AXXES
IT CONSULTANCY

Stop punching yourself in the face!

Hannes Lowette

We need to talk ...

What we do

The magpie

- Bird (black & white)
- Long tail
- Up to ~45cm long
- Wingspan up to ~60cm
- Omnivores
- Intelligent

(Rumoured to be…)

- Attracted to shiny things

The software developer

- Human being (black/white/…)
- No tail
- Up to ~220cm tall
- No wings
- Omnivores
- Intelligent

(libraries, frameworks, architectures, …)

- Attracted to shiny things

Is this surprising?

Technology evolves

No recipe for success

We chose this profession regardless …

Our education …

Can you design and build a house?

We look for guidance

goto; conference

{ } NDC Conferences

TECHORAMA

Build Stuff

Google

amazon

NETFLIX

Spotify

And we do as they do …

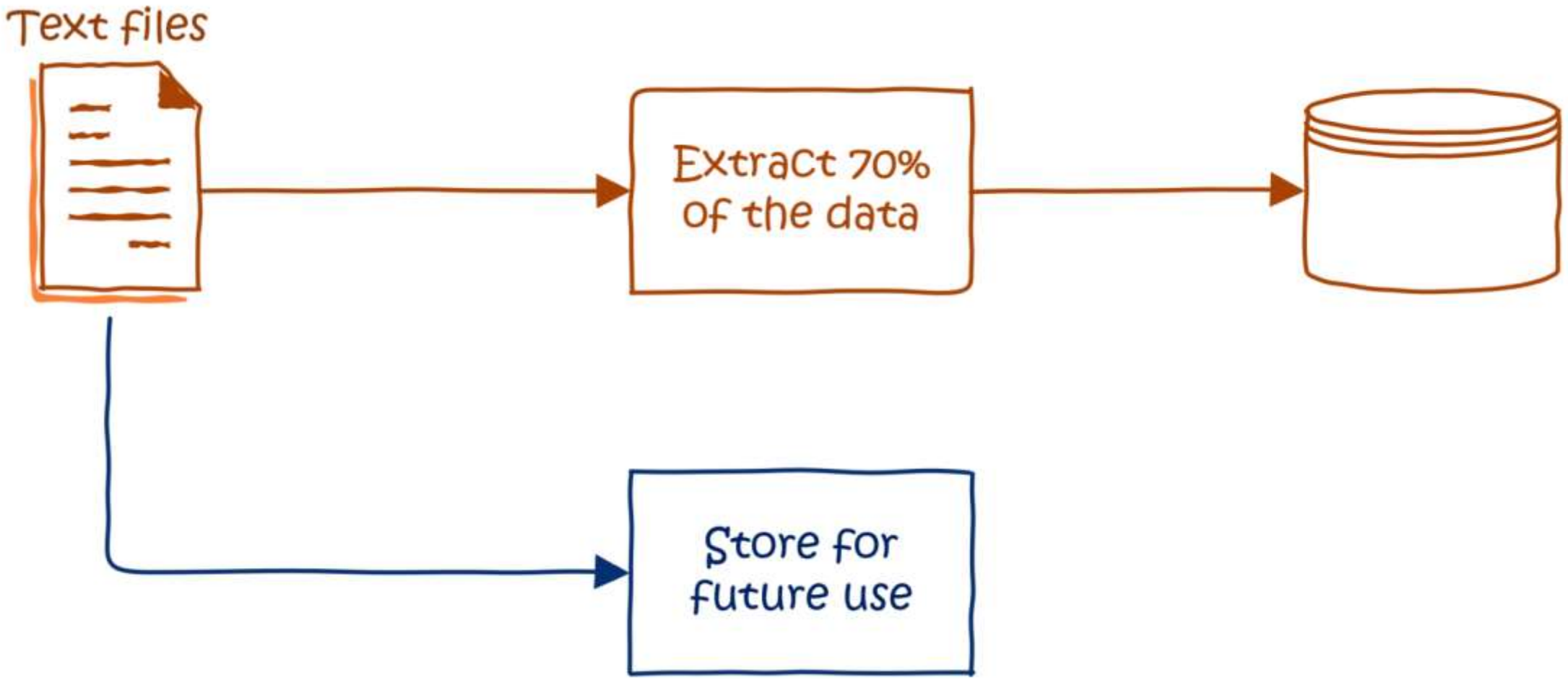
… until we run into a wall!

