

DevWeek 2011
“From Legacy To Epic”
Mark Dalgarno
& Chris Morgan

redgate®

ingeniously simple tools

Who are we?



Mark Dalgarno

Head of Web and Internal Development

mark.dalgarno@red-gate.com

Twitter: @MarkDalgarno



Chris Morgan

Web Developer

chris.morgan@red-gate.com

Twitter: @cjm55

redgate[®]

ingeniously simple tools

Who are Red Gate?

- Cambridge-based ISV
- Make tools for Devs, DBAs and sysadmins
- We target SQL, .NET, Oracle
- “A great place to work” :-)



redgate®

ingeniously simple tools

Takeaways in 90 minutes...

In scope:

- Discuss: *From Legacy to Epic*
- 3 Pillars: Technology, Process, People

Not in scope:

- Silver bullets, code, demos
- 'Academic' solutions
- Nodding off after a big lunch ;-)



Why you might be here...

Developer: *Does all code suck this bad?*

Architect: *Why is change so hard?*

Manager: *Why are estimates so big?*

Everyone: *There must be a better way!*

Our take on “Legacy”

- A world of NO :-(
 - No tests
 - No documentation
 - No experts
 - No buzz / flow
- Hard to improve
- Team want to leave :-)

Sad



Panda

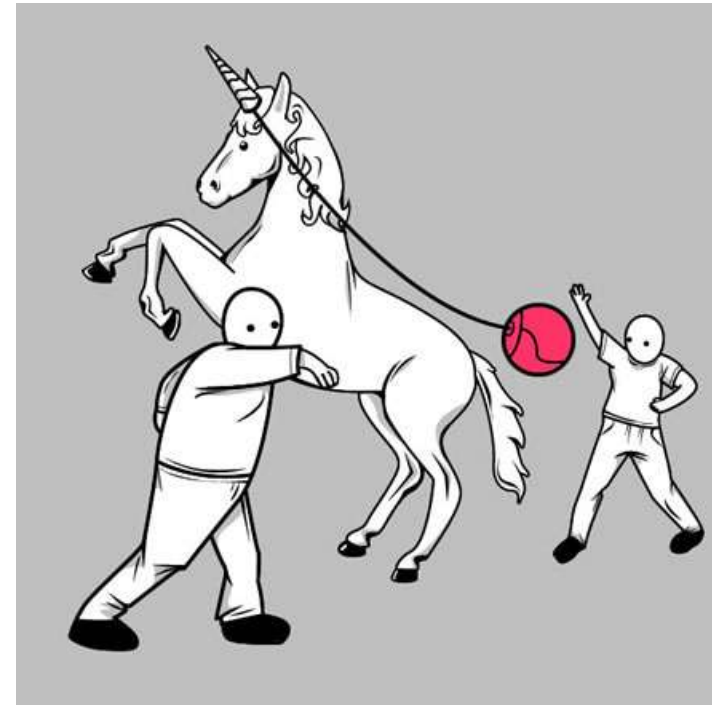
redgate®

ingeniously simple tools

Our take on “Epic”

A world of YES :-)

- **Feedback is welcome**
- **Change is friction-free**
- **Scale without disruption**
- **Reach your potential**
- **Happy team :-)**



redgate®

ingeniously simple tools

A quick exercise to warm up

Think about projects you've worked on

- **What made them terrible?**
- **What made them awesome?**

and in 5 minutes time...

- **Share what you've discussed with us**
(yes, even if your manager is next to you)

Buzzword Bingo

- **Embrace Change**
- **Legacy Code**
- **Clean Code**
- **Unit Tests**
- **Refactoring**
- **Code Smell**
- **Technical Debt**
- **Dependencies**
- **Coupling**
- **Cohesion**
- **SOLID Principles**
- **Design Patterns**

Embrace Change!

- **Change is inevitable**
 - **Business: “adapt or die”**
- **Code calcifies**
- **Legacy code**
 - **Makes change risky**
 - **Leads to fear of change**



Recognise any of these Legacy Code symptoms?

