Helix – the ASE Foundry

Intelligent, Integrated

Executive Summary

Helix unifies the ASE development lifecycle into an integrated, AI-powered platform that super charges engineer productivity. It replaces fragmented tools and workflows with a seamless, modern experience. ***With Helix, ASE will be able to super charge the concept-to-customer velocity while significantly reducing the cost.*** Like the double helix in DNA, Helix forms the intelligent, integrated backbone of modern ASE.

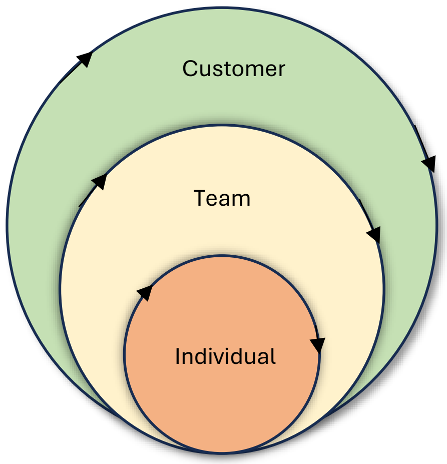
This document covers the pain points, user stores, roadmap, stakeholders, AI opportunities, risks, and success metrics.

Why?

Modern services development is hampered by fragmented tooling, duplicated workflows, and context-switching. Just as world-class products require world-class foundries, exceptional services demand a seamless, intelligent foundry as well. Helix is our bold response: an integrated, AI-powered platform that unifies the developer journey from concept to customer.

User Stories: Pain Points Across the SDLC

Project Helix addresses pain points across every stage of the individual, team, and customer workflows.



Individual - **Inner Loop** (code, test, debug).

Team - Middle Loop (reviews, integration).

Customer - **Outer Loop** (deploy, feedback).

The following user stories reflect the daily friction developers face due to fragmented tooling and disconnected workflows.

Individual Workflows - Inner Loop

1. **“As a new developer**, I spend hours setting up my environment and configuring tools for each repo, leading to lost onboarding time and inconsistent debugging experiences.”
2. **“As a senior engineer**, I find myself duplicating effort across multiple testing frameworks and linters because different teams use different standards.”
3. **“As a developer fixing bugs**, I switch constantly between logs, code, docs, and observability tools — losing focus and momentum.”

Team Workflows – Middle Loop

1. **“As a code reviewer**, I often receive pull requests without context or with inconsistent formats, making reviews slow and subjective.”
2. **“As a team lead**, I waste time reconciling CI results across multiple pipelines that aren't standardized across teams.”

Customer Workflows – Outer Loop

1. **“As a dev on-call**, I struggle to trace customer-facing issues back to recent code changes due to fragmented observability and monitoring tools.”
2. **“As a developer seeking feedback**, I rarely get actionable insights from real user behavior or post-release issues — feedback loops feel broken or delayed.”

Schedule & Milestones

We will execute Project Helix in ‘**Crawl–Walk–Run**’ phases:

Crawl (0–3 months)

- Journey mapping - Inventory existing tools, gaps, and duplication
- Launch stakeholder advisory group
- Deliver a draft tool consolidation plan

Walk (3–9 months)

- Consolidate core tools - editor plugins, debugging, unit tests
- Integrate AI-based static/dynamic analysis
- Measure early productivity gains

Run (9–18 months)

- Launch AI-powered insights engine (developer health, code quality)
- Sunset legacy systems and measure total cost savings
- Celebrate developer success stories and showcase gains

Stakeholder Engagement & Program Kickoff

1. **Engage key stakeholders**: Leaders, DevEx teams, IT, compliance.
2. **Steering committee**: Engineering, Security, PM, DevOps.
3. **Developer listening sessions**: Gather raw feedback across orgs.
4. **Kickoff summit**: Align Objectives and Key Results.

