Taking the hippie bus to the enterprise

Warm Crocodile Developer Conference 2013

Mogens Heller Grabe

d60

mhg@d60.dk

http://mookid.dk/oncode

@mookid8000

What?

- What's your problem?
- Meet Rebus
- Examples
- Wrap up

How?

- Slides
- Talk
- 4 demos

What's your problem?

Big system

size

Integration with external parties

(uhm hello? anybody out there?)

Complex logic

Stuff that makes stuff happen

...that makes other stuff happen

...that makes other stuff happen

...then waits for a while

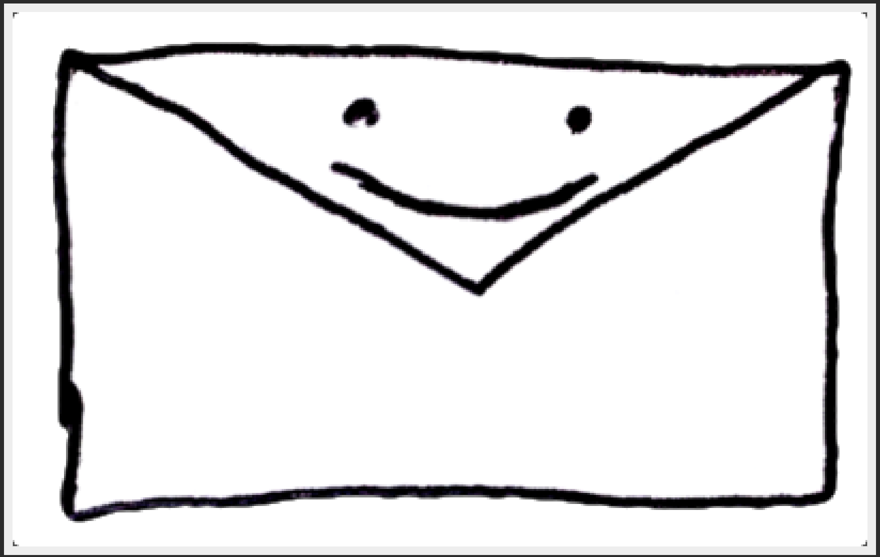
...then other stuff happens

(you probably get it)

Problem summary

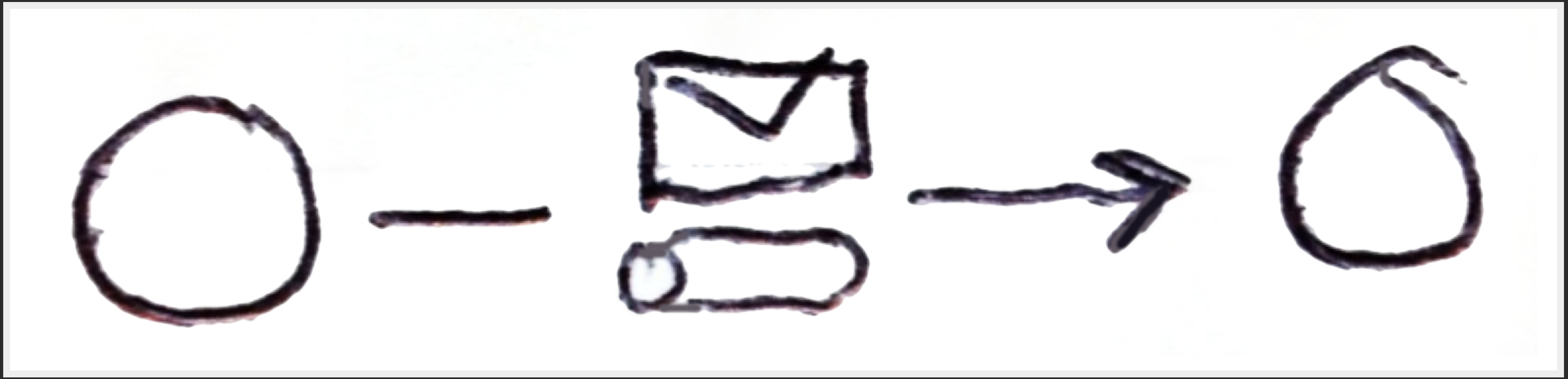
- Monolith
- Integration
- Coordination

The solution: Messaging



(and by "the" I mean "a")