- *“Code without (unit) tests”*
- *“Edit and pray” development*
- *“making changes feels like jumping off a cliff to avoid a tiger” - Mike Feathers*
- *“code providing capability that you’ve not yet paid for” – Harvey & Marks*

Refactoring

“a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its existing behaviour”

- Martin Fowler

Unit Tests

- *“Software without test cases must be assumed [to be] defective”* – DevWeek 2007
- **Test intent, not implementation details**
- **Must run fast – otherwise won’t be run**
- **Must only use local CPU and memory**
 - **NOT database, network, disk, config, ...**

SOLID Principles

- **S = Single Responsibility**
- **O = Open/Closed**
- **L = Liskov Substitution**
- **I = Interface Segregation**
- **D = Dependency Inversion**

Single Responsibility Principle

- **Every class should have:**
 - a single responsibility
 - a single purpose in the system
 - one reason to be changed
- **Naming difficulties indicate problems**
 - e.g. *do-stuff*, *execute-class*

Open/Closed Principle

- **Code should be open for extension but closed to modification**
- **Don't have to change much code to add new features**
- **e.g. add features through subclassing, Design Patterns and Lambda functions**

Let's DRY out our design

- *“One of the startling things that you discover when you start removing duplication is that designs emerge”*
- *“Duplication removal ... makes change faster and easier”*

- Michael Feathers

Dependency Inversion Principle

- **High-level or low-level modules should not depend on one another**
- **They should depend upon abstractions**
- **Typified by a strict layered architecture**
- **Demeter on Facebook: “Share with your friends, not your friend’s friends”**

And now: University of Life

“One must learn by doing the thing; for though you think you know it, you have no certainty, until you try” - Sophocles

Case Study 1

- Your colleague has just left for a fancy new consulting job
- He's left behind a part-written animation application which has been described as “*slow*” and “*buggy*”
- You've been asked to fix the code
- What do you do next?

First Impressions Count

- **Skim read the code**
 - **Why is all of this code in 1 very large file?**
 - **Where are the tests?**
 - **What does this strange bit of code do?**

Refactoring

- **Started to break the code up into separate files – helped me extract different sub-systems**
 - **Made code easier to understand**
 - **Enabled me to start defining interfaces between sub-systems**

Building test coverage

- **Started adding tests for each sub-system plus end-to-end tests exercising several sub-systems**
 - **Gave me more confidence that refactoring was safe**
 - **Again helped me understand what was going on**
 - **Enabled some of the bugs to be found and fixed**

Performance improvement

- **Separating code into different sub-systems made it clearer where all the ‘heavy lifting’ was being done**
- **Test framework made it possible to reliably change code to optimize it**
- **Overall the application still wasn’t great but the improvements did enough to keep our testers (and investors) happy**

Case Study 2

- **The technical ‘brain’ of the organisation has left the company**
- **He’s been working on v2 of an Object-Relational mapping library, which is being built into your flagship product**
- **The project is already 6 months late and you’ve ‘volunteered’ to make it work so the flagship product can ship.**
- **What do you do next?**

Another case of WTF?

- **A quick skim through the code told me that I didn't understand it 😞**
 - **Why does it have its own thread support?**
 - **Is this code too 'brainy' for my tiny mind?**
- **What's 'finished' and what isn't?**
- **Remind me again when it needs to ship...**

Who can help?

- **Application developers who've used the library**
- **Database developer who's written server end of application**
- **Some limited, out of date documentation**
- **No tests 😞**

What I did next...

- **Added (lots of) tests, (lots of) logging & (lots of) assertions**
 - check my understanding of the code
 - Make sure the code still works as it evolves
- **Move any application-specific code out of library into applications!**
- **Also told the boss not to expect a release any time soon.**

Large Scale Refactoring

- **Then spent a month refactoring the code**
 - **Planned how to do it for 1 week**
 - **Step-by-step worked through changes**
 - **Tested as I went, where changes coherent**
 - **The best 1 month's work I've ever done?**
 - **Removed ~500 Lines of Code**
 - **10X speed-up in database writes**

Case Study 3

- **“Finding needles in a haystack”**
- **3 month “gap filler” project in 2007**
- **Goal: extracting code from a huge* WinForms solution to extract components that can be reused for WPF prototypes**

Define your needles...