But why do we run into that wall?

Join me for an adventure with CHAD* …

** **Any resemblance** to actual **persons**, living or dead, or actual events is purely coincidental.*

The case



CHAD’ s solution

What went wrong?

RavenDB

- € Per GB: pretty high
- Maintain the OS
- Maintain the DB engine

The solution

- Took longer to write
- Breaks on file structure changes

Technical fit

Is this technology suitable to solve my problem?

Technical fit

Ask yourself questions such as:

- Does this help with our problem?
- How does it compare against competitors?
- Is there something that will help us more?
- Is this the simplest thing we can do?
- Does it bring unnecessary costs?
- Is there extra maintenance?

Why did CHAD go wrong?

RavenDB is a database

→ If you only use it for storage, it is expensive

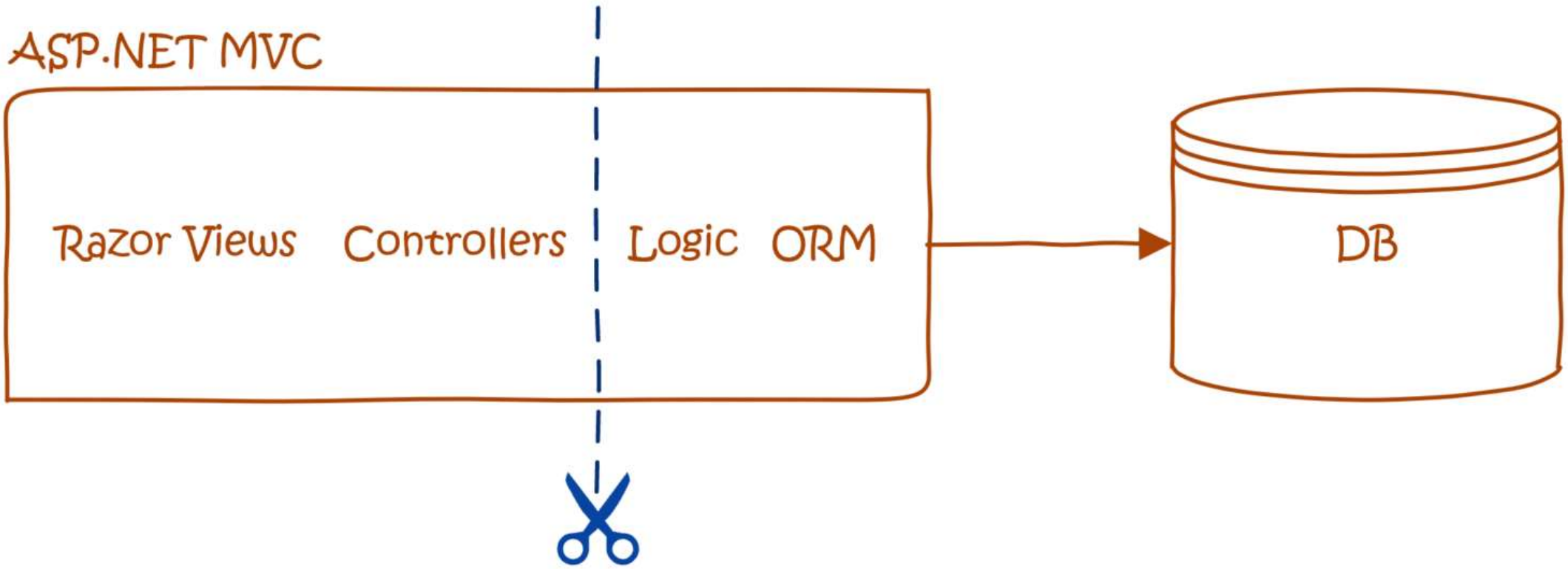
What could CHAD have done?

