BREAKING UP THE MONOLITH

MARTIN HOLT

CADEC 2020.01.23 & 2020.01.29 | CALLISTAENTERPRISE.SE

CALLISTA

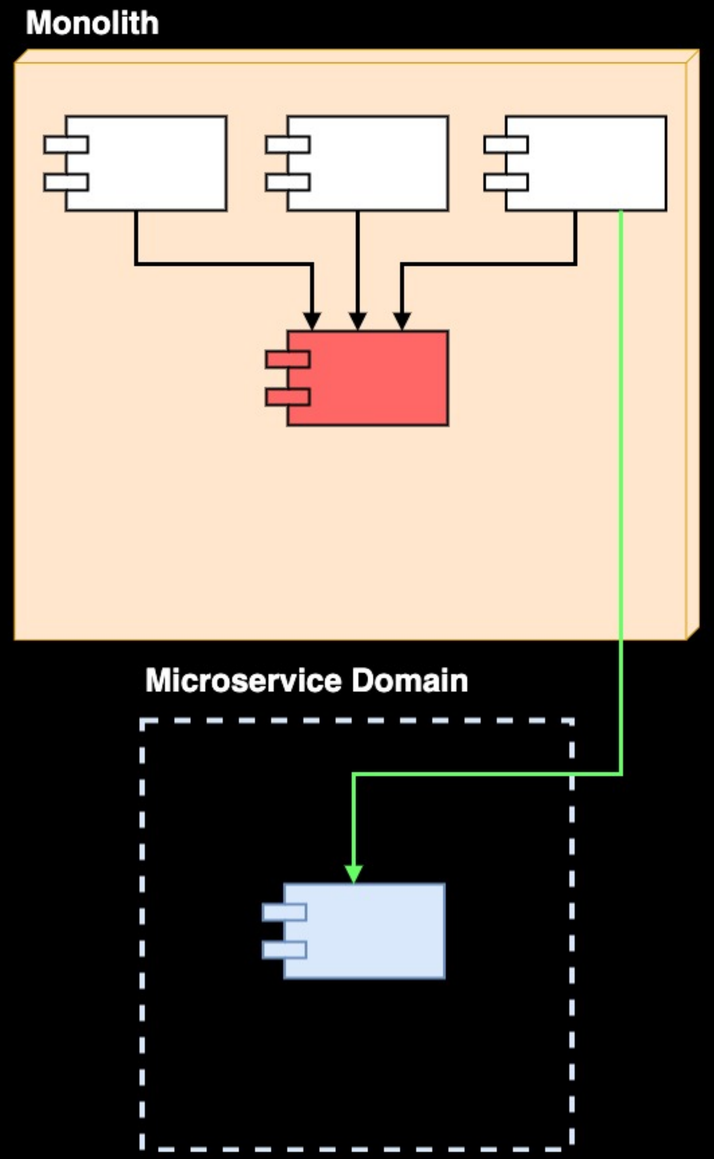
INTRODUCTION

- Monolith Perceived as:
  - Complex - Difficult to Make Significant Change
  - Opaque - Little Insight into Ongoing Processes
  - Slow - Long Release Cycles
- How Will Migrating To Microservices Affect This?
- Presenting Two Case Studies:
  - Extracting A Feature
  - The Minimal Viable Product

CASE STUDY: EXTRACT A FEATURE

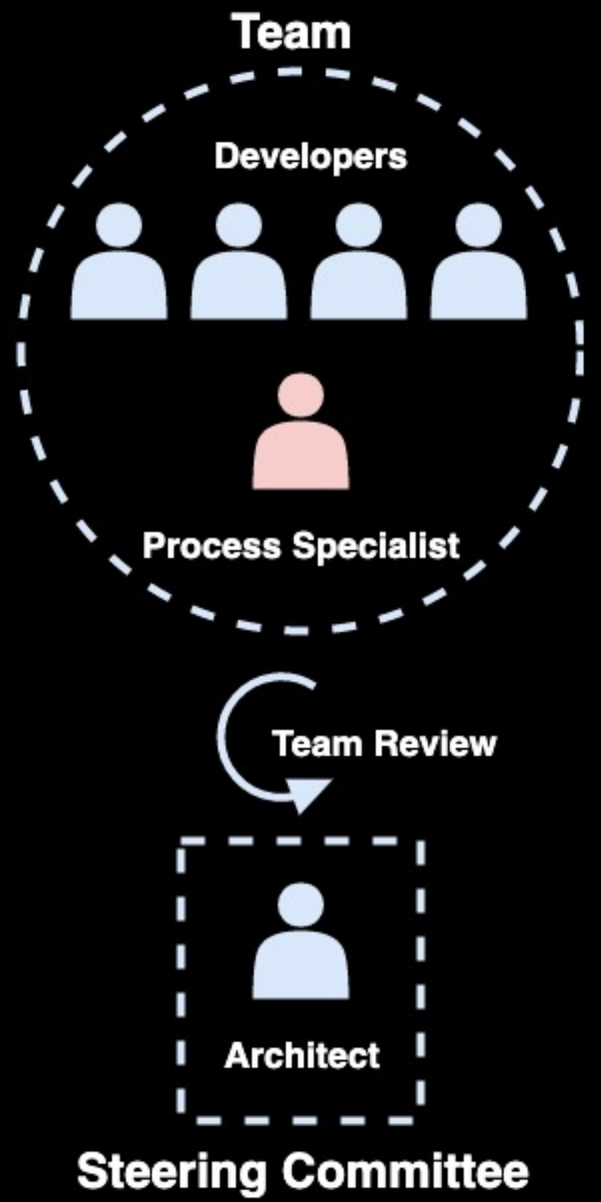
CASE STUDY: EXTRACT A FEATURE

- The Monolith Feature
  - Limited Functionality
  - Difficult to Configure
  - Unmonitored
  - Tied To Quarterly Release Cycles
- Why Microservices?
  - Provide Richer Features
  - …Deliver Faster
  - …Deliver Frequently
  - Want to Experiment