Simple solutions for the real world:

- **Use a magnet to extract the needles**
- **Or a flamethrower to burn the haystack**

... but complex for software:

- **The hay is magnetic**
- **The needles are flammable**
- **Everything is entangled**

People, Process

1st step: define the needles

- Other developers knew the backstory
- Challenge: what is in/out of scope?
- Hard to imagine reuse without a use
- Solution is to extract for prototypes
- Wiki to track conversations

Technology

2nd step: extract the needles

- **Discovery of nDepend**

Helped high-level view & low-level details

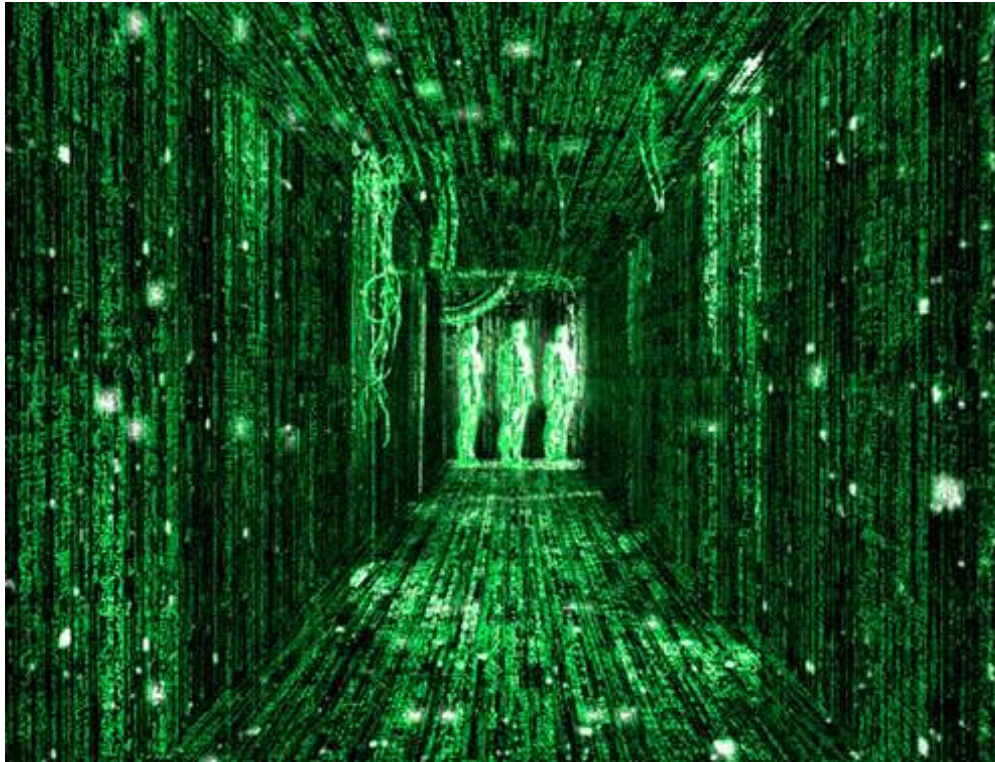
- **Where is the bulk of the complexity?**
- **What are the dependencies?**
- **What are the logical modules?**

Dependency

“Dependency is one of the most critical problems in software development.

Much legacy code work involves breaking dependencies so that change can be easier”
- Michael Feathers

From Anderson to Neo...



“Unfortunately, no one can be told what the Matrix is. You have to see it for yourself”

