

# Legacy to Legendary

Sustainable patterns and practices for app  
modernisation

# Thank You to Our Sponsors and Hosts!



OWASP  
**NEW  
ZEALAND**  
owasp.org.nz



Waipapa  
Taumata Rau  
**University  
of Auckland**



# BASTION

SECURITY GROUP



**SECURE  
CODE  
WARRIOR**



**exabeam™**



**safeadvisory.**



**CyberCX**

**plexure**



**SECDIM**

**Without them, this conference couldn't happen.**

OWASP New Zealand

owasp.org.nz

- Some definitions
- Why it's important to know how to work with legacy code
- A couple of useful patterns for working with legacy
- Practices for working with legacy, including the people and teams
- The connection between working with legacy code and AI-assisted coding
- Conclusions and closing thoughts

**Overview and themes**

<https://www.npr.org/2022/06/29/1108739853/when-subaru-came-out-classic>

**What is legacy anyway?**

**IT LOVES CAMPING, DOGS AND  
LONG-TERM COMMITMENT.  
TOO BAD IT'S ONLY A CAR.**



What's of paramount importance is working software, especially working software that people use and ideally love (Agile Manifesto)

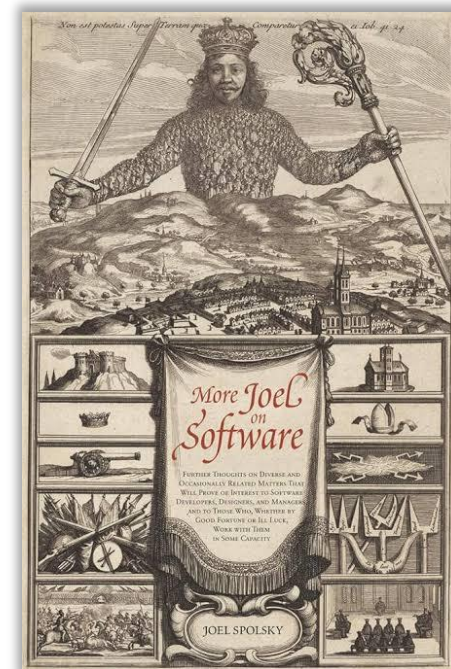
My definition:

**Any code that is in production**

Michael Feathers' definition:

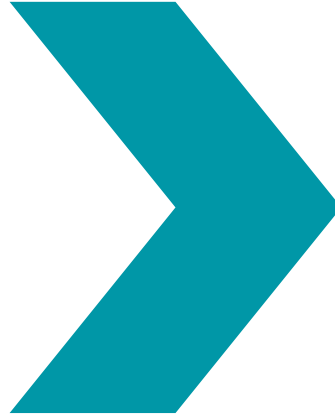
**Any code that is not tested**

What is legacy anyway?