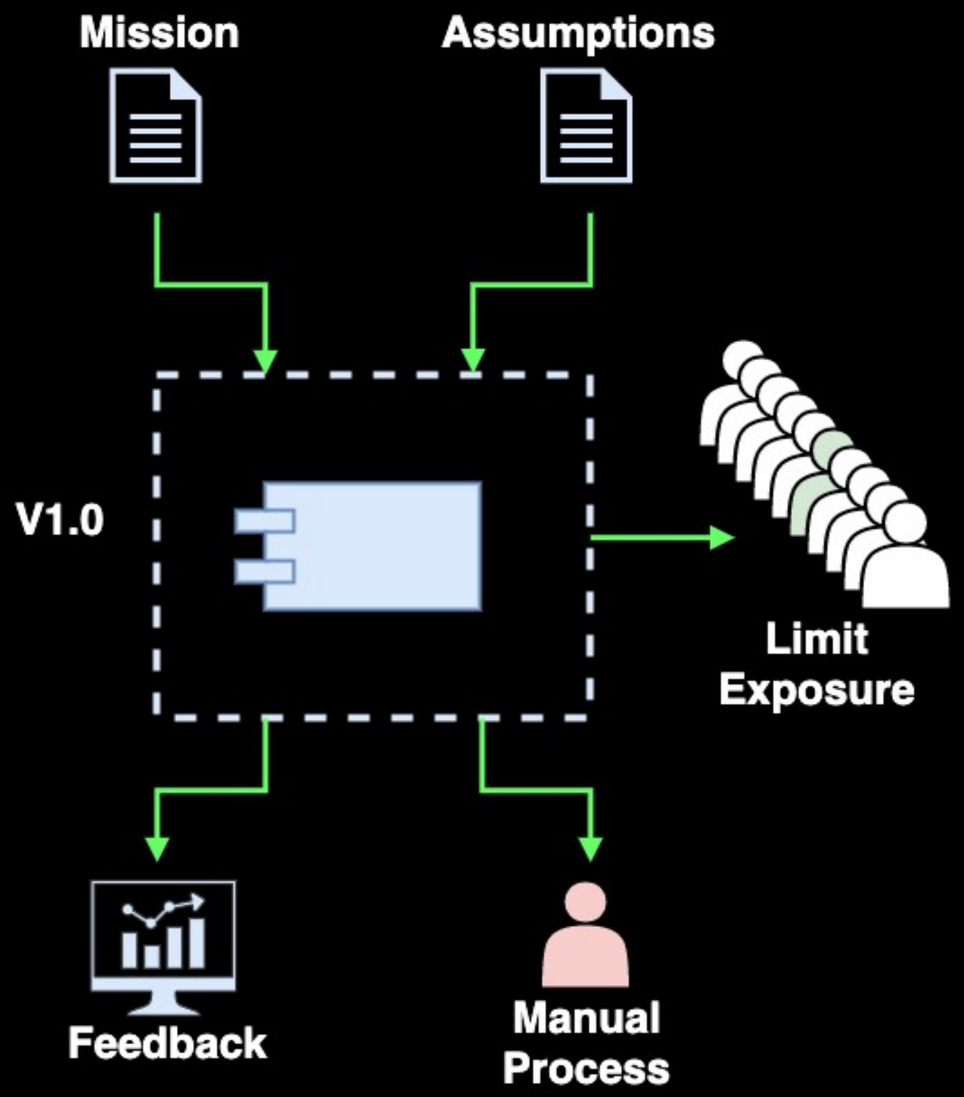
CASE STUDY: EXTRACT A FEATURE - THE TEAM

- Developer Heavy Team
  - Autonomous
  - Short Iterations
  - Time-boxed
- Steering Committee
  - Periodic Review
  - Initially a Single Architect



CASE STUDY: EXTRACT A FEATURE - PREPARATION

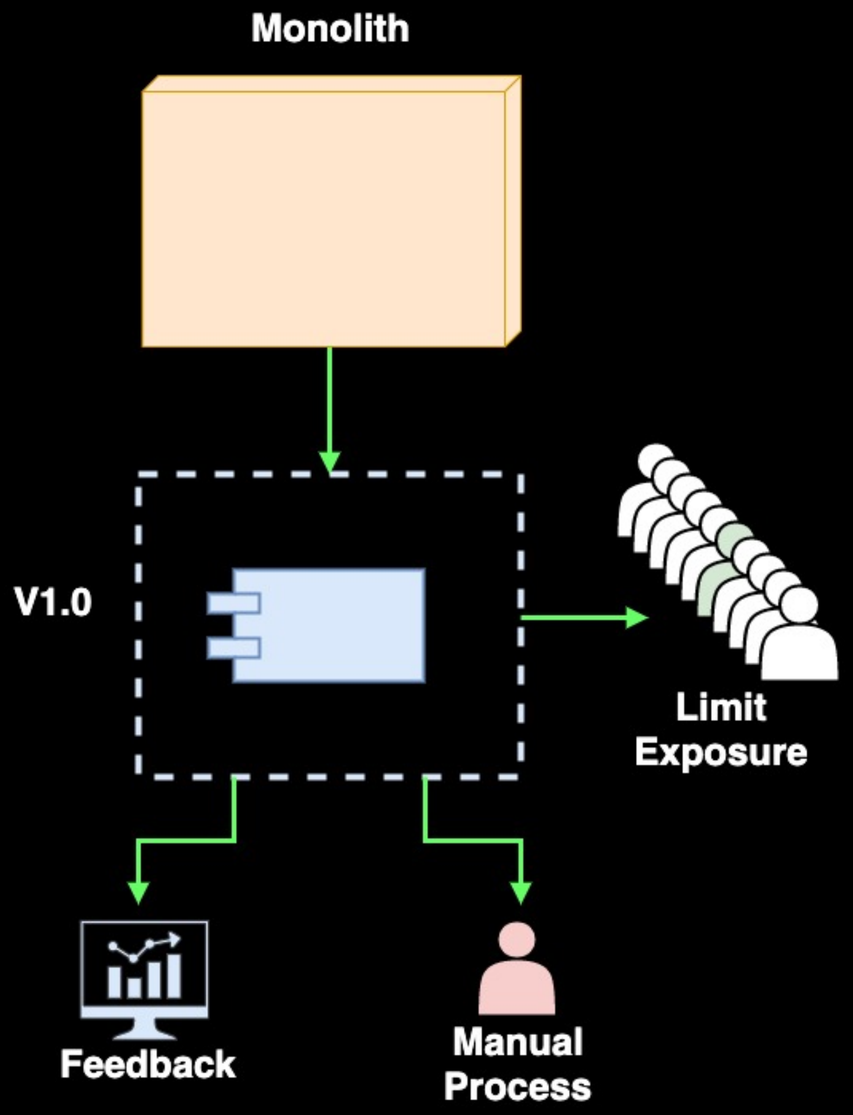
- Set The Scope:
  - Mission Statement
  - Document Assumptions
  - Identify Shortcuts
  - Reduce Risk
    - » Limit Exposure
    - » Manual Controls
    - » Backup Plan
  - Identify Feedback Loops
    - » How Do We Know This Is Working?



A Few Iterations Later...