- Morpheus

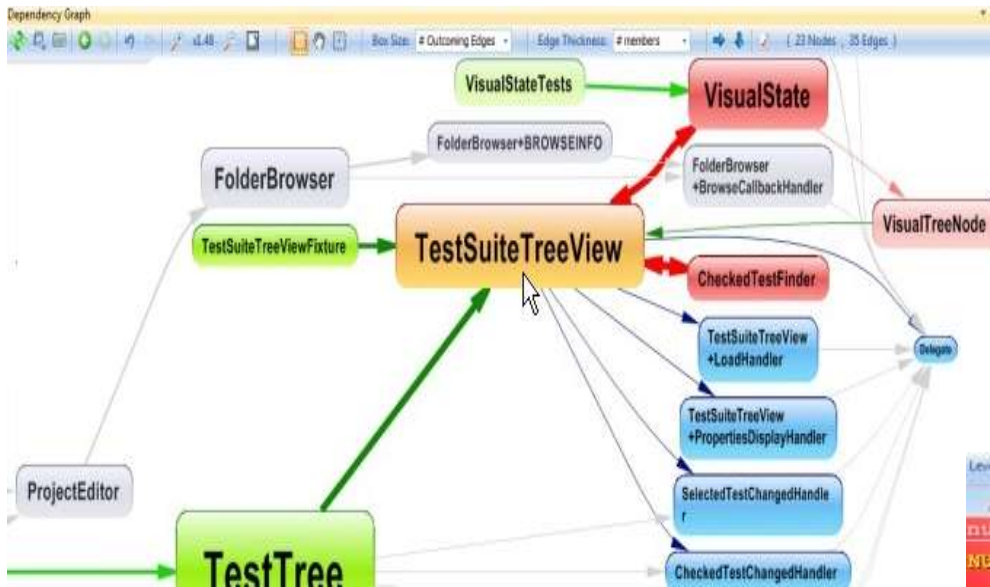
redgate®

ingeniously simple tools

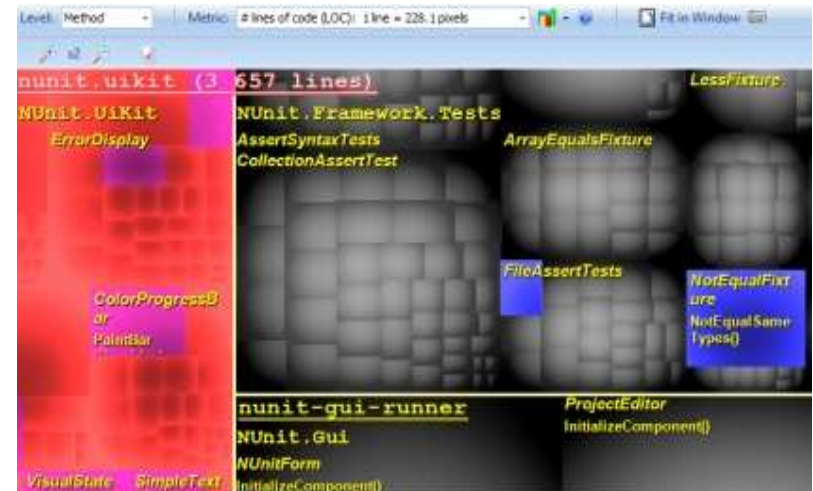
Dependency Matrix

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33			
VisualIDepend	0	1												1																							
KernelImpl	1	1	5	1		1	1					1			1	1											1	2	46	14	1			3			
MainPanel	2	13	2	1	2	1						1	1		1	2	1									1	1	31	8				4	0			
Drawer	3	1	8		10							1														1								2	8		
QueryResultPanel	4	12	25																																15		
Header	5	1	4									1						2	2			6				2									5		
Matrix	6	1	1					32	21	6	4		2	1	6											1										2	
Base	7							28		28		1	3	1												4										4	
BoxAndArrowGraphDrawing	8							24		32					20											7										3	
TreeCodePanel	9							3	18	17					0											2										2	
Tooltip	10							5							3											2										2	
GraphMatrixAlgorithm	11	1	1	2		1		2																												1	
QueryEdit	12			2				2	3																												7
Base	13	1						5	12						1																						2
Compilation	14							14		23	0	3			3																						0
Condition	15		1	1																																	0
Warn	16		1	1																																	1
Comparison	17																																				1
Top	18																																				0
Helpers	19																																				2
Async	20																																				0
InfoPanel	21																																				0
GraphLoader	22																																				4
KernelInterface	23																																				0
SourceCode	24	3	63	26	21	4	14	8	8	6		4	15			13	4	5	3			2	0				2	1	1							0	
Properties	25	1	16	12	15	20	13	13	24	13	0	5	29	3	3	16	0	0	179	77	178	81				0		1	179	0						8	
Base	26		3										2																								5
Helpers	27																																				116
Async	28																																				6
InfoPanel	29	1	11	3				4	6	10		14	5			8	3										3	0	0	0							
GraphLoader	30	2	0	8	25	3	5	4	8	6	8	2	9		0	1	1	12	0																		
KernelInterface	31																																				
SourceCode	32																																				
Properties	33																																				
Base																																					