Azure BLOB storage:

- Easy to set up
- No parsing required
- € per GB: very cheap
- Very robust (and easy to distribute)
- No maintenance needed


The
Agatha
Story

The case

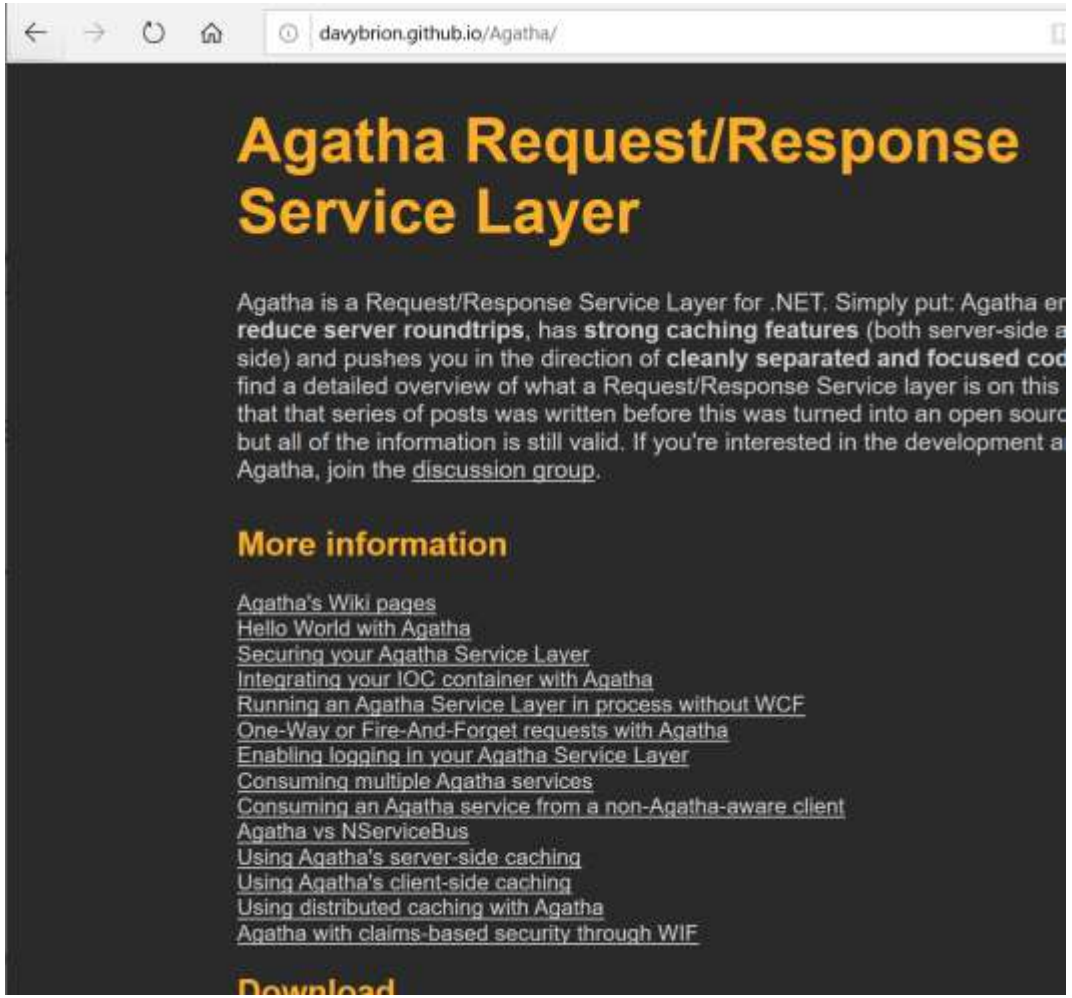


CHAD’ s solution

Agatha RRSL

- CHAD had experience with it
- Agatha in MVC
  - Solved our issue
  - In process (no overhead)
  - But could move to ‘over WCF’

→ Technical fit was OK



What went wrong?

