The Rise of the Methodological Approach

• The term "software crisis" was coined during the NATO Software engineering conference of 1968 to indicate that software projects often ran late, cost far more than expected and frequently did not meet the needs of the client.

• More than half a century ago in "The Mythical Man-Month" a book written "in his own blood," Fred Brooks reminded us that the only unforgivable failure is the failure to learn from our previous mistakes, and yet the software crisis persists. In the new millennium, software is still difficult to develop and information systems still frequently fail to meet user expectations.

Brooks, F (1975), The Mythical Man Month: Addison Wesley.
The Software Crisis

• Research by the Standish Group in 1994 produced disappointing statistics for US information systems development projects. During the early 90's, more than $250 billion was spent each year in the US, on the development of around 175,000 projects. The average cost of a development project was $2.3 million for a large company, $1.3 million for a medium company and $434,000 for a small company.

• The Standish Group research showed that over 30% of these projects were cancelled before completion. That 53% cost nearly twice as much as their original estimates and that well over half failed to meet anything like the user requirements. The average project was 189% over budget, 222% behind schedule and contained only 61% of the originally specified features.

The State of Contemporary UK IT Projects

• Research based on data collected about 421 projects from 1,500 practising IT project managers between October 2002 and January 2003.

• Report produced by Chris Sauer and Christine Cuthbertson of Templeton College, University of Oxford.

• Identifies typical project features, ‘success rates’ and risk factors.
The Financial Services sector’s project intensity with more than 20% of the total is consistent with what we know about the typical IT spend against other sectors.

For example, an IT spend of 10% of total company expenditure has not been uncommon in the FS sector. By comparison, manufacturers may spend as little as 0.5-1.0% on IT.

We might therefore expect in the order of ten to 20 times as many projects in Financial Services as in manufacturing.
Figure 2: Differences between private and public sector IT

<table>
<thead>
<tr>
<th>Successful private sector projects</th>
<th>Public sector projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focussed on measurable financial and service outcomes</td>
<td>Have multiple aims, so hard to measure success</td>
</tr>
<tr>
<td>Business driven by competition</td>
<td>Generally not in competition with other projects</td>
</tr>
<tr>
<td>Often not visible to the public or shareholders</td>
<td>Highly visible to the public and the media</td>
</tr>
<tr>
<td>Less constrained by legislation and regulations</td>
<td>Constrained by UK and EU legislation</td>
</tr>
<tr>
<td>Open to risk taking</td>
<td>Managed in a risk averse culture</td>
</tr>
<tr>
<td>Designed to limit damage when they are in difficulty</td>
<td>Difficult to adapt to change because of scale and complexity</td>
</tr>
<tr>
<td></td>
<td>Likely to interact with other departments</td>
</tr>
</tbody>
</table>

Source: Getting IT Right for Government, Intellect (formerly the Computing Services and Software Association), June 2000
Chaos Reports for the ‘noughties’

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Successful</td>
<td>34%</td>
<td>29%</td>
<td>35%</td>
<td>32%</td>
<td>37%</td>
</tr>
<tr>
<td>Challenged</td>
<td>51%</td>
<td>53%</td>
<td>46%</td>
<td>44%</td>
<td>42%</td>
</tr>
<tr>
<td>Failed</td>
<td>15%</td>
<td>18%</td>
<td>19%</td>
<td>24%</td>
<td>21%</td>
</tr>
</tbody>
</table>

Project resolution results from CHAOS research for years 2002 to 2010.
# Top Project Risk Factors

- Lack of top management commitment
- Misunderstanding of scope/objectives/requirements
- Lack of client/end-user commitment/involvement
- Changing scope/objectives
- Poor planning/estimation
- Inadequate project management
- Failure to manage end-user expectations
- Conflict among stakeholders
- Change in senior management ownership
- Lack of adequate change control
- Shortage of knowledge/skills in the project team
- Improper definition of roles and responsibilities
- Artificial deadlines
- Specifications not frozen
- New or radically redesigned business process/task
- Employment of new technology
- Poor control against targets
- Number of organisational units involved
- **Lack of effective methodologies**
- Staff turnover
A 2001 survey across all sectors published by the BCS

- Found that only around one in eight IT projects (13%) were successful (i.e. delivered on time, cost and to specification).

