
From Monoliths to Microservices

Strangler Pattern

- In this session we will look at the *strangler pattern*
- Motivated by a real-world telecoms use-case we'll see how a business domain decomposition can be used to drive the eventual phasing out of a monolith in favour of a microservices architecture
- At the end of this session we will hold a mini-retrospective to discuss what the team did well (and less well!)

Context

- Telecomms provider in North America
- Triple play digital phone/broadband/TV
  - Would like to become quad play with Mobile
- Have recently had major success with digital phone roll out, but worry about the future
- Becuase…

The Monolith

- Order management
- Rollout
- CRM
- Billing
- Accounting
  - And accounts outstanding, aka send the boys round
- Making the tea?

Why Change?

- Business frustrations:
  - The monolith caused delivery contention
    - Should it have?
- Technology pressures:
  - The monolith was expensive to support
    - Outdated tech, arcane
  - Trend towards service technology strong

Stevenson and Pols*

- Don’t reproduce legacy code
- Always ask the users what the problem is
- Refactor legacy application by delivering business value
- Incrementally build trust
  - Prove that you can do the hardest part of the system
- Involve the whole team with larger refactorings
  - So the team can move on as quickly as possible

** An Agile Approach to a Legacy System, Chris Stevenson and Andy Pols, XP 2004*

Guided Decomposition to Services

- We needed a target systems architecture
  - Otherwise we'd just be porting the monolith to new languages/runtimes
- Our goal was business capability -> service decomposition
  - Identify business owners, responsibilities, and dependencies


LIBERTY
Clarks
Clarks


£10
£20
£50
£5
Bank of England
FIVE
TEN
Five Pounds
Ten Pounds

Map Journey from Customers to Cash

- Start with the happy path
- In business terms only
  - No folk IT
- Be prepared to iterate several times
- Encourage debate (and reign in debate)


ANALYSIS

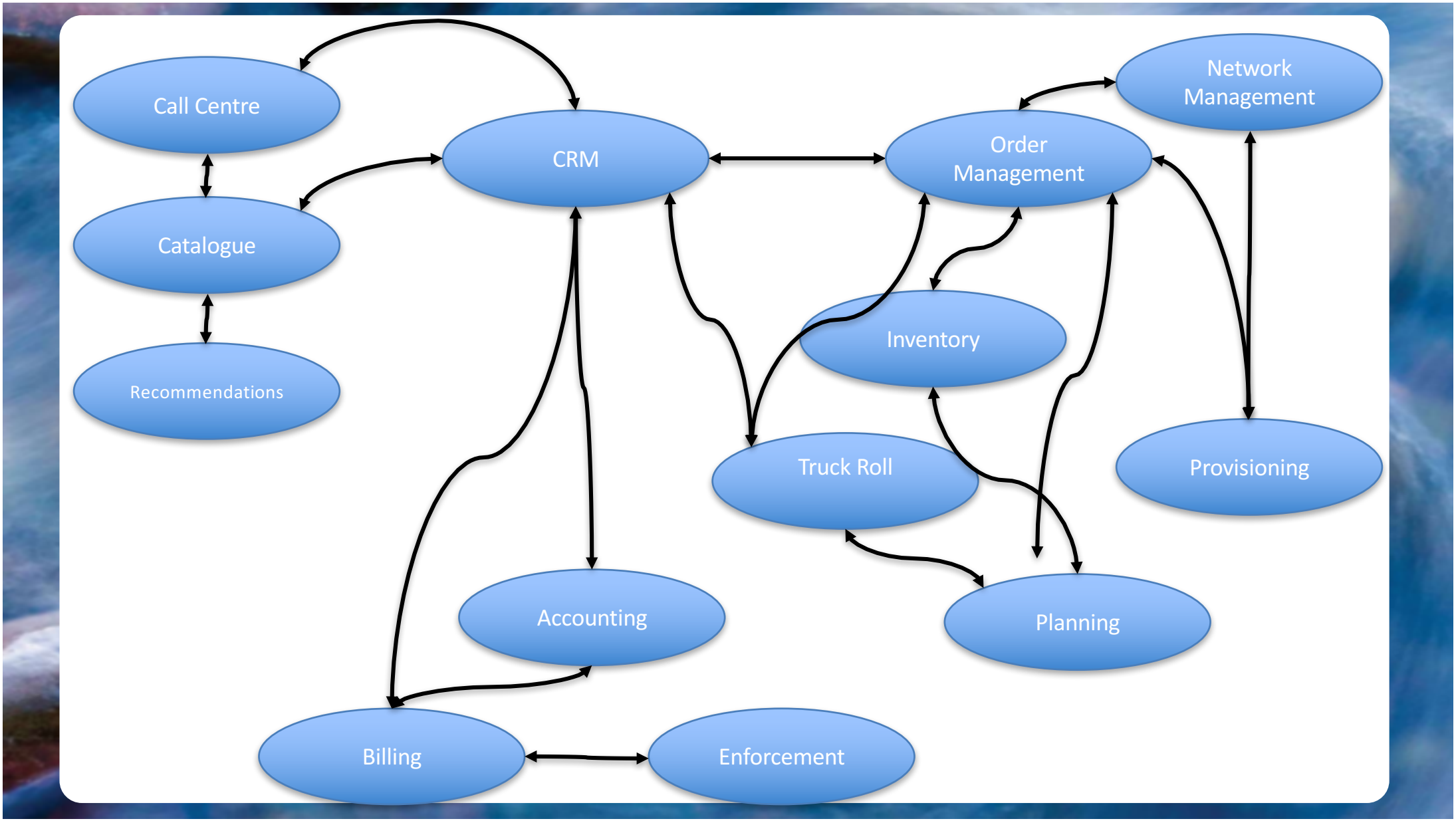
Life is paradoxically coincidental to the ironical tyranny applicable to the unparalleled definition of reverse entropy.