Agatha:

- Mainly maintained by 1 person
- Maintenance stopped in 2013

→ Agatha had become a liability

Check for warning signs

Can this technology become a risk?

Warning signs

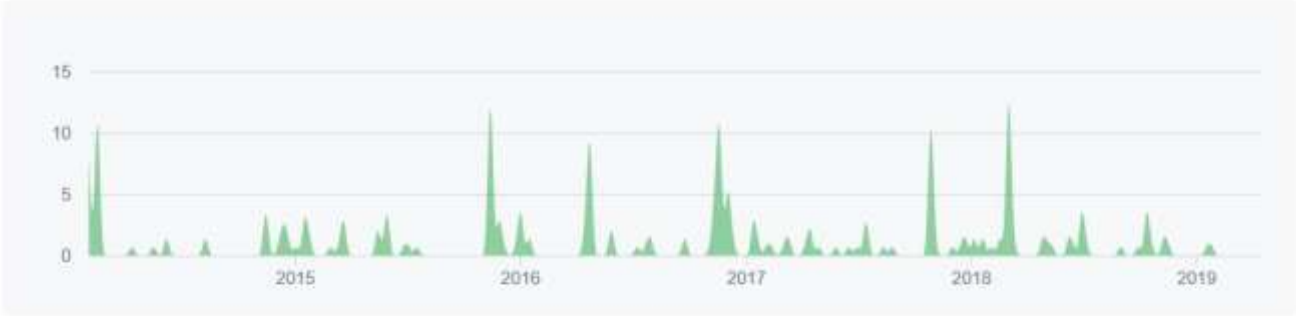
Ask yourself questions such as:

- Who maintains this library?
- Are they still actively working on it?
- Is the source available?
- What licensing model does it use?
- Has the product recently been acquired by another party?

What could CHAD have done?

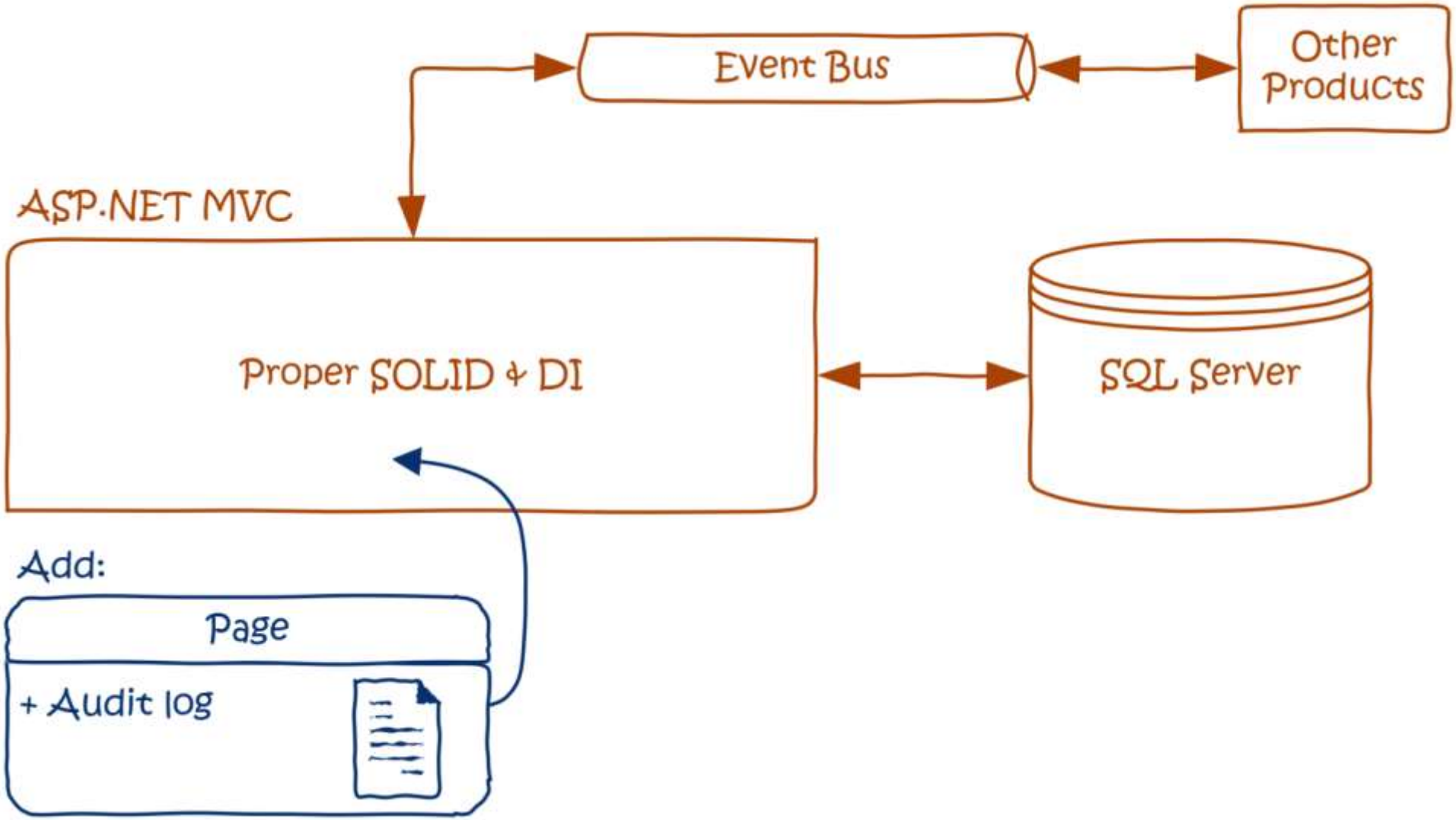
MediatR:

- Similar functionality
- Open source, Apache license
- Actively maintained
  (by multiple people)




The
EventStore
Story

The case


Event Bus
Other Products
ASP.NET MVC
Proper SOLID & DI
SQL Server
Add:
Page
+ Audit log

CHAD’ s solution

EventStore

→ Technical fit was OK
→ No warning signs

What went wrong?

The team:

- Juniors
- Remote contractors
- No EventStore knowledge

→ The team never touched the EventStore stuff
CHAD was the only one who worked on that

Team skills

Can we get everyone up to speed on this?

Team skills

Ask yourself questions such as:

- Do we have the knowledge?
- If not, is the knowledge available?
- Can we teach the team?
- Can everyone work on it?
- What documentation do we need?

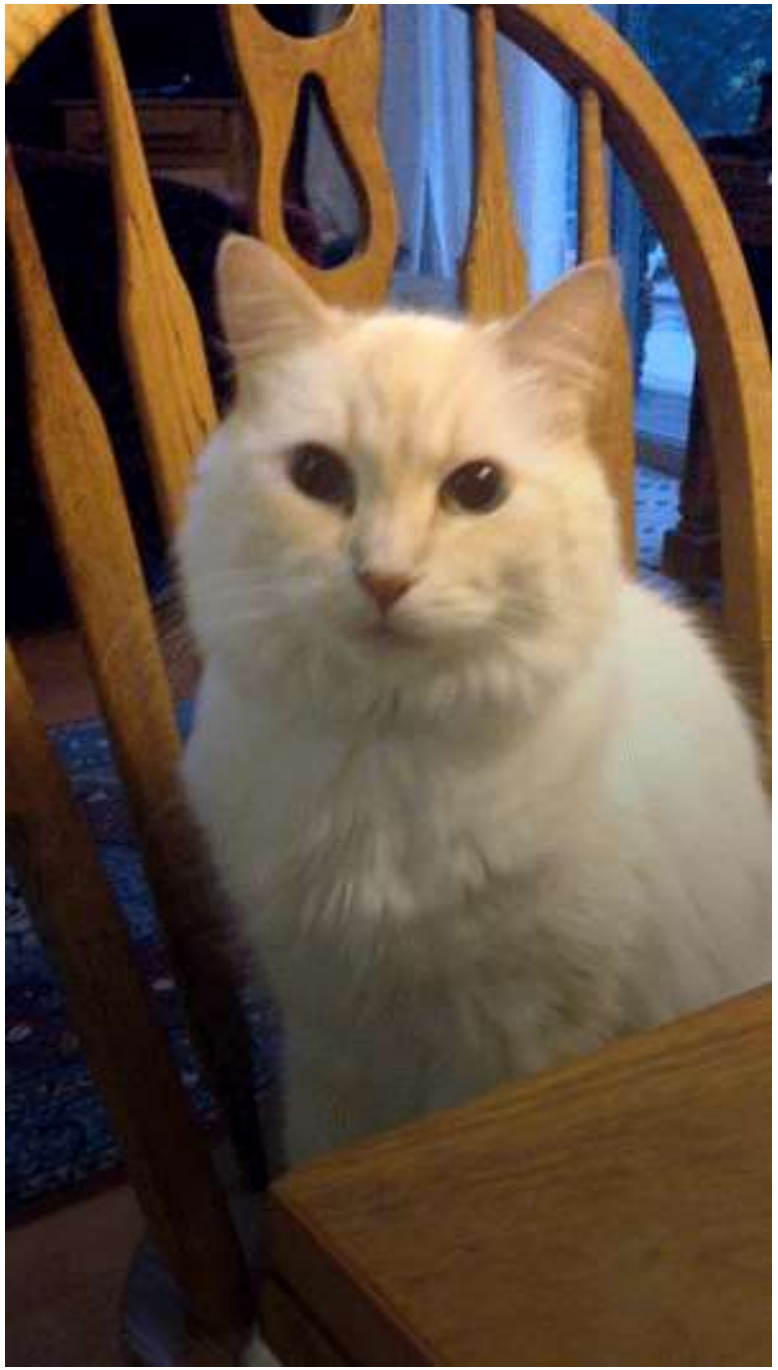
What could CHAD have done?

Spread the knowledge:

- Teach the team
- Write code together
- Write documentation
- Help others take their first steps

What happened

"We need to stop using Entity Framework"



EF frustrations

- Very slow
- Does too many things
- No control over SQL
- The SQL is very messy

→ True, but …

What went wrong?

We used:

- SQL server profiler
- Query plan analyzer

We saw:

- DbContext initialization
- N+1 select
- Missing DB index
- Too many fields fetched

Until that day

Entity Framework was
magic to CHAD

Take away the magic

Do you understand how it was built?