- For development projects (rather than maintenance or data conversion projects) the figure was even worse, with less than 1% succeeding.

- The Royal Academy of Engineering and the British Computer Society are currently conducting a study on the challenges of complex software projects, which aims to provide recommendations for increasing the likelihood of success.

*IT projects sink or swim*, Andrew Taylor, British Computer Society Review, 2001
<table>
<thead>
<tr>
<th>Dimension</th>
<th>'Hard' Rational Design</th>
<th>'Soft' Political Reality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
<td>Emphasis on standardised, formal, quantitative information</td>
<td>Emphasis on contingent, informal, qualitative</td>
</tr>
<tr>
<td>Technology</td>
<td>Simple enabling mechanism</td>
<td>Complex, value-laden entity: status symbol for some, tool of oppression for others</td>
</tr>
<tr>
<td>Processes</td>
<td>Stable, straightforward and formal; decision outcomes as optimal solutions based on logical criteria</td>
<td>Flexible, complex, constrained and often informal; decision outcomes based on ‘power games’</td>
</tr>
<tr>
<td>Objectives and values</td>
<td>Formal organisational</td>
<td>Multiple, informal, personal</td>
</tr>
<tr>
<td>Staffing and skills</td>
<td>Staff as rational beings</td>
<td>Staff as political beings</td>
</tr>
<tr>
<td>Management systems</td>
<td>formal, objective processes and structures</td>
<td>Informal, subjective processes and structures</td>
</tr>
<tr>
<td>Resources: e.g. time and money</td>
<td>Used to achieve organisational objectives</td>
<td>Used to achieve personal objectives</td>
</tr>
</tbody>
</table>

Brooks’ ‘Essential’ Difficulties of Software Development

• Complexity
• Conformity
• Changeability
• Invisibility
Complexity

• IT projects can be very complex, with millions of lines of computer code and perhaps billions of execution paths.

• The ‘Governing Mode of Failure is hard to predict’

• It is often not possible to calculate accurately the difficulty of such projects before they have started. Estimation tools give widely varying results and most suppliers rely on previous experience, which is necessarily very subjective - and can be misleading. (as Phillips discovered to their cost)
Conformity

• Systems need to adapt to many different (ever-changing) user requirements

• It is difficult for management (especially non-technical management) to judge the quality or completeness of software as it is being developed. Providing oversight in the years between awarding a large contract and the delivery date can therefore be problematic.
Changeability

• Software generally outlives Hardware

• Upgrades, enhancements, improvements, re-engineering are the norm for CBIS – not true of most branches of engineering.

• IT projects generally have interfaces with other systems, which may also be changing. Ensuring these systems interact successfully is often a major challenge, and without an overall plan new systems development can re-enforce differences between systems and services rather than helping to join them together.
Invisibility

• Software is a logical construct rather than a physical artifact
• Design Abstractions used are often very weak remember DD?
• SW development can be a game of ‘Chinese whispers’
Other Important Difficulties: Limited Skills

- Many software developers do not have formal qualifications in Software Development.

- The BCS has proposed that a better regulated profession is needed to ensure competency, quality and consistency.

- The BCS proposes that all bids for government IT contracts should be required to include the accreditation and qualifications of those who will be working on the project.

- However, there remains a significant IT skills shortage, and major government suppliers will be unable to deploy experienced developers on all projects.
Other Important Difficulties: Fast moving technology

- IT projects differ from other types of project in that the technology used is developing rapidly. This has a number of implications.

- Customers are often not familiar with the latest IT developments, so may be unable to judge whether suppliers are overselling a particular technology and the ease with which it can be delivered.

- Technological advances can make projects obsolete before they have been completed.