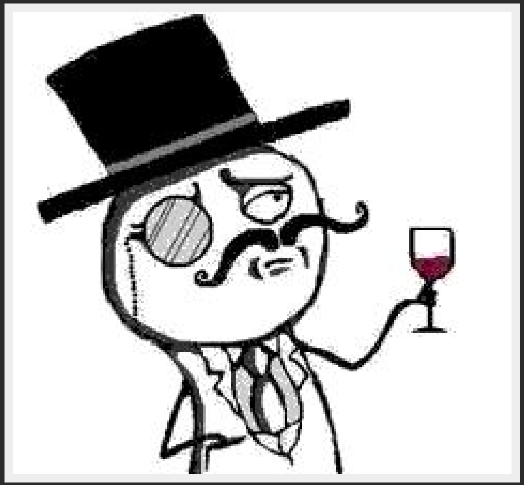
Messaging via durable, asynchronous queues



Windows FTW!!1

Meet Rebus

"Service bus"?



"Hippie bus"?

No, seriously - what is it?

- Messaging library
- Layer on top of MSMQ
- One single .NET 4 DLL
- Additional DLLs if you want
  - RabbitMQ
  - RavenDB
  - MongoDB
  - Castle Windsor
  - StructureMap
  - Unity
  - Autofac
  - Ninject
  - Log4net
  - NLog
  - ...

Motivation

- I really like NServiceBus
- Can't use NServiceBus where I want to though
- Sometimes I lost my patience with NServiceBus
- Wanted to use MassTransit
- Wanted to fork NServiceBus when it was still Apache V2

Philosophy

- Free
- Easy

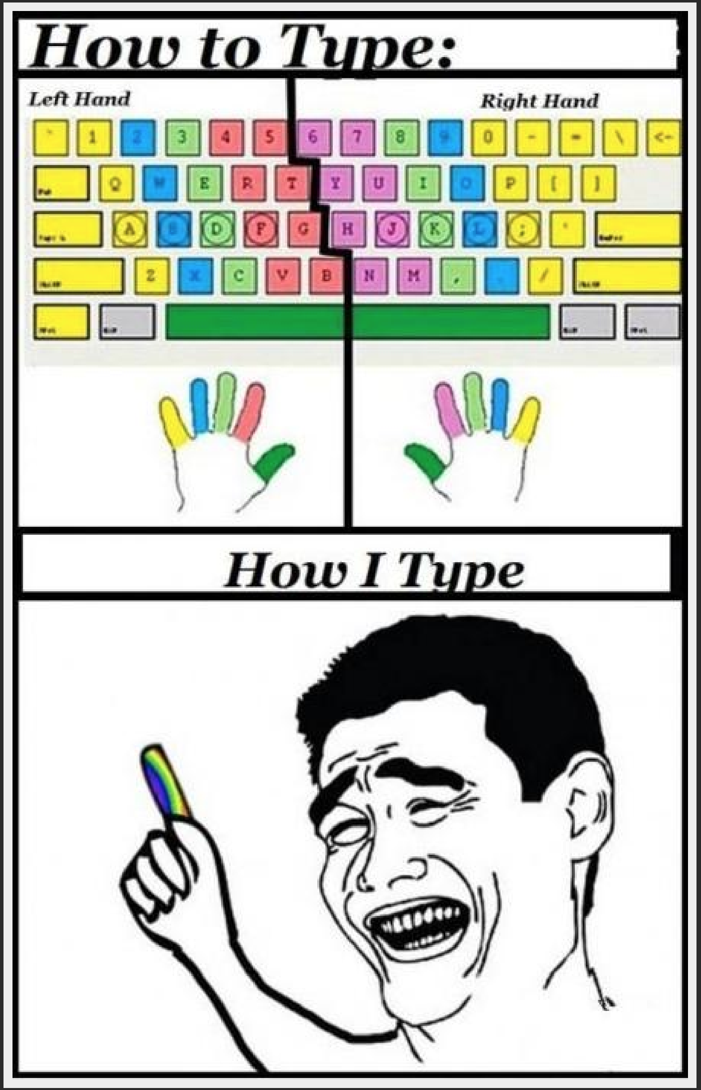
Meta

- 3000 lines of C# 4
- Code on GitHub: **https://github.com/mookid8000/Rebus**
- Has contributions from 7 developers besides me
- Binaries available via NuGet **http://nuget.org/packages?q=rebus**
- Current version: 0.28.3
- Has been moving money around since 0.14-alpha
- Has controlled power plants since 0.17-alpha

Good to know

- Messages are POCOs
- Each logical service/endpoint/process has its own input queue
- Each message type is owned by one logical service

Demo 0



Examples

The three problems I talked about

1. System that's becoming too big
2. Integration with external parties
3. Complex logic with coordination and timing

1st problem

System that's becoming too big

Summary: We're building a trading platform where traders in "front office" strike deals with counterparts and record their trades, while administrative personnel in "back office" make sure that the counterparts are charged.