CASE STUDY: EXTRACT A FEATURE - DELIVERY

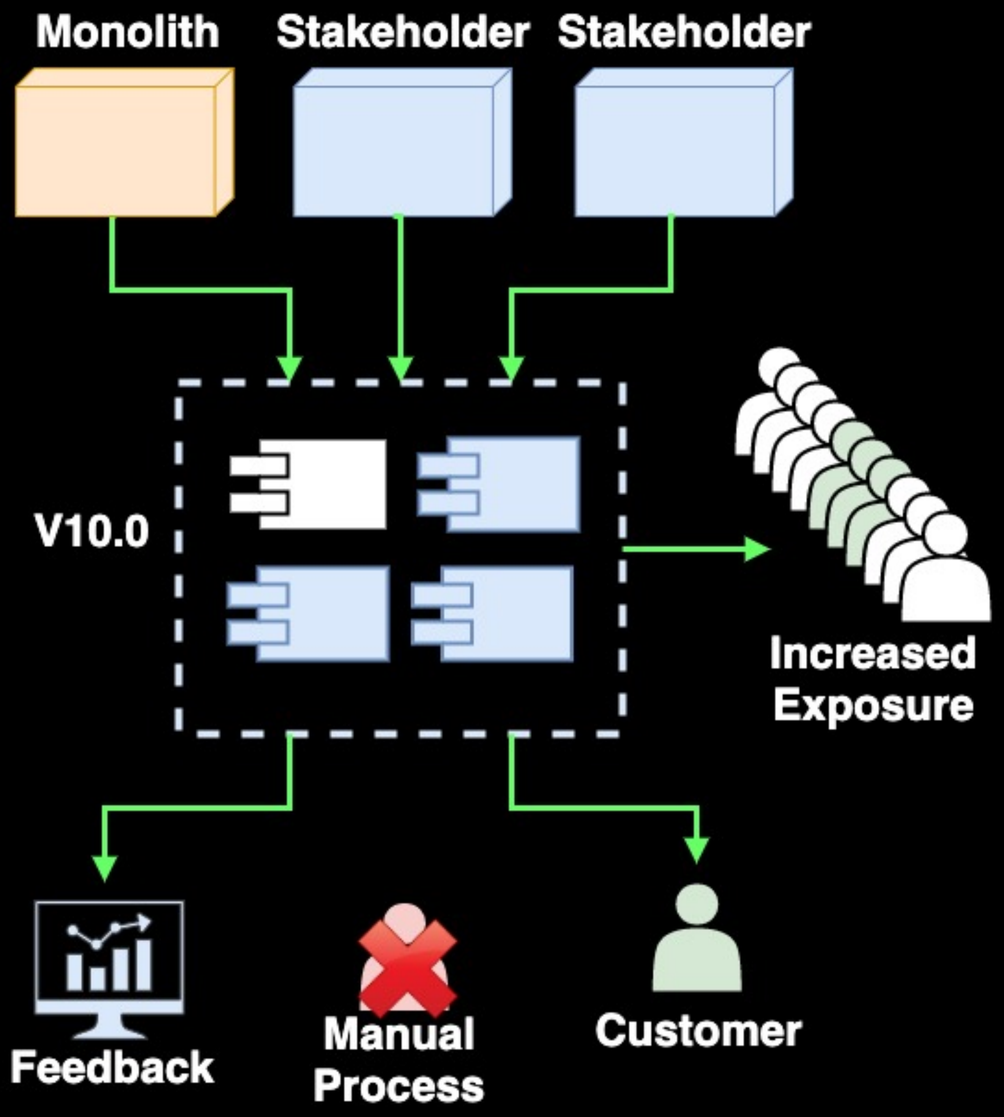
- Service Is Live In Record Time
- Bugs Are Found And Fixed Quickly
- Learning From Feedback
  - Metrics
  - Manual Controls
  - Steering Committee Review
- Time For Next Iteration!



A Few Iterations Later...

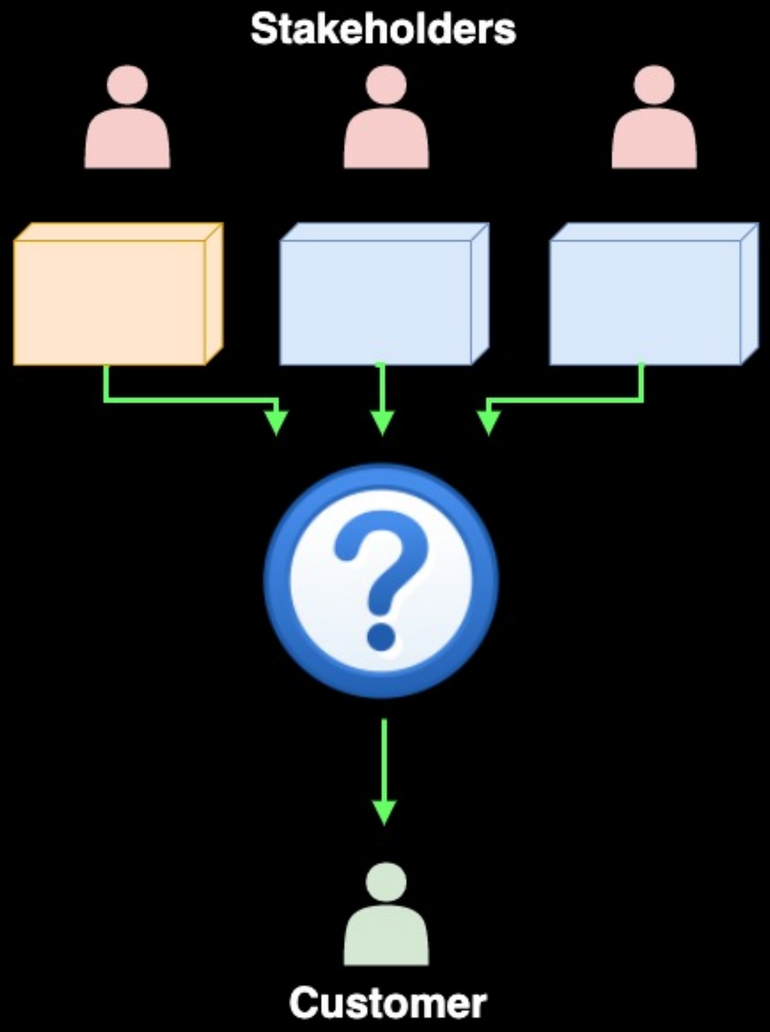
CASE STUDY: EXTRACT A FEATURE - MATURING THE DOMAIN

- Feature Rich Environment
  - Domain Complexity Increased
- New Stakeholders
  - Increased Volumes
- Consolidation
  - Removed Manual Processes
  - Increased Exposure
  - Dealt with Tech Debt