- There is a tendency to desire cutting-edge solutions, which carry greater risk, rather than use tested commercial 'off-the-shelf' products.
Other Important Difficulties: Defining requirements

• The BCS survey found that poor management of the requirements and scope of a project were the most common causes of failure. For IT projects, user requirements are often not clear at the start, for a number of reasons:
  – Users may be unsure of what they want
  – It may be difficult to identify their tacit knowledge about day-to-day processes
  – They may not have been consulted sufficiently
  – User requirements may be misunderstood.
  – Departmental and strategic requirements may be poorly defined.
  – External factors can cause requirements to change.
  – A 'simple' change to requirements may require a fundamental redesign of the system, with large time and cost implications.

• According to the BCS study, three quarters of IT project managers reported that in their experience no project had ever been delivered to the initial specifications.
So we probably need ‘good’ Project Management

- e.g. PRINCE2 sets out a series of processes which cover all the activities involved in a project, from start-up to close. It attempts to define each process, detailing its inputs and outputs, objectives and activities. It specifies the roles and responsibilities for managing a project, including setting up a project board with representatives from the customer, user and supplier. The method also explains how to manage risk, quality and change. Overall, PRINCE2 aims for projects to have:

  - A controlled and organised start, middle and end
  - Regular reviews of progress against plan and against the Business Case
  - Flexible decision points
  - Automatic management control of any deviations from the plan
  - The involvement of management and stakeholders at the right time and place during the project
  - Good communication channels between the project, project management, and the rest of the organisation.

Source: www.prince2.org.uk
We probably need ‘good’ Systems Development Methods

• A Taxonomy of Approaches to Systems Development
  
  – Participative/Socio technical Approach. (Mumford, 1979; Lundeberg, 1982)
  – Traditional Approach. (NCC, 1985)
  – Structured Systems Approach. (Yourdon & Constantine, 1979; De Marco, 1979; Jackson, 1983,)

    (Avison & Fitzgerald, 1995)

  – More recent approaches like UML/RUP; SCRUM/XP represent a seventh approach, which is much more closely allied to 4, 5 and 6, than to 1, 2 or 3.
Advantages of using a Structured Development Methodology are cited as including:

• Standard basis for development
• Precision and non ambiguity of specification
• Rigour in development
• Clarity of communication & interfaces between design groups and users
• Entrapment of errors at design stage through design testing and reviews
• Reduction in development time and cost
• Improved control over the development process
• Increased portability through design documentation and modularisation
• Easing of the maintenance burden
The Waterfall Model

W.W. Royce
SSADM

- SSADM is one particular implementation and builds on the work of a number of leading academic-practitioners including:
  - Peter Checkland  Soft Systems Methodology
  - Tom DeMarco  Structured Analysis
  - Ed Yourdon & Larry Constantine  Structured Design
  - Michael A. Jackson  Structured Programming
History of SSADM

- 1980 Central Computer and Telecommunications Agency (CCTA) evaluate analysis and design methods
- 1981 LBMS method chosen from shortlist of five
- 1983 SSADM made mandatory for all new information system developments
- 1984 Version 2 of SSADM released
- 1986 Version 3 of SSADM released, adopted by NCC
- 1988 SSADM Certificate of Proficiency launched, SSADM promoted as ‘open’ standard
- 1989 Moves towards Euromethod, launch of CASE products certification scheme
- 1990 Version 4 launched
- 1993 SSADM V4 Standard and Tools Conformance Scheme Launched
- 1995 SSADM V4+ announced, V4.2 launched
The SSADM method involves the application of a sequence of analysis, documentation and design tasks concerned with:

- Analysis of the current system
- Outline business specification
- Detailed business specification
- Logical data Design
- Logical process design
- Physical design
Components of SSADM

• Structures
  – define the frameworks of activities, steps and stages and their inputs and outputs

• Techniques
  – define how the activities are performed

• Documentation
  – define how the products of the activities, steps and stages are presented
SSADM Techniques and Models

• Logical Data Models
• Data Flow Models
• Requirements Definition
• Function Definition
• Specification Prototyping
• Relational Data Analysis
• Entity/Event Modelling (Entity Life Histories and Effect Correspondence Diagrams)
• Business and Technical Options
• Dialogue Design
• Update and Enquiry Process Models
• Physical Data Design
• Physical Process Specification
• Physical Design Control
Feasibility Study in SSADM

