

# How to build successful complex systems: lessons from open source

Les Hatton  
Kingston University, UK.\*

March 29, 2009

## 1 An overview

Complex systems appear on the surface to present special problems. Major failures in their operation or in their development are very frequently reported. In the UK, this prompted the Royal Academy of Engineering to produce a report, [5] in order to highlight the problem and hopefully promote a discussion into why such failures are so frequent. This eventually led to the formation of the Large Scale Complex IT systems institute here, after “wide-ranging discussions with relevant stakeholder groups in industry, academia, and the public sector to identify a programme of appropriate research and training”. Unfortunately I did not get the opportunity to provide any input to these discussions so this paper condenses thirty years of personal experience and observations into how systems get into trouble in the hope that it might be useful to somebody.

Note that the problems of large evolving software systems have prompted other countries into pursuing similar initiatives such as the Max Planck Institute for Software Systems in Germany and the Centre for Complex Software Systems and Services at Swinburne in Australia.

### 1.1 Observations of complex systems

Since the perennially fresh work of Fred Brooks [2], advice on dealing with large, unwieldy systems has been available. Indeed reading this book again more than thirty years after its first appearance, the reader is struck by how little seems to have changed. Many large systems still fail to appear at all and often for the same reasons. The UK publication Computer Weekly<sup>1</sup> has long kept a track on the rather dismal record of such systems in the UK.

When these problem systems are analysed, it becomes obvious that there are a number of common themes:-

- Nobody has an overall specific view of the system - the system appears out of intellectual control

---

\*L.Hatton@kingston.ac.uk, lesh@oakcomp.co.uk

<sup>1</sup><http://www.computerweekly.com/>

- Requirements are never better than hazy and change relatively often in the time-scale of the system
- Project planning and, above all tracking, is poor often because the tasks are too vague
- The systems tend to be designed top down mirroring frequently chaotic management hierarchies
- Users very often feel let down during the whole process even when it is claimed 'they have been consulted'. They are often bludgeoned into using a system rather than included in its design.
- There is often little continuity at any level as the systems can take a number of years to develop and deploy
- They usually use proprietary technology
- Requests for independent audit are nearly always rebuffed
- In the case of Government sponsored systems, they are often simply political vehicles without any thought of feasibility
- They are hugely expensive, usually costed in the hundreds of millions to billions of pounds

From a research point of view, these aren't even interesting. These are the same old human problems of over-ambition, inadequate training, management hierarchies which consider management to be more important than engineering, incomprehensible bureaucracy and so on. In software engineering, they are just writ large and flourish because of poor empirical underpinning from software system researchers. If a research institute could come up with anything useful, it might be to place software engineering on a similar footing to conventional engineering areas such as civil engineering. In civil engineering, we have the same management structures but no manager would try to force a civil engineer to do something dangerous or stupid, or if they did, legal retribution would swiftly follow. In software engineering, we are only dimly aware of what dangerous or stupid yet means.

## 1.2 An example: the NHS Connecting for Health Project

This is one of the most expensive IT projects of all time and is estimated at a total cost currently of around 12 billion pounds. It has had many problems and has repeatedly been requested to be subjected to external audit. This has so far always been refused. Some background behind this can be found here<sup>2</sup>. The NHS has responded by a its so-called mythbusters site<sup>3</sup> whose content is minimal to say the least. It certainly flies in the face of my own experience, for example in 2007, I personally interviewed 10 different NHS employees from nurse to consultant along with several GPs and all reported the same dissatisfaction. In several cases, this was extreme dissatisfaction.

<sup>2</sup><http://www.computerweekly.com/Articles/2007/09/13/226709/wanless-report-calls-for-audit-of-nhs-connecting-for.htm>

<sup>3</sup><http://www.connectingforhealth.nhs.uk/factsandfiction/mythbusters>

How might this have been handled differently ? We can get some clues from its handling in Wales which does not use the all-encompassing English system. Note the following quotations<sup>4</sup>:-

In the latest report, the chairman of the Commons public accounts committee, Edward Leigh MP, said: "Essential systems are late, or, when deployed, do not meet expectations of clinical staff; estimates of local costs are still unreliable; and many NHS staff remain unenthusiastic."

