THE SIMPLE LIFE
OF MICROSERVICES
Microservices, microservices, microservices, microservices, microservices, microservices, microservices, microservices, microservices.
Once upon a time…
Monolithic architecture
Starting from scratch
Confirmation bias, also called confirmatory bias or myside bias, is the tendency to search for, interpret, favor, and recall information in a way that confirms one's preexisting beliefs or hypotheses. It is the natural human inclination to interpret information in a way that is consistent with one's existing beliefs and biases, even if that information is not completely accurate or objective.
“Organisations which design systems ... are constrained to produce designs which are copies of the communication structures of these organisations”

—M. Conway
design
Simplest thing that can work

\[ \frac{X \times Y \times 6 \times 3}{213141} = \frac{144}{3614} \]

\[ 65 \times 100 \times \frac{9 \times 34}{44} \times \frac{100}{10} \]
Run it!

- Console app
- ASCII art
GET /talking/to/it

200 OK

Content-Type: application/json

{
    "http is": "client-server",
}
The blue bible

Root aggregates
1994
Welcome to Microsoft's World Wide Web Server!

Where do you want to go today?

If your browser doesn't support images, we have a text menu as well.

Knowledge Base
Back Office & Windows NT Workstation
ftp.microsoft.com
Microsoft TV
Employment Opportunities
Windows News: gopher.microsoft.com
Developer Network OffRamp
Microsoft TechNet
Microsoft Sales Information
Windows Sockets
What's New
The Microsoft Network
Send us your feedback
Facts about this server
Resources

Concepts “named” with URIs
Hypermedia

Links, forms, even unicorns.
MARTY, WE HAVE TO GO BACK
1994… 2002
Ports and adapters
“[...] a symmetric architecture in which the database is considered not at the bottom of the stack, but fully outside the application”

–Alistair Cockburn
HTTP

In-memory Persistence

Recon

Mongo Persistence
(because internet scale or something)
public class ReconHandler
{
    private readonly IReconStorage _storage;

    public ReconHandler(IReconStorage storage)
    {
        _storage = storage;
    }

    public Link<Recon> Post(Enquiry userEnquiry)
    {
        var newRecon = new Recon(userEnquiry);
        _storage.Store(newRecon);
        return Link.To(newRecon);
    }
}
Semantic Persistence
public class Recon
{
    public ICollection<object> Events { get; set; }
    public Recon(Enquiry userEnquiry)
    {
        ReconStarted(userEnquiry);
    }

    private void ReconStarted(Enquiry userEnquiry)
    {
        Events.Add(new UserStartedRecon(userEnquiry));
    }
}
public class InMemoryReconPersistence : IReconStorage
{
    Dictionary<int, BsonDocument> _storage;
    public void Store(Recon newRecon)
    {
        _storage[newRecon.Id] = BsonSerializer.Serialize(newRecon);
    }
}

public class MongoReconPersistence : IReconStorage
{
    public void Store(Recon newRecon)
    {
        foreach(var @event in newRecon.Events)
        {
            if (@event.GetType() == typeof(UserStartedRecon))
            {
                imaginaryMongo.Insert(((UserStartedRecon) @event).Enquiry);
            }
        }
    }
}
VeST

Vertical Slice Testing - building one slice of the system at a time, scenario-driven
public class user_makes_a_new_enquiry : contexts.recon
{
    public user_makes_a_new_enquiry()
    {
        given_car_maker_has(product: "Dolorean", price: 1000000m);
        when_user_makes_enquiry(new Enquiry {Product = "Dolorean"});
    }

    public void quote_has_car()
    {
        base.quotePersistence.Single().Product.ShouldEqual("Dolorean");
    }

    public void quote_has_price_given_by_car_maker()
    {
        base.quotePersistence.Single().Price.ShouldEqual(1000000m);
    }
}
Adding behaviour
HTTP as the application protocol
Layering
{
    "enquiry": {
        "product": "dolorean"
    }
}
{
  "providers": {
    "dmc": { "active": false },
    "VW": { "active": maybe }
  },
  "enquiry": {
    "product": "dolorean"
  }
}
Wiring
public class ApplicationStartup
{
    public void OnStartup()
    {
        this.owinApplication = new OwinApplication(
            app => app.UseRecons()
                .UseProductEnrichment());
    }
}
“If it doesn’t work for 1,000 machines, it doesn’t work.”

—Me
AUTOMATE ALL THE THINGS
Introspection
Divide and conquer
Key learnings
• Log all boundaries
• Keep an audit log
• Make the implicit explicit
• Design key metrics in each feature from the get go
• Monitoring dashboard + devops-oriented specialised interface
@serialseb