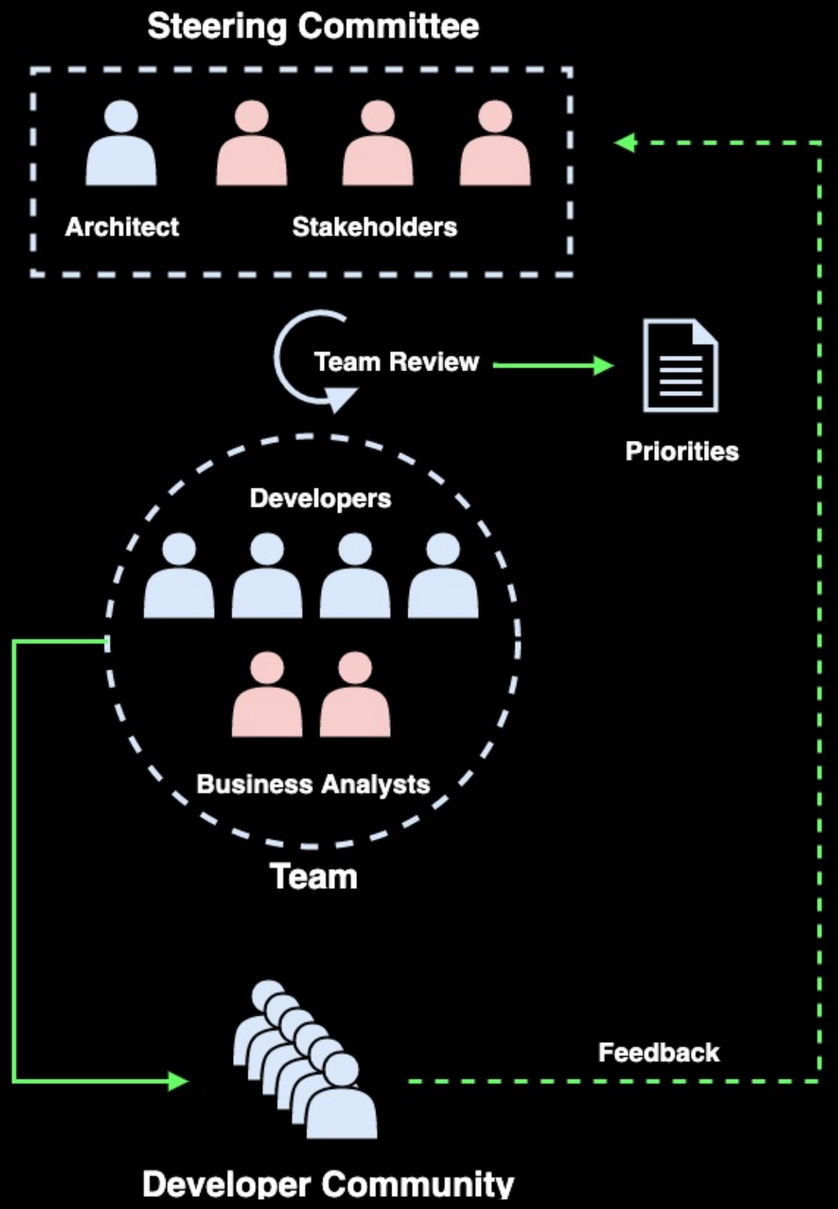
CASE STUDY: EXTRACT A FEATURE - GROWING PAINS

- The Forgotten Feature
  - Development Often UI Driven
  - Perceived as “Opaque”
- Infrastructure
  - “It Should Just Work”
  - Harder to Gain (Good) Attention
  - Perceived as “Complex”
- Multiple Stakeholders
  - Competing Priorities
  - Perceived as “Slow”



CASE STUDY: EXTRACT A FEATURE - REMEDIES

- Expand The Steering Committee
  - Invite Stakeholders
  - Prioritise Together
  - Highlight New Initiatives
- Extend The Team
  - Permanent Business Representation
- Engage With Developers
  - “Brown Bags” For Presence
  - User Groups to Identify Requirements
  - Allow Contributions
- All Remedies Driven By Team
  - Required Significant Time Investment



SUMMARY

- Extract A Feature To Microservices
  - Rapid Initial Success
  - Low Initial Risk
  - Fits Into Existing Organisational Structures
  - Solution Can Mature Iteratively
- Maturity Exposes Problems
  - Similar To Monolith (Complex, Opaque, Slow)
- Engagement Required:
  - With Stakeholders
  - With Developer Community
  - …but is it Sustainable?

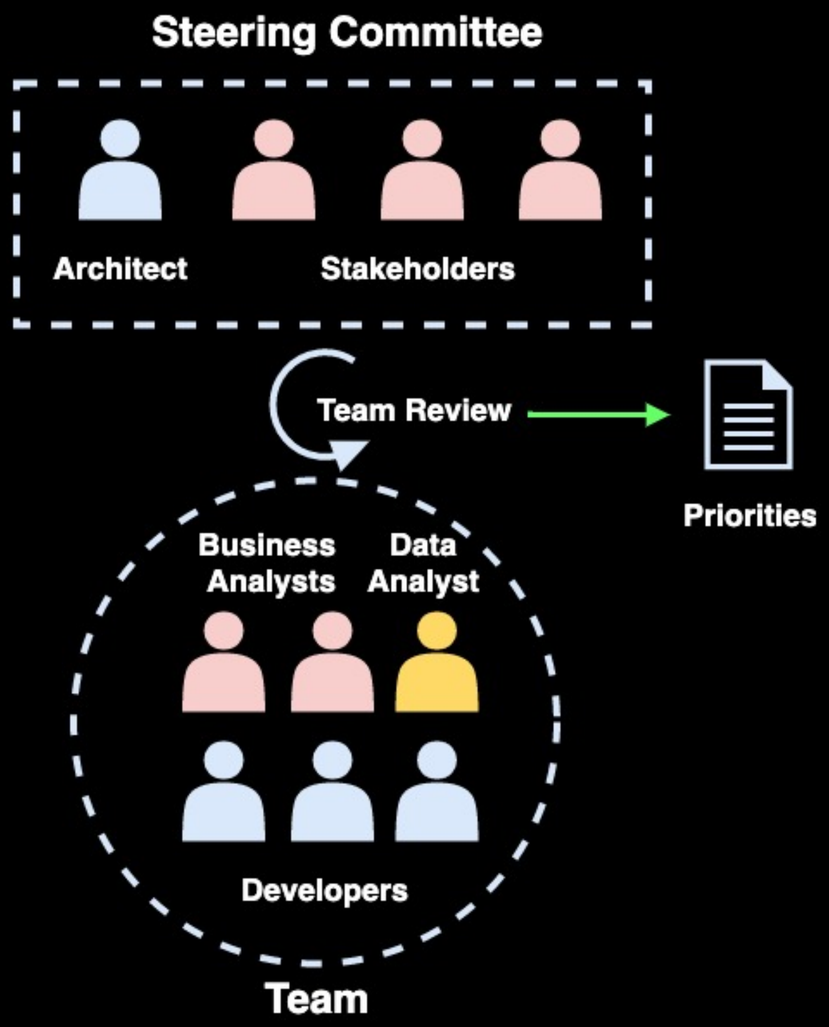
CASE STUDY: MINIMAL VIABLE PRODUCT

CASE STUDY: MINIMAL VIABLE PRODUCT

- Market Opportunity Identified
  - Need to React Quickly
  - Product Still Maturing
  - High Level of Uncertainty
  - Unwilling to Invest Heavily Upfront
  - Constraints on the Monolith
- Solution
  - A Minimal Viable Product
  - …in a Microservice Environment

