEST (Alert Test)

- Forces fatal_fault alert for testing

5. CFG (Configuration Register)

Code phase – Snapshots (2/2)





Impact & Roadmap

Tvastaa VP impact and roadmap

15min

Comparison with Vibe Coding

VIBE CODING

Problem Decomposition

- Manual task breakdown
- Manual plan review

Objective Implementation

- Continuous human guidance
- Observe closely (High Human Touch)

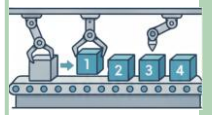
Feedback Interpretation

- Large prompts required
- Prompt engineering efforts

Context Management

- Manual context gathering
- Context varies per task (Manual Context Gathering)

Tvastaa VP AGENTIC WORKFLOW



Problem Decomposition

- Automated structured planning
- Pre-defined workflows

Objective Implementation

- Well-defined execution flow
- Human interference is low and well defined (Low Human Touch / High Autonomy)

Feedback Interpretation

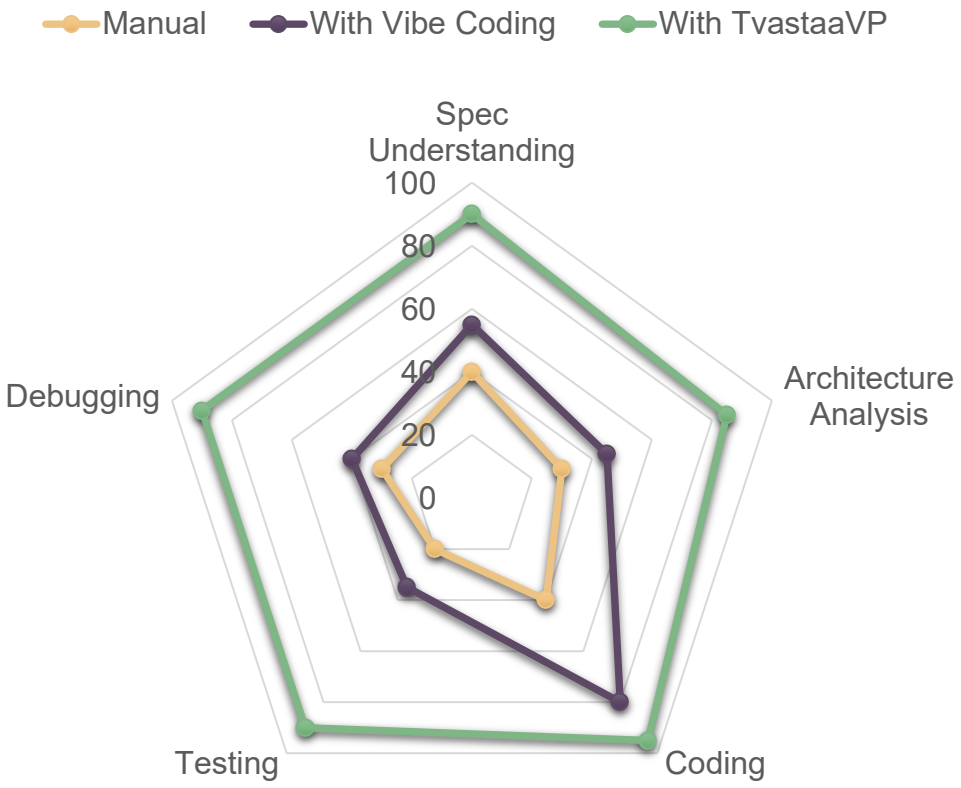
- Structured feedback handling
- Minimal or no prompt effort

Context Management

- Automated context retrieval
- Pre-configured integration (Automated Context Integration)

Tvastaa VP Agentic Workflows BEATS Vibe Coding

Same Engineer – Improved Workflow



Traditional Approach/Manual

- Manual spec study – gaps and assumptions
- Architecture quality depends on experience
- Hand-written SystemC, slow iterations
- Sparse testbenches and weak coverage
- Debug mostly reactive

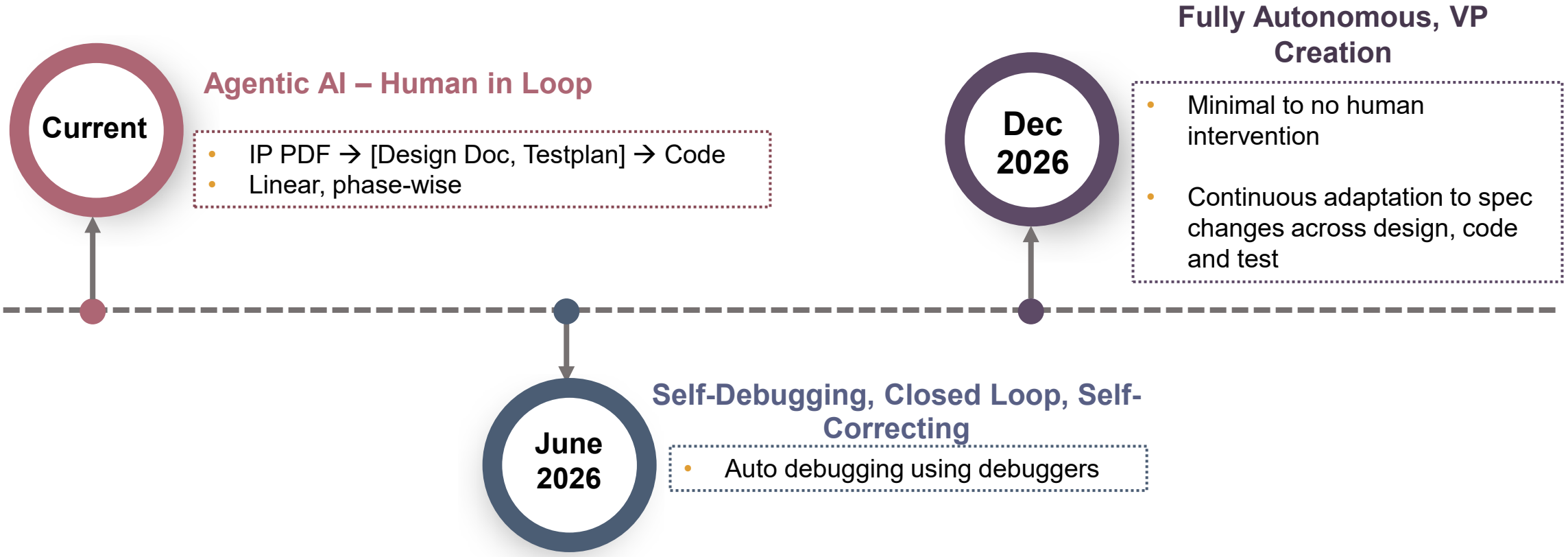
With Vibe Coding

- Repeated prompting to refine output
- Partial spec comprehension
- Code-first, architecture later
- Testbench mostly manual
- Debug driven by user feedback

With Tvastaa VP

- Guided spec understanding & extraction
- Assisted architecture and design decisions
- Structured SystemC generation
- Testbench created alongside model
- Faster debugging and fixes

Roadmap



Take Aways..

Tvastaa VP Value Proposition

- Faster spec-to-model turnaround
- Reduced dependence on SystemC human experts
- Reduced design and documentation efforts
- Consistent code quality across engineers
- Automated test generation and debugging
- Faster iteration and prototype availability

Thank You

www.vayavyalabs.com