All's Well That Ends
IT projects and how to survive them
Richard Bland
Watson Wyatt
GIRO 2009

All's Well That Ends - Agenda

- The project – who’s involved?
- Can you safely outsource your IT work?
- A brief tour of modern development environments
- Where did it all go wrong?

The Project

- Assumptions:
  - The project involves delivering some kind of large scale application, or a change to an existing application
  - It could be used internally or made available to clients / customers
  - You’re on the project team responsible for delivering the application
Who’s on the team?

- Project director / managers
  - Some sort of professional IT manager?
- Business sponsors
  - You
- Developers
  - Your in-house specialists, or
  - An outsourcing team

The tasks and decisions

- How do I communicate my business needs to the IT professionals?
- Should we try to develop in-house or employ an outsourcing team?
- Do I need to worry about the technology used?
- When do we stop?

The specification

- Options:
  - High-level business spec
    - (this is what I want – just get on with it)
  - Detailed object model design
    - (I want to see exactly how the internal design works)
  - Prototyping
    - (I don’t know what I want, but I’ll know when I see it)
The developers

- You can control:
  - Costs
  - Quality
  - Delivery
  - … but not all of them at the same time

Outsourcers

- A straight choice of:
  - Fixed price contract
    - Controls cost, but the specification has to be absolutely nailed down, and the developers will cut corners wherever possible
  - Time and materials
    - Controls quality — you can make them build the application the way you want, but the bill will rise at an alarming rate

Outsourcers

- You can hire specialists in a particular field
- You can hire cheaper developers in another country
  But
- You will need a really well written specification
- You will also need a project manager to manage their project manager
Outsourcers

- Best for
  - A specialist project which you couldn’t build yourself
  - A straightforward, well-defined piece of work requiring no knowledge of the business
- Not for
  - A project which has to integrate with other systems
  - A project where prototyping will have to play a part

A brief glance at modern development

- Previous generations of applications were single-layer designs running directly on personal computers or mainframe/mini systems with terminals – a direct interface to the user
- Modern applications are multi-layer designs running on PCs or servers, with an interface separated from the business layer and storage

The old game

- Languages
  - C / Fortran / Pascal / Basic …
  - Procedural languages compiled into executable programs
  - Local execution only
- Data
  - Stored in flat binary files, system and application specific
The new game

- Languages
  - C# / VB / Java
  - Object-oriented languages with a rich class structure, which compile to intermediate code running in a platform-independent common run-time
- Data
  - Accessed through data-linked objects via data providers from local or remote databases

Web applications – the old way

- Webserver acts as a file server
- Delivers static .htm files, possibly containing Java functions for the browser to execute
- Functionality limited by browser capability
- The browser does all the work

Web applications – the new way

- Webserver acts as an application server
- The application runs on the server, executing program code there which builds web pages on the fly
- The browser acts as an interface between the user and the application running on the server
- The server does most of the work
The consequences of modern development

- Platform convergence
  - The same application can now be available as a Windows or web application
- Separation of layers – separate teams can
  - design the forms
  - design the business layer and objects
  - design the database / persistency layer

So where did it all go wrong …?

- Specification creep
- Quality control
- Deadlines
- Knowing when to stop

Specification creep

- Arises from an incomplete specification and optimistic planning
- Business users frequently fail to mention the “obvious” requirements
- Builds turn out to have usability flaws
- A requirement for legacy support
Quality control

- You want:
  - The best possible performance
  - Easily maintainable code
  - A friendly interface
- Developers want:
  - A solution that’s quick and easy to program
  - To recycle something they’ve done before

Deadlines

- No Gantt chart survives contact with the enemy
- The developer's deadline
  - To complete development the day before release
- The tester’s deadline
  - To complete testing the day before release

Knowing when to stop

- You want a complete, tested, bug-free application released on time.
- In practice, you must decide:
  - How late a delivery you can get away with
  - How many bugs you’re prepared to tolerate
  - Which features you’re prepared to leave until the next release
Knowing when to stop

- If you set specific limits on all of these, you may never satisfy them all
- You can design a utility function which combines them – when this reaches an optimal level, you just st

Contact details

- Richard Bland
  - Watson Wyatt Ltd
  - +44 1737 274541
  - richard.bland@watsonwyatt.com

All’s Well That Ends
IT projects and how to survive them

Richard Bland
Watson Wyatt
GIRO 2009