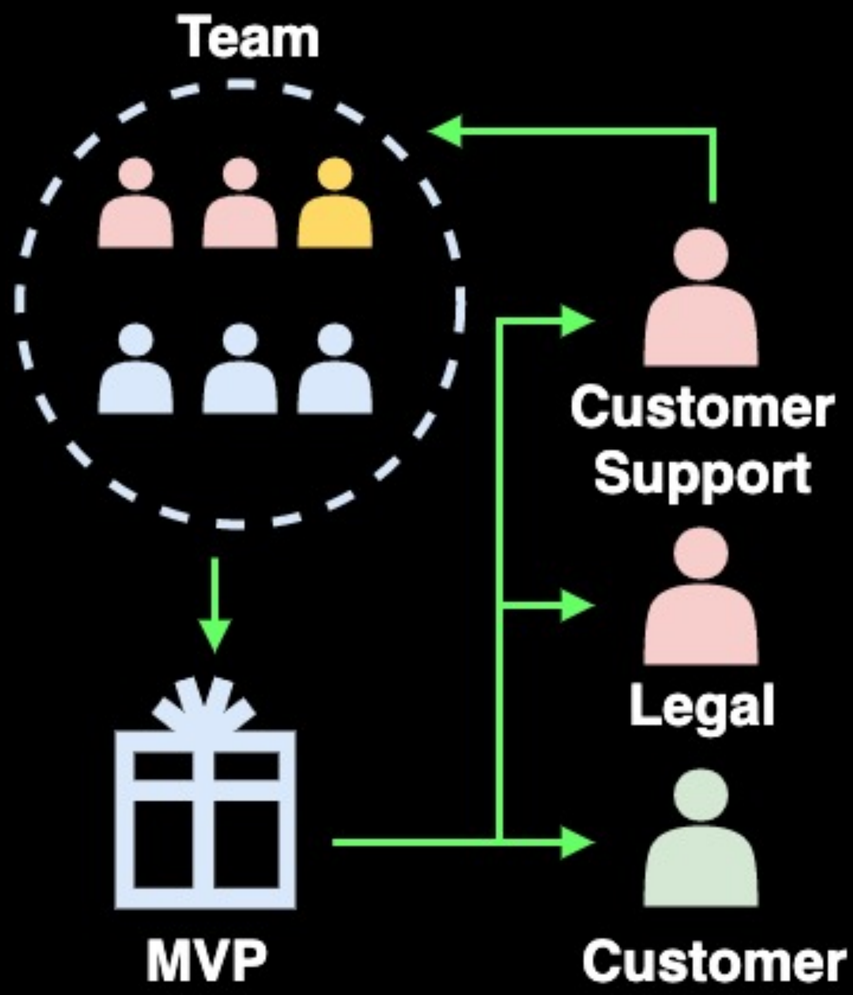
CASE STUDY: MINIMAL VIABLE PRODUCT - THE TEAM

- Stakeholder Initiated Team
  - Clear Business Presence and Goals
- Data Driven
  - Metrics Identified Early (volumes, revenue streams)
  - Metrics Are Measurable
- Steering Committee
  - Mostly Stakeholders
  - Regulate Direction of Product



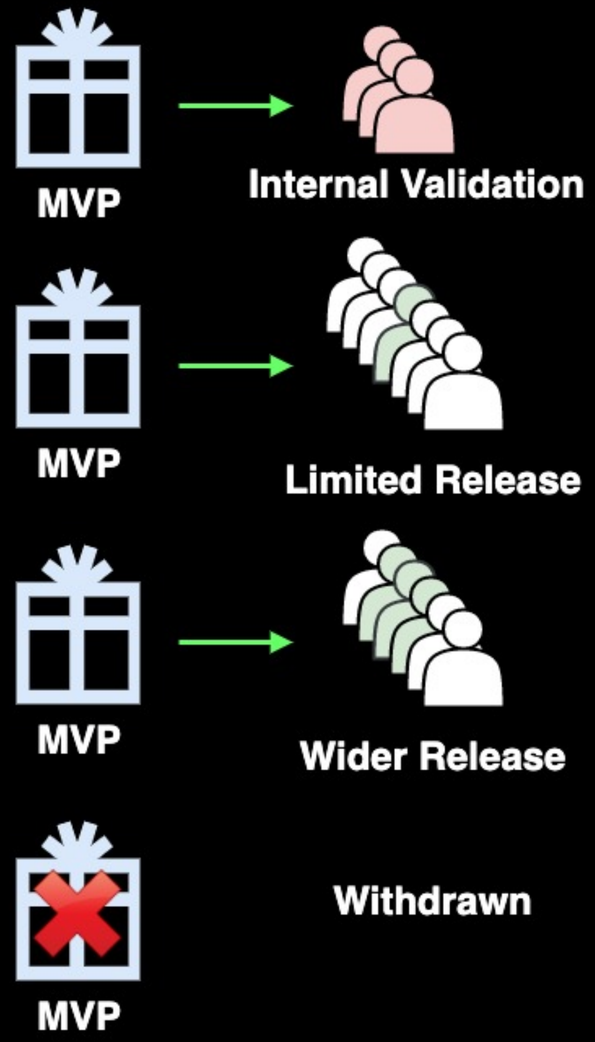
CASE STUDY: MINIMAL VIABLE PRODUCT - DEFINE

- What is Smallest Deliverable That:
  - Makes Sense to the Customer
  - Will Fit In The Organisation
  - Can be Maintained by the Team
  - Is Compliant
- Iterate And Gather Feedback



CASE STUDY: MINIMAL VIABLE PRODUCT - MANAGE RISK

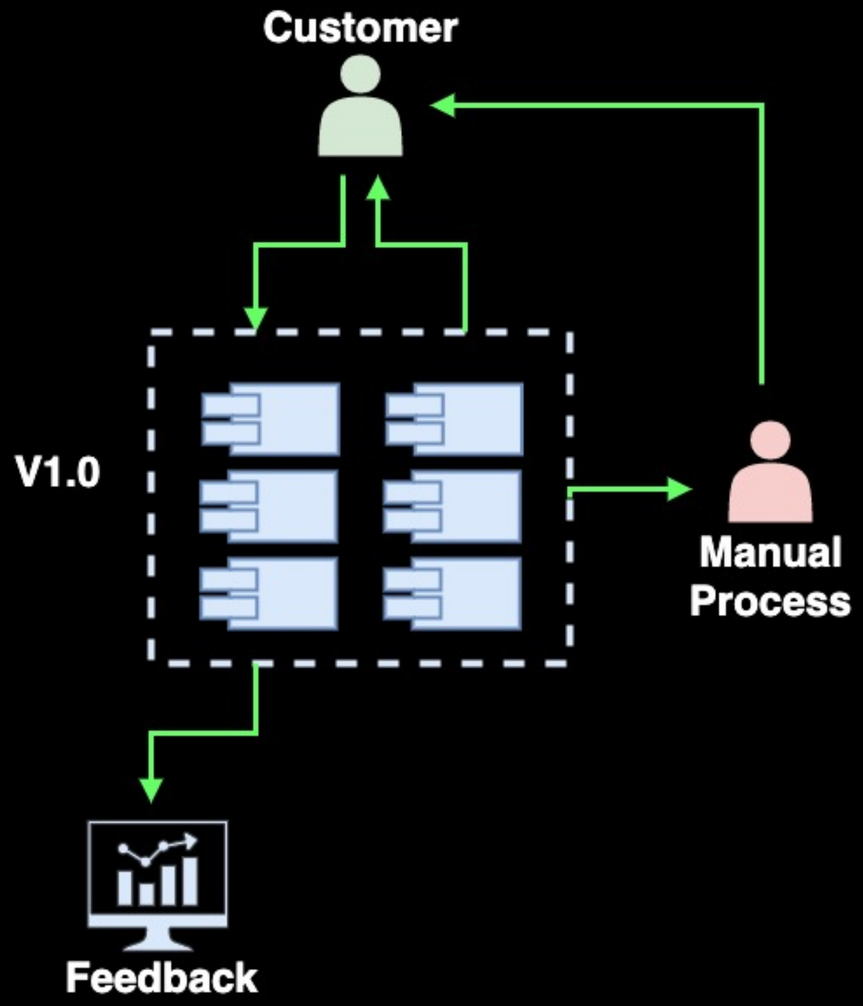
- Limited Initial Exposure
  - Internal Validation Phase
  - Gentle Rollout To Customers
- Limited Initial Commitment
  - May Not Sell as Expected
  - May Trigger Incidents
  - May Be a Target for Fraud