Top Risks and Mitigations

Risk	Mitigation
Tool resistance from teams	Offer migration support and early wins
Lack of leadership buy-in	Use Objectives and Key Results to tie to business outcomes
Scope creep	Phased roadmap guardrails and steering committee approvals
Legacy dependencies	Phased deprecation, wrappers for transition
Security/compliance	Integrate early with security, set policy baselines

---

Getting Endorsement

- **Build the narrative**: "It's not just a tools cleanup — it's a developer experience revolution."
- **Show cost of inaction**: Lost productivity, increased defects, talent attrition.
- **Secure executive champion**: Senior Vice President level sponsor.
- **Use data**: Developer surveys, tool usage metrics, time-to-release delays.
- **Highlight AI opportunity**: "AI to enhance every stage of the SDLC."

AI: Where It Adds Superpowers

Stage	Problem	AI Opportunity
Coding	Boilerplate, slow ramp-up	AI pair programming, code suggestion
Debugging	Time spent on root causes	AI log analysis, intelligent breakpoints
Static analysis	Rule-based tools, noisy results	AI-prioritized findings
Code reviews	Delays, low quality	AI review, comment summarization
Testing	Test gaps, flaky tests	AI-generated test cases, anomaly detection
Integration	Pipeline failures	Predictive failure detection
Deployment	Manual steps, rollback fear	AI-based canary rollout planning
Monitoring	Reactive issue discovery	AI-powered observability, user sentiment

---

OKR – Objectives and Key Results (6–9 months)

Objective

An intelligent and integrated services engineering platform that amplifies engineer productivity from concept to customer.

Key Results

1. Reduce average context-switching time by 30% as measured by internal telemetry.
2. Migrate 60% of teams to consolidated inner loop toolset (editor, debugger, unit testing).
3. Improve engineer satisfaction score by 20% (survey baseline vs. 6-months).

Measuring Progress: On-Track vs. Off-Track

On-Track Indicators

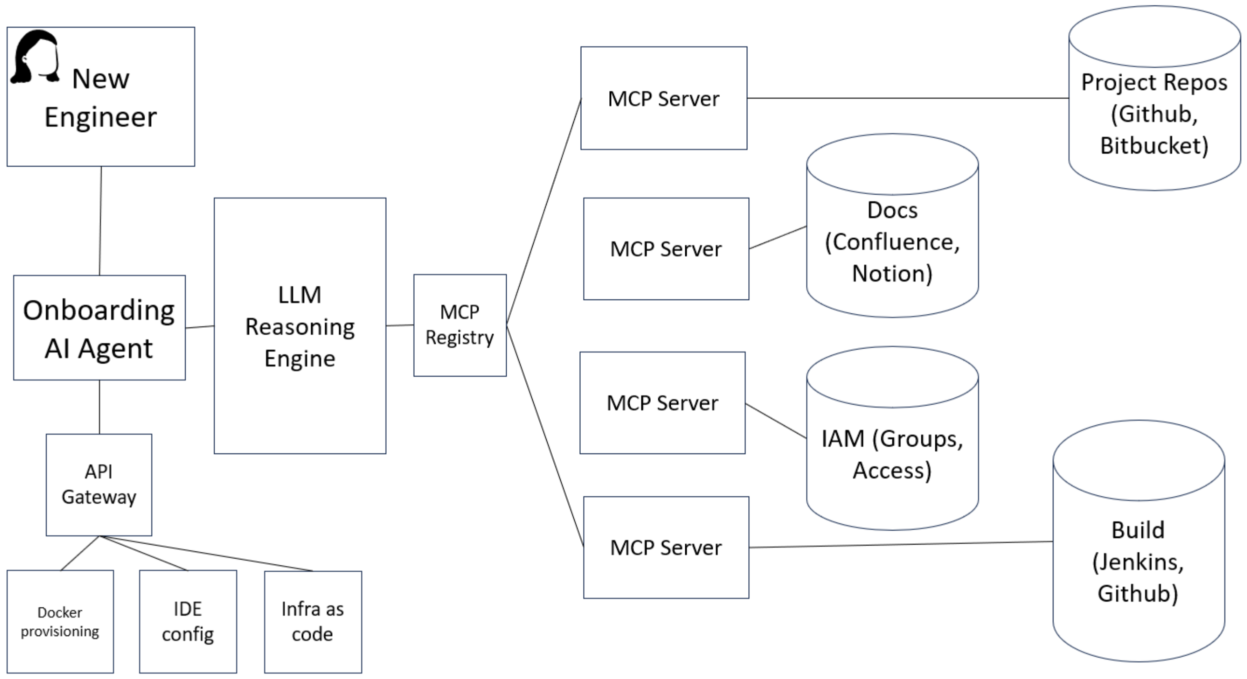
- Monthly adoption growth
- Milestone delivery within 10% schedule variance