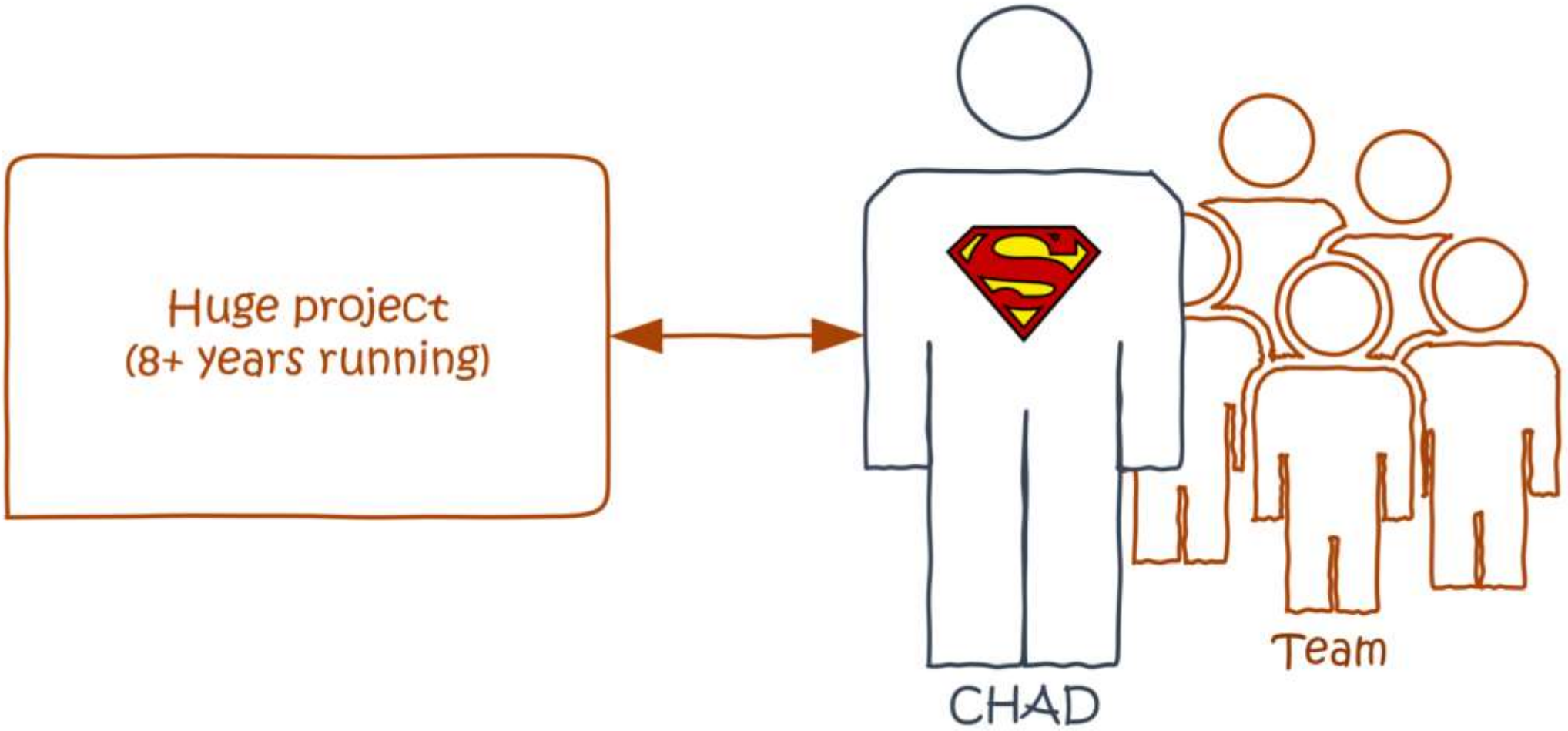
Take away the magic

Ask yourself questions such as:

- Do we know how it works?
- Do we understand how it was built?
  - What did the developers use?
  - What concepts are applied?
  - Could we (theoretically) build our own?
- Do we know the effect of our usage?


The
Big Old Project
Story

The case



Their approach

- When CHAD learned
  - A better way
  - A new framework
  - An awesome library
- He explained it to the team:
  - Presentation
  - Pitfalls
  - Tips & tricks

→ From then on,
they would do it like that!

What went wrong?

Old code stayed ‘the old way’

Trim your stack

When new tech goes in, old stuff must go out!

Trim your stack

Ask yourself questions such as:

- What problem does this new tech solve?
- Do we really need that?
- Which old, similar, thing do we have?
- Can that be replaced by the new one?
- How much effort would that be?
- What effect does it have to keep booth?
  - On maintenance
  - On onboarding

What could CHAD have done?

"Hollow out the trunk"

The case

The solution

- Angular
- IdentityServer
- Docker & Kubernetes
- Microservices
- MongoDB

→ All of this was new to them

What went wrong?

The team got stuck on:

- Learning the new stack
- Deployments
- Debugging
- MongoDB

The project got a huge delay

→ Massive ‘team skills’ violation
→ Over-engineered

Team skills - revisited

New skills in real projects:

- Don’ t combine learning & delivery goals
- Budget extra time
- 1 at a time
- Explain the risk to the business

“This new way of doing things will make sure we can deliver more, with higher quality.”

Evolving architecture

Architecture follows your problems, not the other way around!