Dependency Graph



Treemap



redgate®

ingeniously simple tools

Case Study 4

- **“Productionize that demo, pronto!”**
- **Became a 6 month team project in 2008**
- **Goal: (initially) review code from an overseas office that demonstrated integration of legacy and enterprise systems to a major potential customer**

“Big Cheese” expects a demo

- **“So, what does this code actually do?”**
 - **No demo, no documentation**
 - **No source control history – just a ZIP**
- **“Oh, the dev has left the country...”**
 - **He’s also a domain expert, unlike me...**
- **“Okay, let’s run it and step through”**
 - **Except it doesn’t even compile...**

It started with a K.I.S.S.

Get “something” running... quickly!

- **Focus on understanding the intent**
- **(Try to) ignore specific implementation**

Work out how to exercise functionality

- **No main() method**
- **Look for entry, branch, exit points**

Discuss and Document your Discoveries

Technology

- **nDepend analyses assemblies...**
 - **#ifdef “shotgun surgery” to get a build**
 - **Source Control Diff for your changes**
 - **Detach UI and focus on functionality**
- **“I’ll need to install more stuff...”**
 - **Prerequisite software and config is evil**
 - **Virtual Machine snapshots are awesome**

People and Process

- **Pair programming is good**
 - **Pair reverse engineering is essential**
 - **Grab a domain expert & don't let go!**
- **Use forensics to find context**
 - **Make time with the dev team**
 - **Get in their face but not up their nose**

I never thought it would come to this

Big Cheese:

- Now make it a Product...
- “But I expected it would be faster...”

If you can't delegate, get someone to do it

- Too much for one head, grow a team
- Watch out for Mythical Man Month

Technology

Define a vision for architecture

- **Use the tools to work out capabilities**
- **Design something you can build**

Diagrams are a series of sketches

- **Knowledge evolves**
- **Don't get hung up on UML notation**

People

- **Be the benevolent dictator**
 - You can't know it all, but it's your call
- **Domain Specialist scripts the demo**
- **SOLID code enables scaling out**
- **Direction, focus, aptitudes, interests**
- **Share knowledge with Code Review**

Process

- **Big Visible Charts: Stories, Tasks, Bugs**
- **Self-organising team**
- **MoSCoW prioritisation**
- **Each sprint ends on focused demo**
- **Push back against rewrites**
 - **Original code is the ONLY specification**

How to avoid this badge

- **Automated Build and Test on Server**
- **Check in Early & Check in Often**
- **Unit Test coverage & quality increase**



Let's take a moment to reflect

- 4 different case studies
- Your own experiences
- Your current project

Case Study 5

- **Red Gate's eCommerce systems**
- **Mostly in-house development**
- **Many people, many years**
- **There may be some legacy code ;-(**
- **Live systems, customers next door...**

Goal: Ingeniously Simple Systems

1. Where are we now?
2. How did we get here?
3. Where do we want to go?
4. How can we get there?
5. What obstacles are there?

What would you advise?

redgate®

ingeniously simple tools

Things to consider

Technology

- **Buy vs. Build**
- **Keeping pace with upgrades**

Process

- **Rewrite vs. Refactor**

People

- **Skills, interests, motivation**

Legacy Code Change Algorithm

From an initial understanding:

1. Identify change points (seams)
2. Find test points
3. Break dependencies
4. Write tests
5. Make changes and refactor

To the village hall!