A Few Iterations Later...

CASE STUDY: MINIMAL VIABLE PRODUCT - DELIVERY

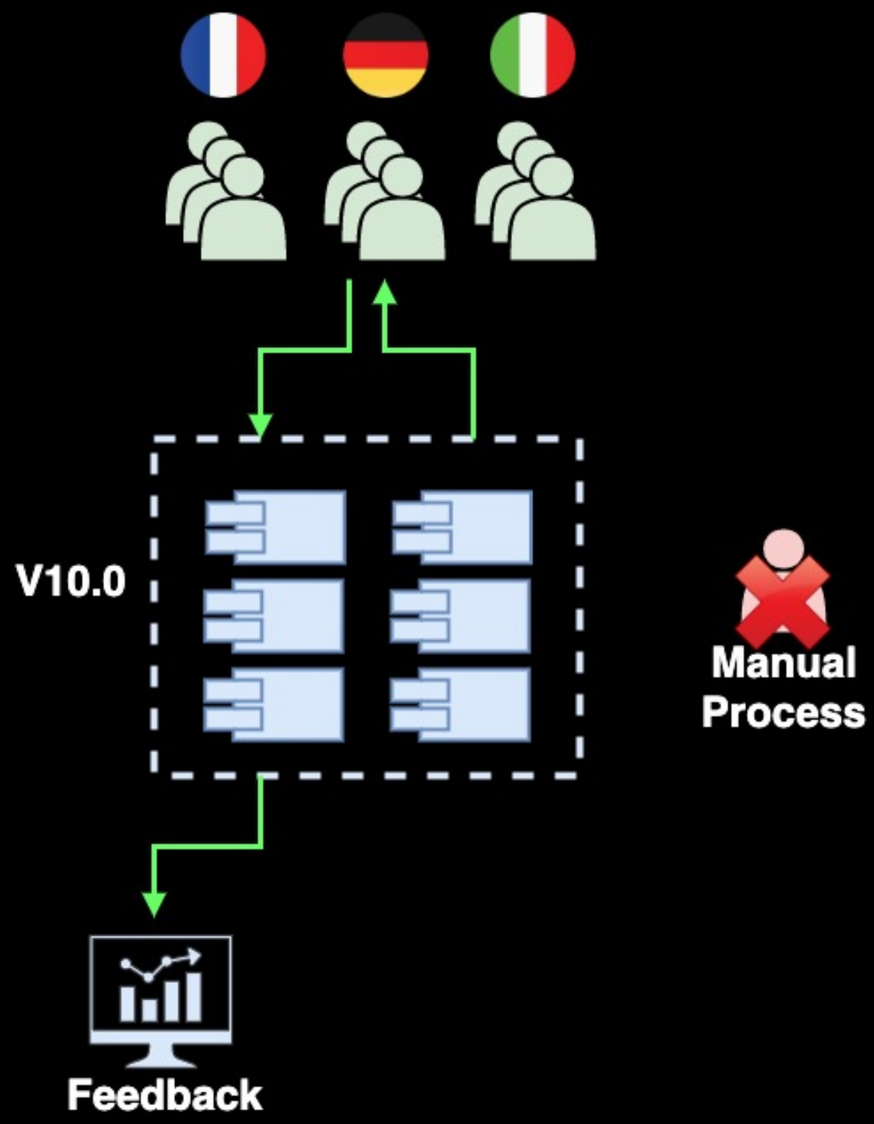
- Service Is Live!
  - Customers Are Flooding In…
  - …But In A Controlled Manner
- Bugs Are Found And Fixed Quickly
- Learning From Feedback
  - From Metrics
  - From Customer Feedback
  - From Other Teams
- Time For Next Iteration!



A Few Iterations Later...