Cues

- ubiquitous language
- bounded context
- anti-corruption layer
- distributed domain-driven design

Demo 1

Split into separate Trading and Billing systems

2nd problem

Integration with external party

Summary: When new trades are made, "middle office" needs to confirm all trades, e.g. depending on the current credit status of the counterpart. Credit status can be retrieved by querying the CreditAssessment SOAP service.

Cues

- asynchronous
- reliable
- automatic retries

Demo 2

Web service calls via messaging facade

3rd problem

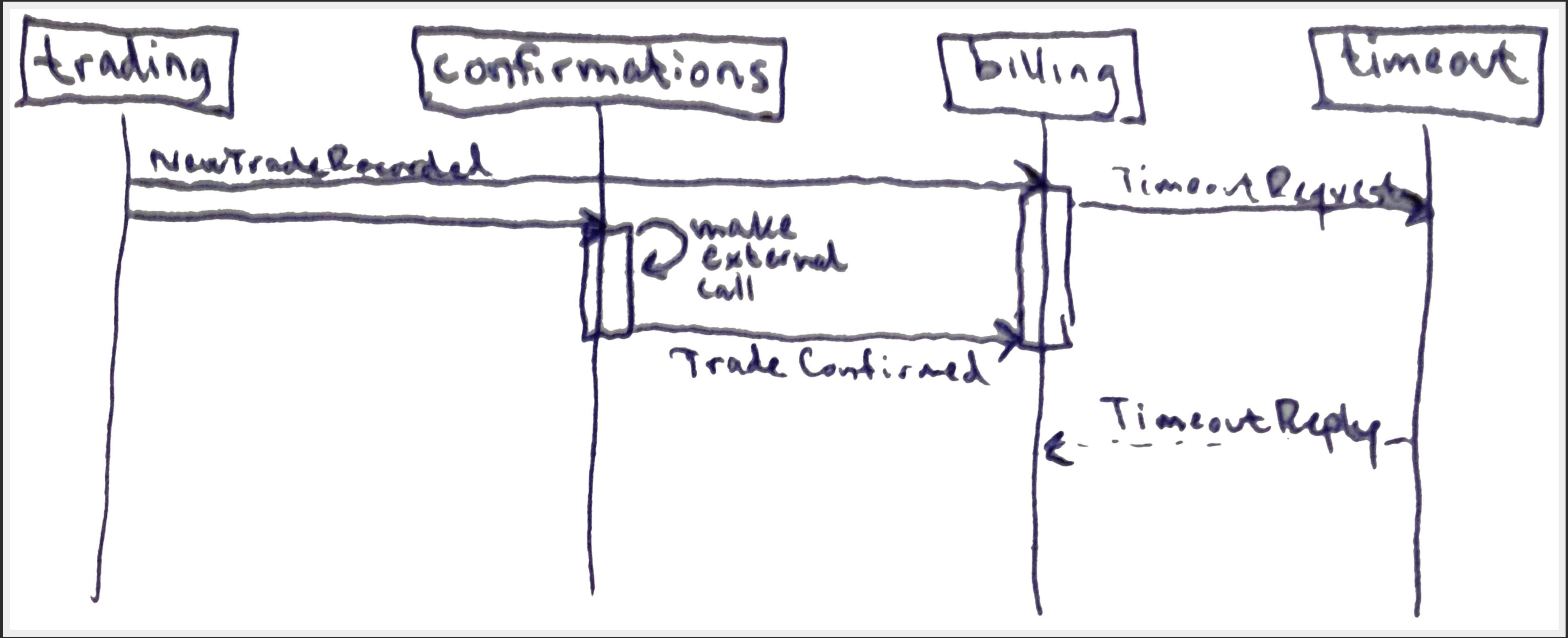
Complex coordination and timing

Billing needs to charge the customer, but it can be optimized by sending one big invoice if the credit status is OK. Therefore, when trades are made, billing awaits confirmations' judgment to determine invoicing terms.