redgate®

ingeniously simple tools

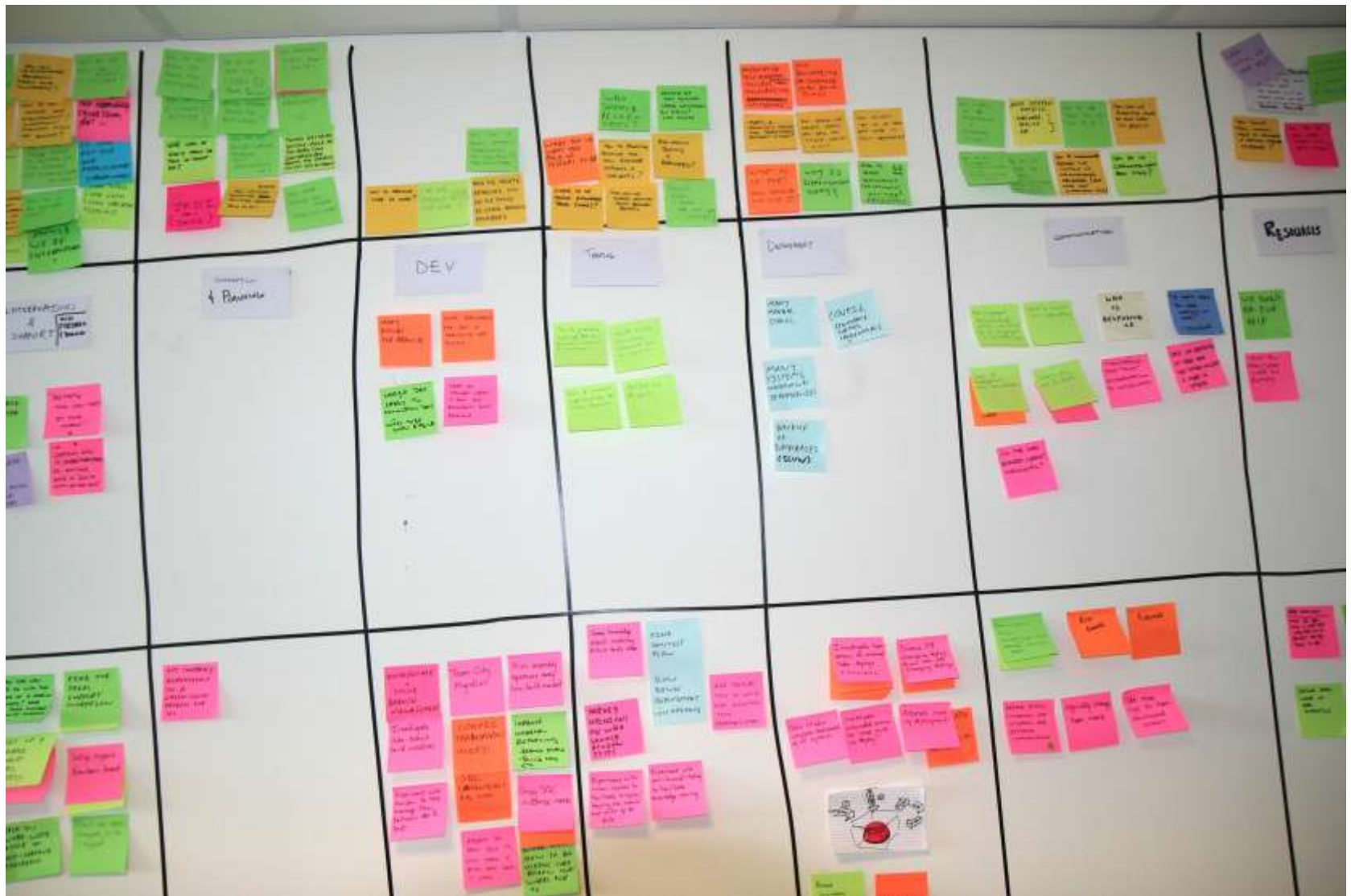












Keep in touch!