The article then goes on to compare implementations in England and Wales respectively as follows:-

The Welsh IHR draws data directly from GP records, with sensitive data such as terminations removed. Patients are asked for consent every time their record is viewed - unlike in England, which initially assumed patients to have given consent unless they explicitly opted out. The IHR was first tested at doctors' out-of-hours services in one county, Gwent, and then at one hospital, Royal Gwent. Crossing the gulf between GP and hospital is itself a breakthrough for NHS computing. It was made possible by technology from a specialist firm, Graphnet, which, because of its small size, was not eligible for any of the big English contracts.

The latter sentence above is particularly telling. In England, influence is far more important than competence.

In England, the NHS took it for granted that the right technology was available and that staff were enthusiastic about adopting it. The central challenge was seen to be procuring the technology on the best terms, and implementing it to timetable. This involved a series of billion-pound contracts to provide central services and to rip and replace hospital systems across five regions created solely for the IT programme.

It doesn't take much to see what is going on here from a systems point of view. Driven by grandiose top-down expensive ambitions, the English system flies in the face of all accepted wisdom of building software systems incrementally. In contrast, the Welsh system does precisely the opposite. In fact, deliberately or otherwise, the Welsh system is mimicking the very architecture which underpins open source.

### 1.3 Open source

Open Source software is obviously a very successful movement. The Linux kernel, Apache and a number of other initiatives have been very influential. They are comparably reliable to very high quality commercial systems, [3], [4] and have numerous other attractive features. The earliest writings I can find on this are by me [1] and, far more eloquently and comprehensively, Eric Raymond, [6] and later [7], (if I have missed any out, please let me know). Most of the questions which open source doubters pose such as support, reliability, security

---

<sup>4</sup><http://www.guardian.co.uk/technology/2009/jan/29/computing-nhs>

and so on are comprehensively answered by one or more of these sources. There is no longer any question as to their efficacy and today a substantial part of the web is powered by such technologies. The following all seem important contributors:-

- Open source permits large-scale code inspection and large-scale testing
- The users and the builders often include the same people
- Prototyping is extensive with many attempts made to build the same thing with the user base effectively voting on the best as I commented on here<sup>5</sup> and here<sup>6</sup>
- Perfective maintenance. This part of the maintenance cycle is simply trying to make the software better without changing the functionality. There are all kind of things that need to be done to an evolving piece of software and all experienced software engineers will know exactly what I mean by this. It is caused by that sinking feeling when you look at a piece of code and think 'Yuk'.
- Experience. The best systems are built by the best engineers. That's it. This appeared first in Fred Brook's masterpiece [2] and periodically ever since. It requires management structures which pay engineers to stay engineers rather than become inexperienced managers<sup>7</sup>.
- Apprenticeships. The nature of open source means that it is as if less experienced engineers are working alongside vastly experienced engineers. In other words, you serve an apprenticeship and 'promotion' in the open source world is based on meritocracy. There are strong indirect financial rewards for the best open source engineers if they want to take advantage of them simply due to visibility and reputation.

So how might we have applied all this knowledge ?

## 2 Case history: how the England Connecting for Health project might have been built

Let us consider competence first. Real expertise can be found throughout the UK higher education system. Many CS academics have significant industrial experience overlaid by researching these areas. I am quite representative and have built and maintain significant commercial systems<sup>8</sup> still. I also research into them. I also have some significant disasters under my belt which adds to my engineering experience immeasurably. There are lots of people like me in the higher education system.

---

<sup>5</sup><http://www.leshatton.org/A29.html>

<sup>6</sup><http://www.leshatton.org/A17.html>

<sup>7</sup><http://www.leshatton.org/A39.html>

<sup>8</sup><http://www.leshatton.org/>

It is fair to say that this expertise is rarely utilised successfully. In many universities including my own, computing system services are systematically outsourced rather than use any local knowledge. This means that my university mailbox is about half full of spam each day because the expensive service we outsource to is not as good as I am at maintaining mail servers, (mine admit around 1 spam message a week out of approximately 1,000,000 messages received and have lost no genuine messages; the spam load is around 99.9%). There are many examples of this but I won't labour the point any more. Simply take it as read that there are thousands of us in UK higher education institutes who are eminently capable of building very good systems.

There follows a multi-phase plan as to how a large system like Connecting for Health might have been built.

## **2.1 Phase 1**

There are approximately 120 CS departments in the UK in higher education. First give them all 100,000 pounds with a brief to build a patient record system connecting a randomly chosen general practice within 50 miles to a local hospital subject to the following ground rules.