Prepare for the Feasibility Study

Define the problem Situation

Define the Required System

Select Feasibility Options

Assemble the Feasibility Report

Rich picture (SSM)
Requirements (SSADM)

Root Definitions (SSM)
Conceptual Models (SSM)
Consensus Primary Task Models (SSM)
Entity & Relationship Definitions (SSADM)

Maltese Cross (SSM)
Organisation Mapping (SSM)
Data Flow Model (SSADM)
Logical Data Model (SSADM)
Requirements (SSADM)

Business Systems Options (SSADM)
Data Flow Model (SSADM)
Logical Data Model (SSADM)
Technical Systems Options (SSADM)
Complementary techniques

- Quality Assurance Reviews
- Formal Documentation
- Project Control Methods – PRINCE 2
- Use of CASE Tools – SSADM SELECT

- SSADM was for a number of years a recommended practice in the development of UK government information systems, along with the PRINCE2 method for project management.
The Spiral Model

B. Boehm

29
The Evolutionary Model

T. Gilb
Generic Agile Life cycle

- **Project Initiation**
- **Project Plan**
- **Elaborate Requirements**
- **Architecture**
- **Release**
  - Timebox
  - Timebox
  - Timebox
  - ...
- **Release**
  - Timebox
  - Timebox
  - Timebox
  - ...
- ...
Scrum

- Pitched as: "Management and control process that cuts through complexity"
- Key Names: Jeff Sutherland, Ken Schwaber, Mike Beedle.
- Where invented: USA
- Year first used: 1994
- First used on: Advanced Development Methods - process automation software. 8 developers. VMARK - OO software development environments.
- Now used on: All over the place with different groups/people.
Crystal Orange

• Pitched as: A method to run “a cooperative game of invention and communication, with a primary goal of delivering useful working software and a secondary goal of setting up for the next game”
• Key Name: Alistair Cockburn
• Where invented: USA (although he is a Brit)
• Year first used: Pre-1988
• First used on: Project Winifred; team of 20-40.
• Now used on: Not used again. A variant Crystal Orange Web used at eBucks.com.
XP

- Pitched as: Addressing “the specific needs of software development conducted by small teams in the face of vague or changing requirements”
- Key Name: Kent Beck.
- Where invented: USA
- Year first used: Pre-2000
- First used on: C3 project Chrysler; 8 developers.
- Now used on: All over the place by different groups/people.
DSDM

• Pitched as: “A framework of controls and best practice for rapid application development”
• Invented by: DSDM Consortium
• Where invented: UK
• Year first used: 1995
• First used on: Don’t know but been used at/by BT, Orange, Dept. of Health, Syndeco/Boston Globe, Sema Group, Logica and British Midlands.
• Now used on: All over the place with different groups/people.
Formality as Deliverables

- Scrum Backlog, Running Code
- XP Stories, Running Code, Tests
- Crystal Orange Release sequence; Schedule (user viewings and deliveries); Annotated use cases or feature descriptions; Requirements document (purpose, use cases, non-functional requirements, interface definitions); Design sketches and notes as needed; UI Design / Screen drafts; Common object model; Running code; Migration code; Test cases; User Manual; Status reports; Inter-team specs
- DSDM Feasibility Report; Outline Plan; Business Area Definition; Non-Functional Requirements List; Systems Architecture Definition; Development Plan; Functional Model; Functional Prototype; Design Prototype; Tested system; Delivered system; Implementation Plan; Development Risk Analysis Report; Review Records; Test records; User documentation; Project Review Document
A Scrum Sprint Backlog Chart
Agile Vision Statements

• XP uses a “Metaphor” as a product vision, for example, the new product is like a spreadsheet. This helps everyone understand basic elements of the product and their relationships.

• In many ways DSDM's “Feasibility Report” is a project vision as it defines general scope and objectives.

• Scrum's “Sprint Goal” is a timebox vision.
Agile High Level Requirements

• XP
  – Customer writes/collects Stories. The result is a pile of Story Cards. There is no separate Elaboration phase, although the Customer is expected to Elaborate as much as necessary at the start of the Iteration (Timebox).