Mark Dalgarno

Head of DevOps

mark.dalgarno@red-gate.com

Twitter: @MarkDalgarno



Chris Morgan

Aspiring Architect aka. DevOps Dev

chris.morgan@red-gate.com

Twitter: @cjm55

redgate®

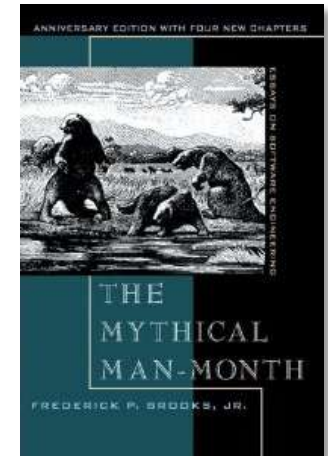
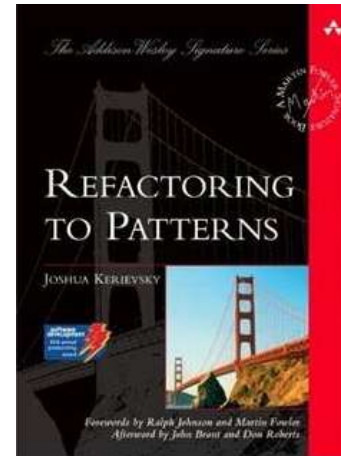
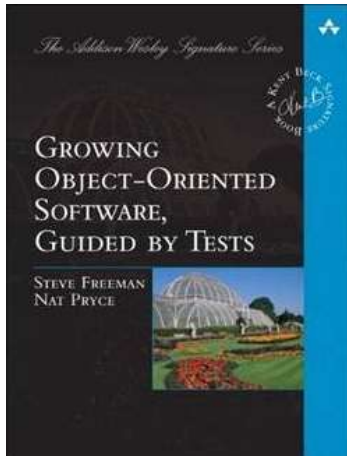
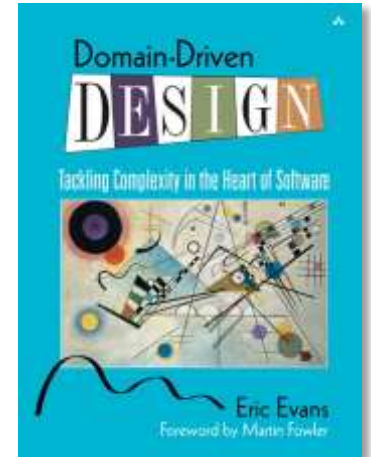
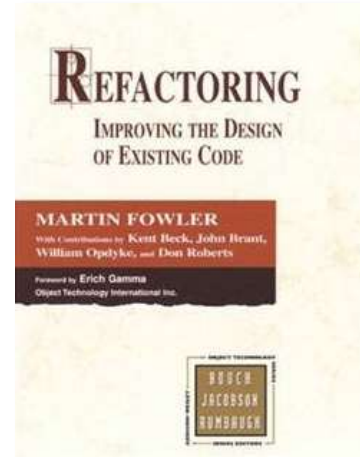
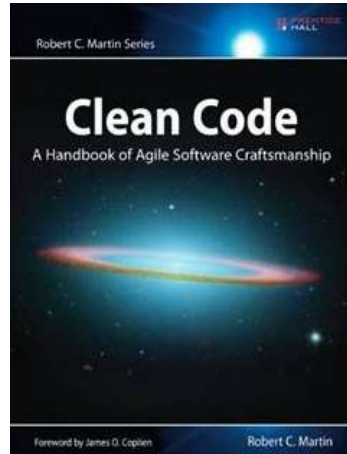
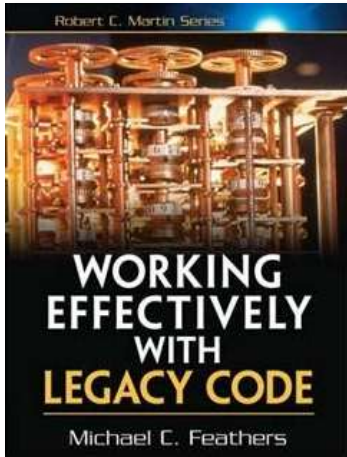
ingeniously simple tools

Thanks for participating!

**Don't forget to fill in the
feedback forms :-)**

redgate®

ingeniously simple tools



VS2010 Architecture Edition

- Introduced me to the ominous term “Death Star Architecture”
- Peter Provost - Architecture without Big Design Up Front (PDC 2008)
- <http://tinyurl.com/67tsds9>