## Why I love legacy

<https://www.goodreads.com/book/show/2718384-more-joel-on-software>



**Patterns: Strangler Fig Application (Martin Fowler)**



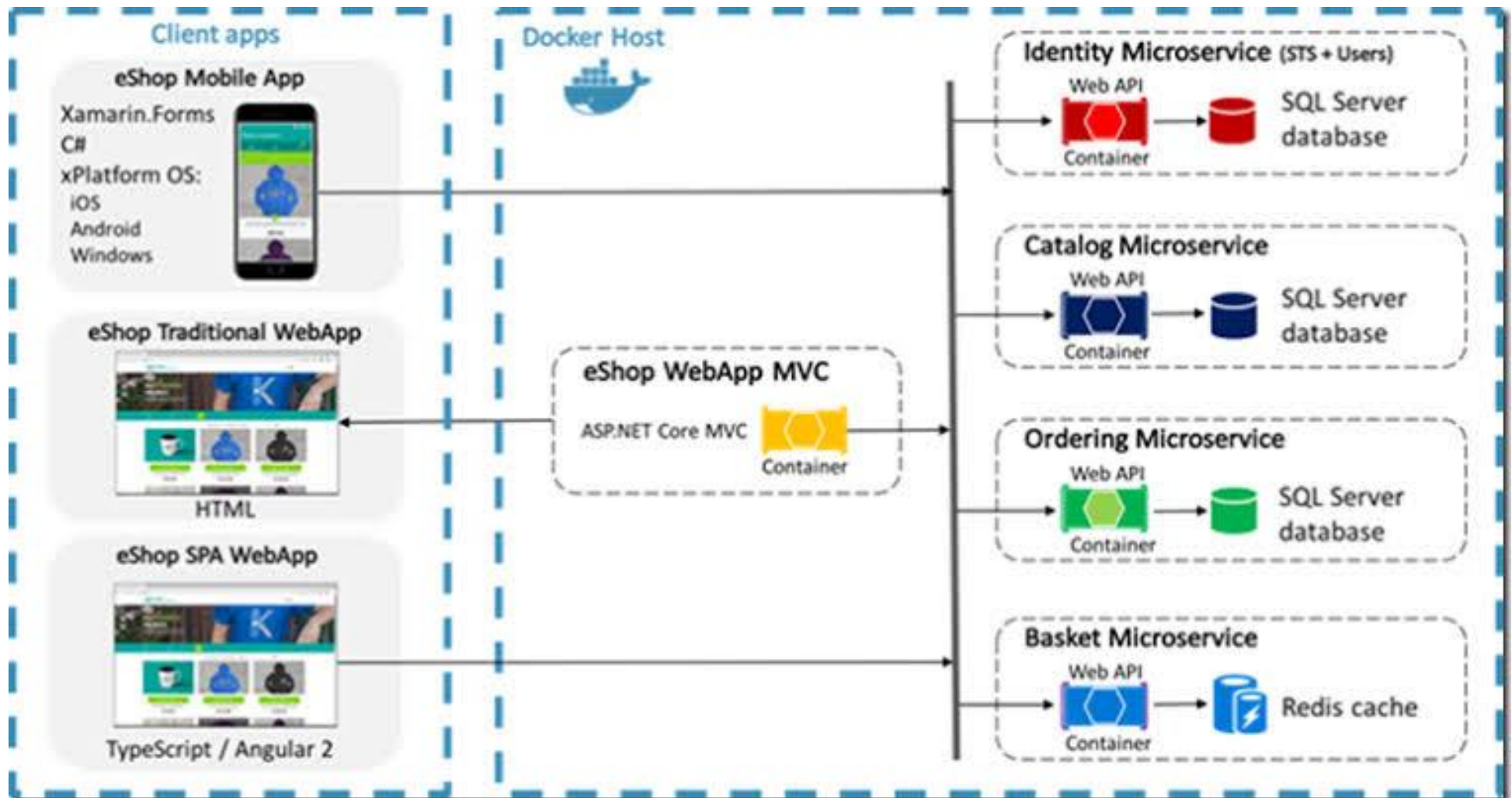
The diagram illustrates the Strangler Pattern for migrating a monolith to a microservices architecture. It shows a timeline of development over time, represented by a blue arrow at the bottom labeled "Time".

**Monolith Shrinking:** Below the timeline, a series of yellow boxes labeled "Monolith" represent the existing monolithic application. The boxes decrease in width from left to right, indicating that the monolith's scope is being reduced over time. An upward-pointing arrow from the text "The monolith shrinks over time" points to this sequence.

**Strangler Application Growing:** Above the timeline, a dashed box labeled "Strangler application" contains a growing stack of services. Each service is represented by a yellow box (existing service) and a green box (new feature) stacked vertically. The number of services and the height of the stack increase from left to right, showing the gradual accumulation of new features into the new architecture. A downward-pointing arrow from the text "The strangler application grows larger over time" points to this stack.

**Migration Process:** A green callout bubble labeled "New features" points to the green boxes, indicating that new functionality is being developed in the strangler application rather than the monolith. The overall process shows the monolith being decomposed into smaller, manageable services that are then incrementally replaced by the new strangler application.