Off-Track Indicators

- No reduction in tool fragmentation
- AI tools underused or bypassed

Design Example – AI Agent for New Engineer Onboarding

User story: *"As a new engineer, I spend hours setting up my environment and configuring tools, leading to lost onboarding time and inconsistent debugging experiences."*



The diagram illustrates the architecture of an **AI-powered onboarding agent** designed to automate environment setup and access provisioning for new developers. The agent uses **Model Context Protocol (MCP)** servers since MCP is fast emerging as a standard for LLM to AI Agent interaction.

Flow Summary

1. **New Engineer Joins and communicates in natural language with an AI Agent**
   A new engineer initiates onboarding via an **AI agent** that understands their team, role, and project context.
2. **Onboarding AI Agent communicates with an LLM Reasoning Engine**
   The onboarding agent is backed by a **Large Language Model (LLM) Reasoning Engine**, which consults a **MCP Registry** to interpret project-specific onboarding instructions encoded using the **Model Context Protocol**.
3. **MCP Servers get data and permissible actions in an LLM friendly format**
   Based on the MCP registry, the agent identifies appropriate **MCP Servers**. The MCP servers are responsible for getting data and actions metadata in an LLM friendly format.
4. **MCP Servers do the following**

- **Docs**: Connect to relevant documentation (e.g. Confluence)
- **Identity and Access**: Data and actions for user and group management
- **Repos**: Information and actions for GitHub or Bitbucket repositories
- **Build**: Data and actions for Jenkins pipelines

5. **API Gateway identifies the right API to take provisioning actions** e.g.
    - **Docker configuration** (via container templates)
    - **IDE configurations** (e.g., extensions, settings, language servers)
    - **Infrastructure as Code** (e.g., Terraform modules to provision resources)
    - **Secrets Management** (If secure credentials are required (e.g., GitHub tokens

Dependencies

Component	Dependencies	Notes
New Engineer	Portal or trigger (e.g., Slack, web app, CLI)	Human interaction entry point
AI Agent	LLM API, API Gateway, MCP Registry	Core orchestrator
LLM	Access to training prompts, MCP schema understanding, Tool plugin API	LLM needs onboarding flows and ability to call APIs
MCP Registry	YAML/JSON spec storage (e.g., S3, Git repo, internal DB), Version control	Centralized store for context-driven specifications
MCP Servers	IAM (e.g., Okta, Azure AD), Docs (Confluence), CI/CD systems, Version Control (GitHub)	Distributed execution units that fulfill parts of onboarding flows
API Gateway	Docker API, IDE Config, Infra-as-Code	Action execution
Docker	Docker Registry, Image templates	Spins up dev containers
IDE Config	VSCode/JetBrains APIs, Plugin templates	Automates dev environment setup
Infra as Code	Terraform, Cloud provider (OCI, AWS, etc.)	Provision cloud resources
IAM (Groups)	Identity Provider APIs (Okta, AD, etc.)	Grants access to systems
Build Systems	CI/CD platforms (Jenkins), Job templates	Onboards to pipelines and alerts

Let’s get to work!

World class products are engineered in world class factories. Helix is not just a tooling cleanup — it's a world class, intelligent, integrated foundry for developing world class services. Let’s go!