Just 13 services for the whole business

- One service to represent each business capability, each with an owner
- Some services interact more closely than others
  - E.g. CRM and Rollout, Billing and Accounts
- All play a role in customer to cash journey

13 services doesn't sound very micro

- These were the business capabilities that each had an (executive) owner
- This was the true business view of the world
- But in digitising each executive's capability, you discover that there are other finer-grained business capabilities to build
- And you could build them all into a monolith, or …
- You could build microservices for them


Call Centre
CRM
Network Management
Order Management
Catalogue
Inventory
Recommendations
Truck Roll
Provisioning
Accounting
Planning
Billing
Enforcement

Services are a Platform

- Product design *apps* use the catalogue and power recommendations
- Network planners use the network service
- E-commerce and human customer service all based on the same core services
- etc

Now we have a target architecture

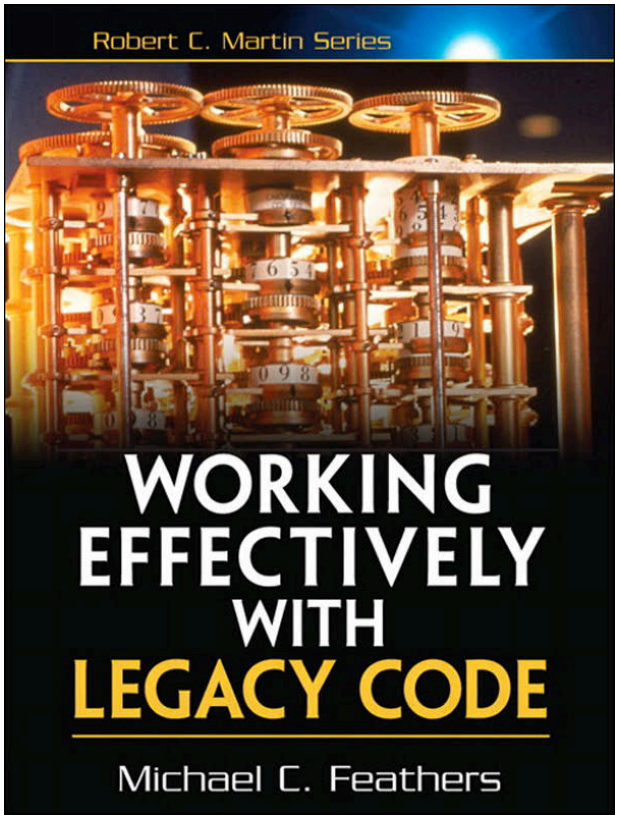
- Expressed as services
- Bounded by business contexts
- Not an artificial architecture exercise
- So we can start to port and enhance functionality but first…

Don’t Just Write Services

- Function by function, feature by feature wrap the calls the monolith and replace
- Your code should be tested for equivalence to the monolith
  - But may surpass that functionality
- Replay tooling exists to assert this at wire level

Don’t forget it’s legacy code work

- So behave that way
- As you slice functionality, make it tested
- Work along seams
- Prop up your seams with tests



Mantraps

- Politics
- Architects
- Vendors (of legacy tech)
- Fiefdoms
- Business/IT divide

- Even just scheduling your senior stakeholders together is painful

Mini-Retrospective

- What went well?
- What could have been done better?
- Any puzzles?

- Let’s take a few minutes to discuss…

Summary

- Strangler (and it's OK to keep the monolith around)
- Customer to cash (or equivalent flow for you)
- Business terminology only, no folk IT
- Generally few top level services drop out
- Other services build atop this platform
- Learn from my fails (politics)