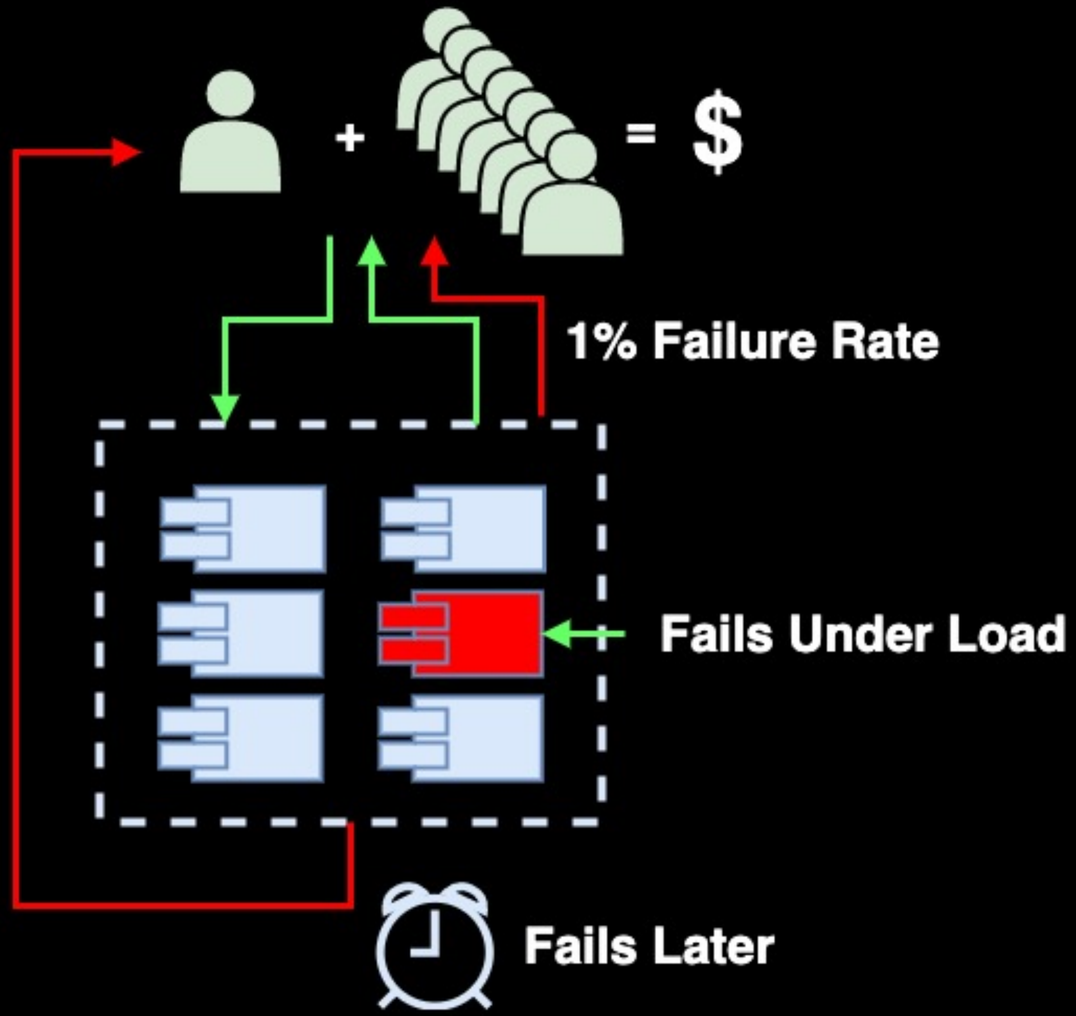
CASE STUDY: MINIMAL VIABLE PRODUCT - MATURING THE PRODUCT

- Enrich
  - Learn From Feedback
- Expand
  - Increase Exposure In Existing Markets
  - Investigate New Markets
- Consolidate
  - Dealt with Tech Debt



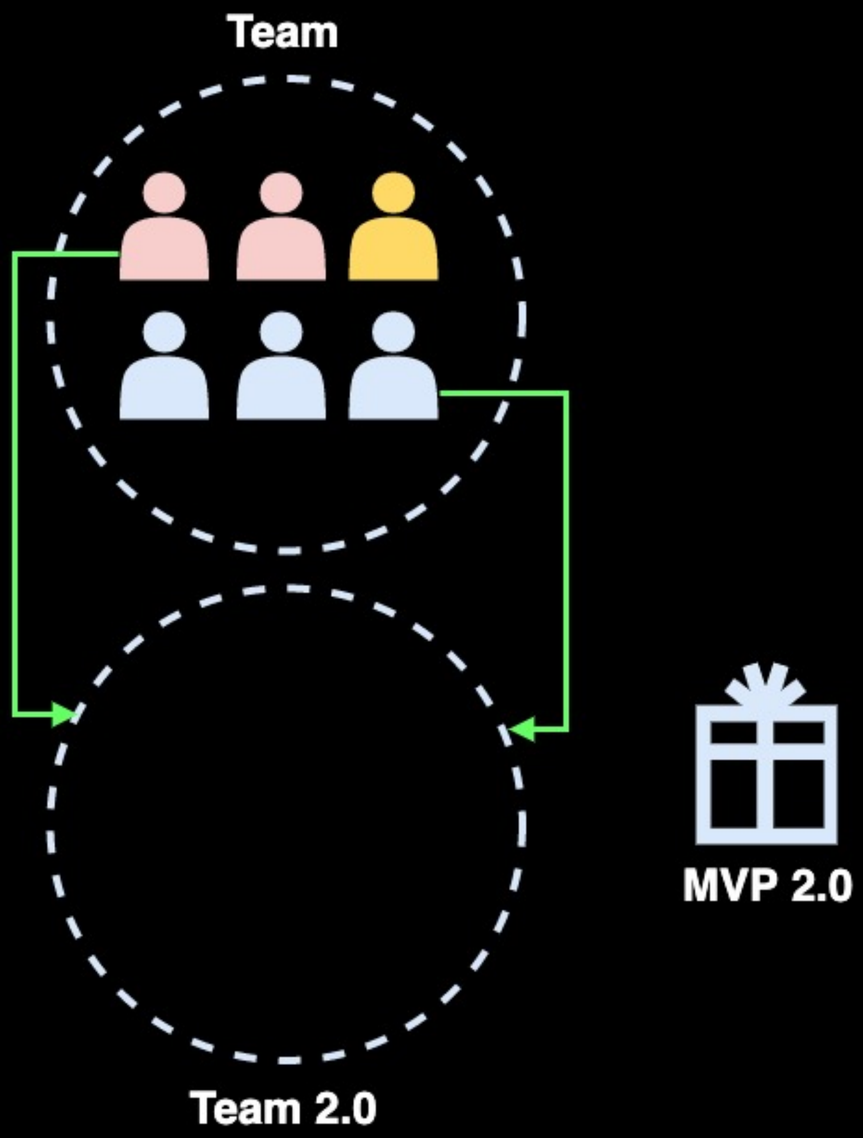
CASE STUDY: MINIMAL VIABLE PRODUCT - GROWING PAINS

- Success Breeds Success
  - Pressure to Add Volume
  - Relatively Easy to Realise
- Consequences:
  - Low Frequency Problems More Obvious
  - Stressed Services Behave Unpredictably
  - Effects May Not Be Immediately Visible
- Often Impossible To Reverse An Expansion
- Resolutions
  - Expand In Cycles
  - Be Open With Incidents And Failures