- All software must be open source and developed on open source systems,
- All base technologies must be open source,
- All software must run on proprietary operating systems as well as open source systems,
- Each team has 12 months to implement a prototype,
- The best 20 projects will be given another set of funding,
- The software is free to any hospital or general practice wishing to use them but they must provide suitable PCs to run this software.

## **2.2 Phase 1: review**

Using a structure similar to the RAE, the systems will be evaluated. The audit would of course include extensive security auditing to allay any fears about prejudicing patient confidentiality.

## **2.3 Phase 2**

After 3 months of evaluation, a shortlist of 20 systems will be selected and extended to include another randomly chosen practice / hospital combination. This time a budget of 200,000 pounds will be given to each and a further 12 months.

## **2.4 Phase 2: review**

As with Phase 1, each prototype will be evaluated for 3 months for reliability, connectivity, security and completeness. A shortlist of 5 will be selected. Academics from teams already rejected will be allowed to vote on the best system.

## 2.5 Phase 3

This time each prototype will be extended to a region to link in at least 5 hospitals and 10 general practices. A budget of 500,000 pounds will be allocated to each project and a time frame of 12 months.

## 2.6 Phase 3: review

The review will be evaluated as before and reduced to 2 teams, one for the North of the country and one for the South. Academics from teams already rejected will be allowed to vote on the best system again.

## 2.7 Phase 4

This time, a budget of 1,000,000 pounds will be given and a time scale of 18 months to implement it. The goal will be to roll this out to 100 General Practices and 10 hospitals in each region, but it will not be forced.

## 2.8 Phase 4: review

The review will be evaluated blindly as before and a best system voted.

## 2.9 Phase 5

The successful team is selected and given a budget of 5,000,000 pounds and a time-scale of 18 months to re-engineer the winning system after studying the best 20 systems. After delivery, the winning system will be rolled out only to those hospitals who want it.

## 3 What will this cost ?

The NHS *Connecting for Health* system has currently cost around 12 billion pounds and has taken 8 years and is scarcely rolled out at all. The above scheme would cost 25.5 million pounds plus auditing costs and take 6.5 years. If it failed, then that's a pretty good indicator that no system could satisfy the conflicting requirements and only 25.5 million pounds would have been lost, (but probably not wasted).

## 4 Afterword

Note that the Welsh system does this on far less lavish scale. There was *no* budget and a single general practice was chosen to talk to a single hospital. It could fail but it very probably will not. When you build systems ground up, gradually adapting and using extensive prototyping and mirroring successful systems such as the web, the results tend to work. You are surrounded by such technologies whenever you walk the web.

An even cheaper way of doing it is simply copy the Welsh model. It is true the English scheme is saddled with largely unworkable schemes like *choose and book*, but coming from a Government that thinks that identity cards can be successfully implemented without trouble whilst managing to lose through carelessness, incompetence and negligence, some 30 million personal records in 2008 alone, the Welsh scheme has much to offer.

The hard bit seems to be to stop politicians / managing directors and so on having delusions of grandeur and trying to map them into software. If we can't do that, then the cycle of failure and waste will go on, and on, and on.

Of course the modern reality is that 12 billion pounds is a drop in the ocean to what the banking system has managed to waste, gamble or otherwise mislay through greed, stupidity and poor governance and we will be paying this back for generations. I would like to think therefore that incrementally building open source systems for public infrastructure in the way I describe above, is even more important as we will no longer be able to afford grandiose, expensive schemes which are unlikely to provide any benefit. We may yet have something to thank the bankers for.

## References

- [1] L. Hatton. Why is linux so reliable ? *Proceedings of ASM 2000, San Jose, California*, April 2000.
- [2] F.P. Brooks Jr. *The Mythical Man Month*. Addison-Wesley, 1975. ISBN 0-201-00650-2.
- [3] Audris Mockus, Roy T. Fielding, and James Herbsleb. A case study of open source software development: the apache server. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, pages 263–272, New York, NY, USA, 2000. ACM.
- [4] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3):309–346, July 2002.
- [5] Royal Academy of Engineering. The challenge of complex it projects, 2004. Royal Academy of Engineering report, London, ISBN 1-903496-15-2.
- [6] Eric S. Raymond. *The cathedral and the bazaar*. O'Reilly, February 2001.
- [7] Eric S. Raymond. Eric raymond's home page, 2006. <http://www.catb.org/esr/writings/cathedral-bazaar/>.