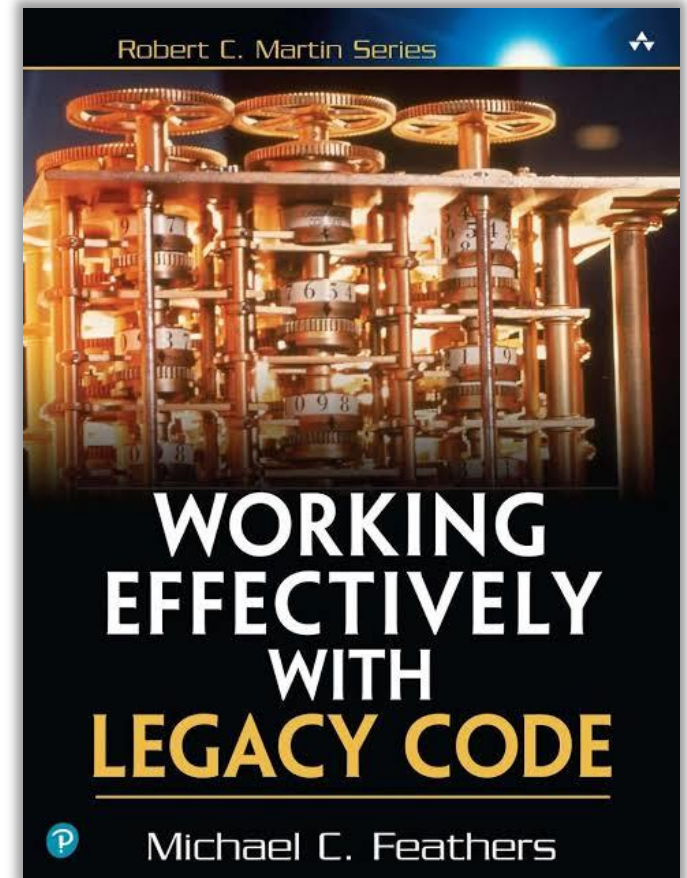
<https://devblogs.microsoft.com/cesardelatorre/free-ebookguide-on-net-microservices-architecture-for-containerized-net-applications/>



## **Patterns: Seams (Michael Feathers)**

<https://www.nzgeo.com/stories/the-obsidian-island/>

1. Identify seams in the codebase, ideally where domains converge (bounded context pattern)
2. Break dependencies across seams by introducing tests (dependency injection pattern)
3. Change legacy code safely and incrementally, automating test and release as-you-go
4. Manage/reduce fear in the team
5. Repeat...



**The playbook for refactoring legacy code according to Michael Feathers**

[https://www.goodreads.com/book/show/44919.Working\\_Effectively\\_with\\_Legacy\\_Code](https://www.goodreads.com/book/show/44919.Working_Effectively_with_Legacy_Code)