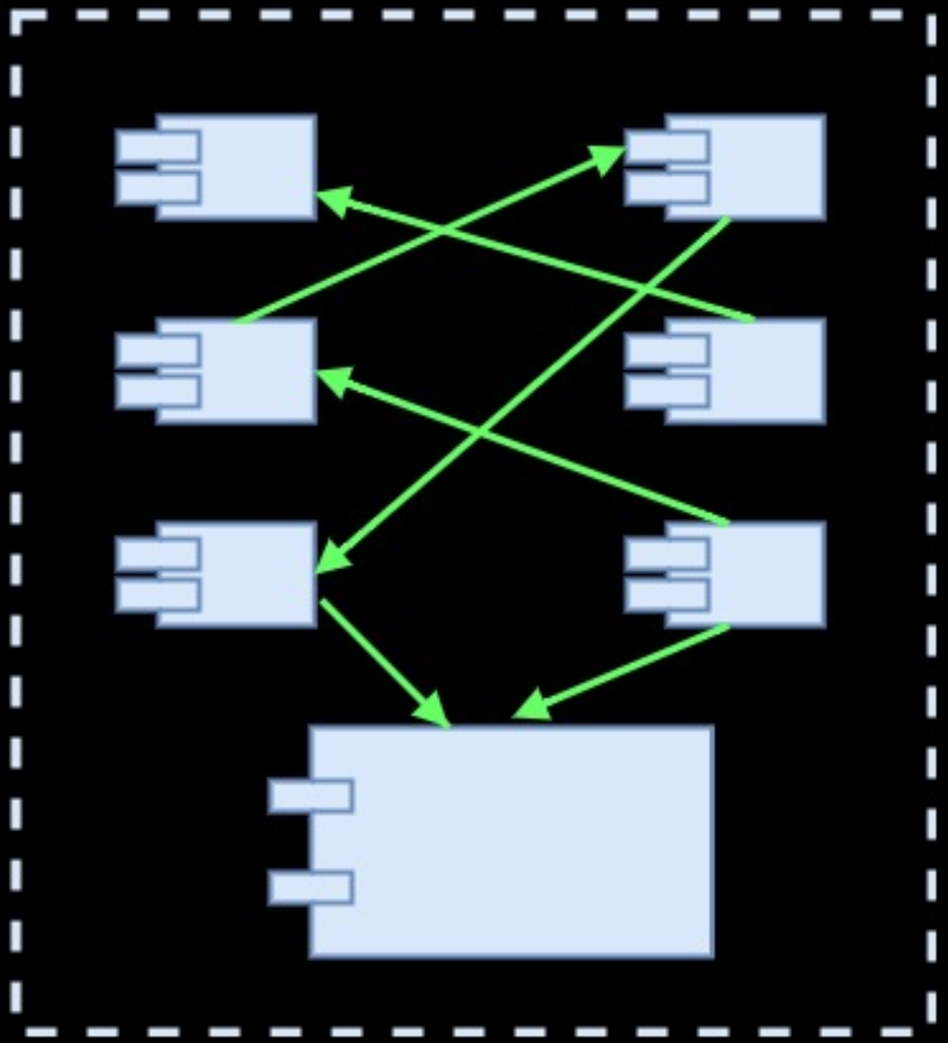
CASE STUDY: MINIMAL VIABLE PRODUCT - GROWING PAINS

- Success Inspires Others
  - Experience is Valued
  - Team Members Will Move On
- Resolutions
  - Spread the Knowledge
  - Documentation
  - Avoid Specialists
  - Plan For Transition



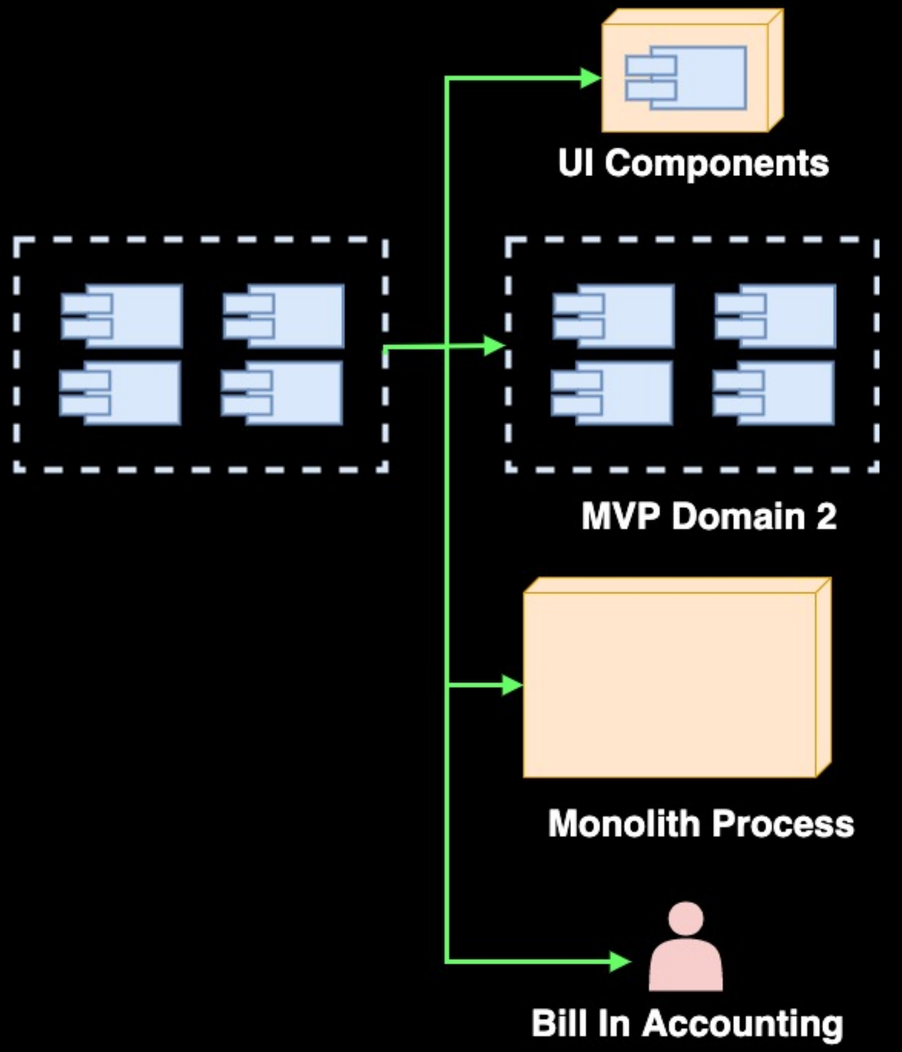
CASE STUDY: MINIMAL VIABLE PRODUCT - GROWING PAINS

- Design Compromise
  - Simplicity Allows For Rapid Change
  - Firefighting Blurs The Vision
- Consequences
  - Services With Unclear Roles
  - Complicated Call Chains
  - Differing Granularity In Services
- Creeping Complexity
  - Solution Becomes Opaque To The Team
- Resolutions
  - Engage Whole Team in Design Decisions
  - Document a Target Architecture
  - Allow Time To Refactor



CASE STUDY: MINIMAL VIABLE PRODUCT - GROWING PAINS

- In Reality:
  - MVP Delivered By Many (Temporary) Teams
  - Custom Code In Established Services
  - Dependency on Monolith
  - Manual Routines
- Consequences
  - MVP Decisions Made In Other Domains
  - Conflicting Priorities In Other Teams
  - Divergent Tech Stacks Hinder Ownership
  - Increasingly Longer To Deliver
- Resolutions
  - Assimilate Functions That Fit
  - Propose Architectural Change Where They Do Not



SUMMARY

- Microservices Enable Rapid Delivery Of MVPs
  - Relatively Low Risk
  - Iterative Expansion
  - Optimise Product using Feedback
  - In Parallel To Existing Organisational Structures
- Maturity Exposes Problems
  - Similar To Monolith - Complex, Opaque, Slow
  - Requires A Mature Team To Address
  - May Require Wider Architectural Change…
  - …Which May Need Organisational Change

WHAT ABOUT THE MONOLITH?

WHAT ABOUT THE MONOLITH?

- Started To Break Up The Monolith
  - …But Only Partially
  - …Maybe Never Completely
- Requires (Significant) Investment in the Monolith
  - Needs Competent and Motivated Teams
- Role of the Monolith Changed
  - Narrower Scope
  - Higher Specialisation
  - Challenges Perceptions

SUMMARY

- Microservices Migration is Powerful
  - Can Deliver More, Faster and Frequently
  - Strategies Available To Reduce Risk
- Best Practice Software Development Still Required
  - Avoid The Distributed Monolith
- Microservices Adoption Affects The Organisation
  - Retaining Agility is Challenging
  - Evolve The Organisation Using Team Feedback
- Monolith Still Has A Significant Role To Play