Evolving architecture

KISS

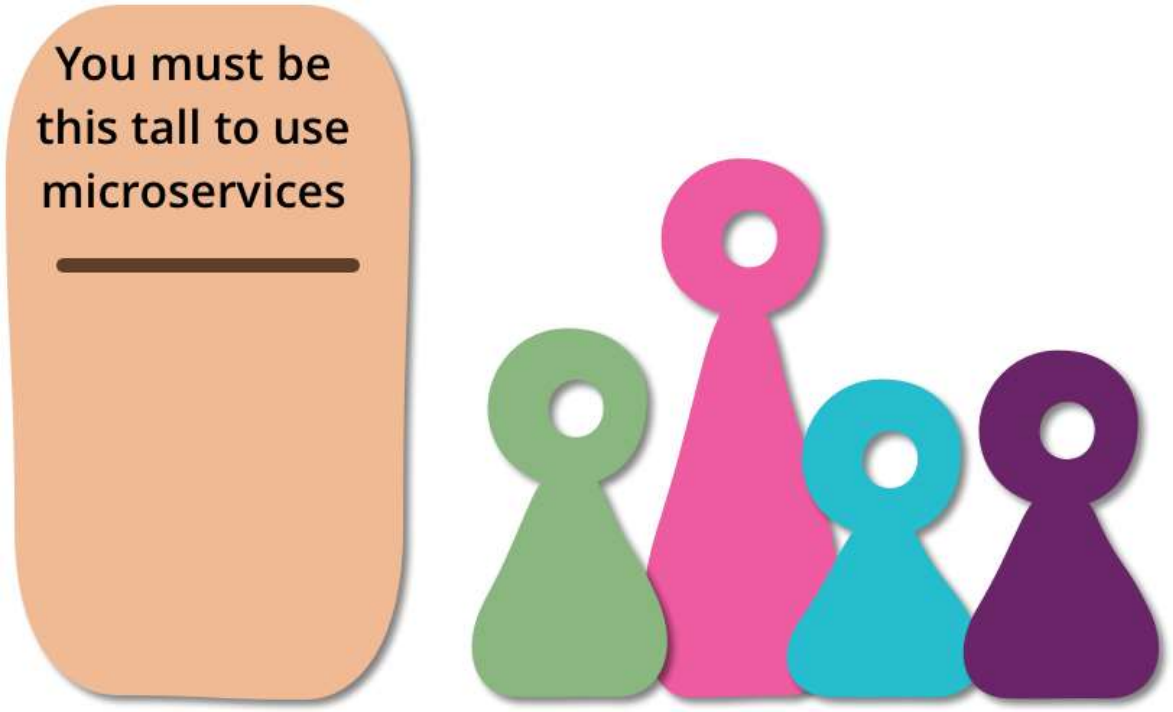
- Start with the simplest solution
- Discover your needs from PROD
- Adapt to those needs, extract services

→ You’ ll have a working product WAY faster

Microservices?

When?

- Scale independently
  → Balance cost with speed
- Multiple clients
  → Clients can adopt features at will
- Different technology stacks
- Conflicting dependencies



What could CHAD have done?

Let the architecture follow the needs

Start simple:

- ASP.NET & Angular (1 new thing)
- No microservices

Structure code to allow decoupling:

- SOLID
- Vertical slices

→ Deliver your value!

Summary

1. Technical fit
2. Check for warning signs
3. Team skill
4. Take away the magic
5. Trim your stack
6. Evolving architecture

No really, who is CHAD?

- Me
- The whole team
- Someone else
- Totally made up

→ He acted because he cared!

Carelessly or Helplessly Acting Developer

Wouldn’ t it be better to be a …

RAD?

Responsibly **A**cting **D**eveloper

Are you on board with this mission?

1. Spread the word!

2. Follow us on Twitter: @RadCert
3. Take the quiz at https://rad-cert.com
4. Come find me later to pick up your SWAG

Hannes Lowette

AXXES
IT CONSULTANCY

- Head of Learning & Development
- Competence Coach .NET

- Father of 3
- Amateur guitar builder
- LEGO® enthusiast

@hannes_lowette

#20086521

Questions?