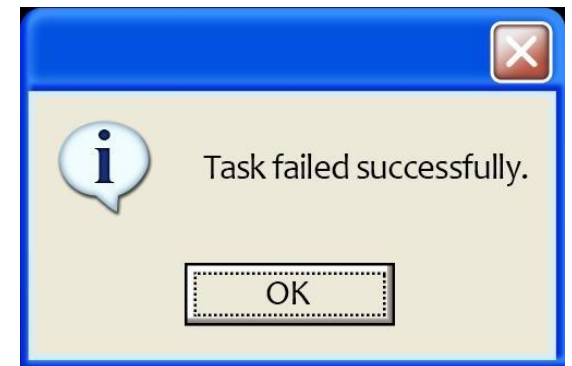


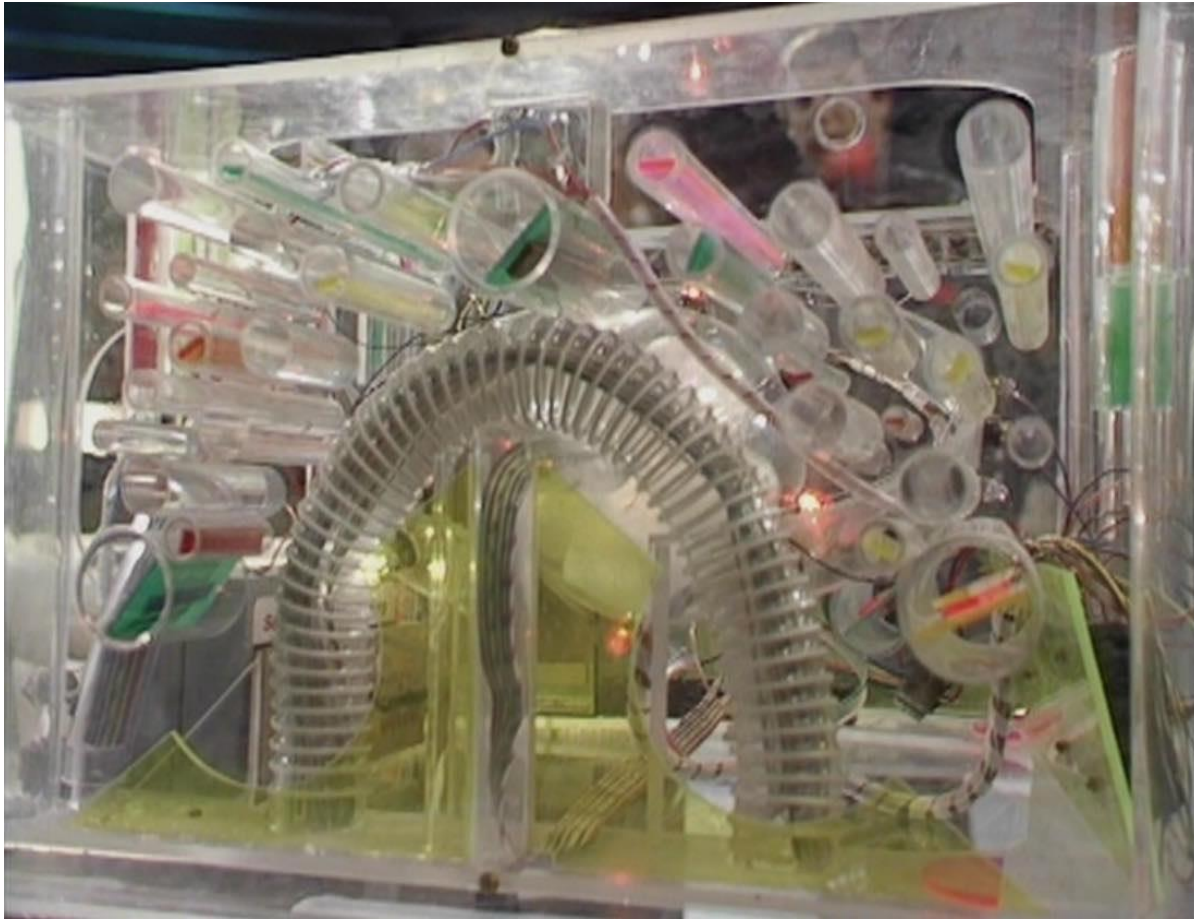
**Practices: table stakes stuff**

<https://knowyourmeme.com/memes/this-is-fine>

- Help your team to see and play the long game – anyone can do this (doesn't need to be the boss)
- Put a modernisation roadmap together and have clear, simple milestones, that target customer-led initiatives (and maybe a little zhuzh)
- Experiment with and use the emergent AI tools that specifically target legacy modernisation use-cases

**Practices: pragmatism, excitement, innovation**





**A perfect match: gen-AI & legacy modernisation**

<https://blakes7.fandom.com/wiki/Orac>

- Don't be scared of legacy - learn to love it; it's healthy
- Use well established patterns and practices to get a grip on your legacy code
- Things have changed since Joel Spolsky's blog post, but the underlying principals still hold – try not to reinvent the wheel
- There are strong parallels between AI-assisted coding and dealing with legacy
- We're probably just at the beginning of how AI will affect and benefit legacy modernisation

## Conclusions