To avoid "forgetting" to send invoices in case something goes wrong, we want to take action if the automated invoicing is not complete within 10 seconds.

Cues

- process manager
- timeouts
- compensating actions


trading
confirmations
billing
timeout
NewTradeRecorded
make external call
TradeConfirmed
TimeoutRequest
TimeoutReply

Demo 3

Complex logic, coordination, timing

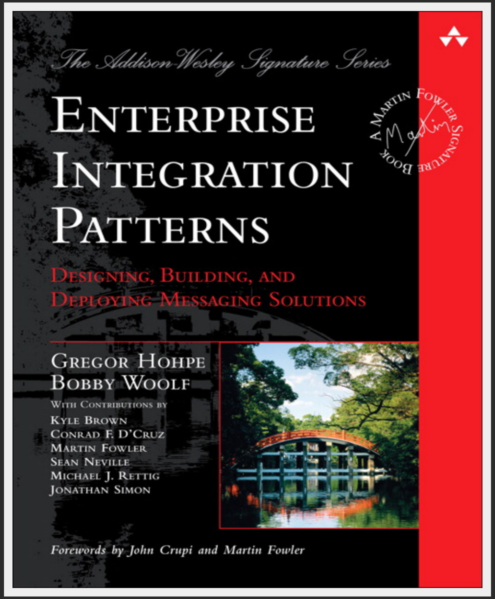
Stuff I didn't show

- Can use MSMQ and RabbitMQ as transports
- Can store subscriptions and sagas in SQL Server, RavenDB, and MongoDB
- Can activate handlers with Castle Windsor, StructureMap, Autofac, Ninject, and Unity
- Can log with NLog and Log4Net
- Can send messages in batches
- Can do handler pipeline re-ordering
- Can do polymorphic dispatch
- Can encrypt message bodies

What now?

- 1.0
- Central monitoring
- HTTP gateway
- Additional transports (e.g. Azure Service Bus)
- Distributor

Litterature


The Addison-Wesley Signature Series

ENTERPRISE INTEGRATION PATTERNS

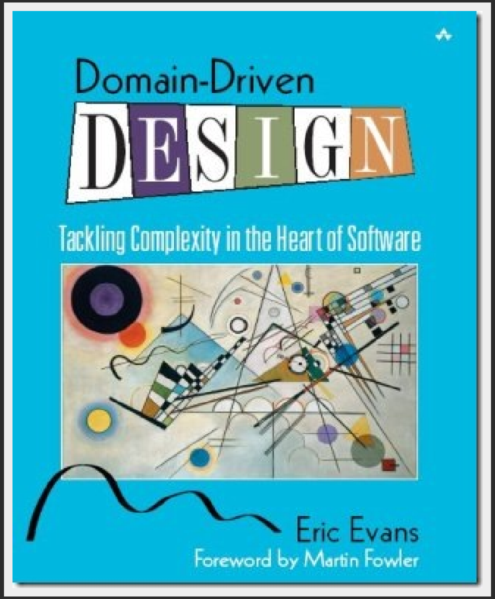
DESIGNING, BUILDING, AND DEPLOYING MESSAGING SOLUTIONS

GREGOR HOHPE
BOBBY WOOLF

WITH CONTRIBUTIONS BY
KYLE BROWN
CONRAD F. D'CRUZ
MARTIN FOWLER
SEAN NEVILLE
MICHAEL J. RETTIG
JONATHAN SIMON

Forewords by John Crupi and Martin Fowler

A MARTIN FOWLER SIGNATURE BOOK


Domain-Driven DESIGN

Tackling Complexity in the Heart of Software

Eric Evans
Foreword by Martin Fowler

Want to read more?

Check out the work of Udi Dahan, Greg Young, Dan North, Rinad Abdullin etc.

Thank you for listening!

...and a big thank you to **Hakim** for creating the immensely awesome **reveal.js**

Mogens Heller Grabe

mhg@d60.dk

@mookid8000

http://mookid.dk/oncode

...and a big thanks to **Silly Dog Hats** for the cute and inspirational artwork