• Scrum
  – Product Owner maintains Product Backlog. There is no separate Elaboration phase.

• Crystal Orange and DSDM
  – Facilitated workshop(s) to identify “High level requirements”. These are collated into a document (Requirements Document / Business Area Definition) that contains use Cases and non-functional requirements. There is a separate Elaboration phase to create these documents.
Planning Based on Timeboxes

• A **Release** is a piece of development where the customer gets some new software. Releases can be from 2 weeks to 6 months, but are usually 3 months long. Release have one or more timeboxes.

• A **Timebox** is 1 – 6 weeks long, but usually 3 – 4 weeks. The most important thing about a timebox is that the delivery date is fixed.

• Unlike other methods, in DSDM one Release is the norm, i.e. there is only one release to the customer in the entire project. DSDM is also unique in that it categorises timeboxes depending on their function: Investigate, Refine, Consolidate. There are more of the former at the start of a project and more of the later at the end.
Timebox Plan Built by Team

- An XP Iteration (timebox) has an “Iteration Planning Meeting” where the customer explains the Stories (requirements). The team list tasks and programmers sign up for tasks.

- Scrum has a “Sprint Planning Meeting” at the start of each Sprint (timebox). The Team picks Backlog that is achievable (the “Sprint Backlog”) and decides how to achieve Sprint goal within the Sprint.

- DSDM has an objectives setting meeting at the start of a timebox. Users reassess MoSCoW priorities, the team agrees quality criteria, and the team agrees minimum that can be delivered.
Small Cross-Functional Teams

<table>
<thead>
<tr>
<th>Concept</th>
<th>XP</th>
<th>Scrum</th>
<th>Crystal Orange</th>
<th>DSDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of teams</td>
<td>1 team per project</td>
<td>1 – 4 or more</td>
<td>Variable; up to 40 people so probably 1 – 10 or so.</td>
<td>1 – 6</td>
</tr>
<tr>
<td>Team size</td>
<td>3 – 16</td>
<td>5 – 9</td>
<td>4 – 8</td>
<td>2 – 6</td>
</tr>
<tr>
<td>Team Members / Roles</td>
<td>Customer, Programmer, Tester, Tracker, Coach</td>
<td>Scrum master, Experienced Engineer, Junior Engineer, [QA Tester], [Writer]</td>
<td>Business Analyst-Designer, Designer-Programmer, UI Designer, [Tester ]</td>
<td>Team Leader, Ambassador User, [Advisor User], Senior Developer, Developer, Scribe</td>
</tr>
<tr>
<td>Project Roles</td>
<td>Big Boss</td>
<td>Project Manager/Scrum master, Product Owner</td>
<td>Sponsor, Project Manager, Architect, Technical Facilitator, Design Mentor</td>
<td>Visionary, Executive Sponsor, Project Manager, Technical Co-ordinator, Facilitator</td>
</tr>
</tbody>
</table>
Review and Reflection

• XP and Crystal Orange suggest the teams be self adapting. XP does this with 1 – 2 hours reflection each Iteration (Timebox);

• Crystal Orange has a “Team Reflection Workshop” in middle and at end of each Increment (Release).

• There is also the issue of the next project - what would it do? This is addressed by a Story Pile (XP), Product Backlog (Scrum), or Project Review Document (DSDM).
Predictive v Adaptive
Some Key Issues

• Scope – breadth & depth of organisational effect
• Size – time, HR, number of functions, transaction volumes
• Complexity – systemic/organisational, deterministic/algorithmic
• Nature – of project, organisation, environment
• Volatility – requirements, markets, technologies
Predictive V Adaptive

• Adaptive
  – Uncertain or volatile requirements
  – Responsible, talented and motivated developers
  – Customer who understands and is willing to commit to the success of the project

• Predictive
  – A large development team (say 100+)
  – A fixed price, fixed scope contract

Martin Fowler, *The New Methodology*,
Found at [http://martinfowler.com/articles/newMethodology.html](http://martinfowler.com/articles/newMethodology.html)
A Pragmatic Approach

• Academics like Mike Jackson (M.C. not M.A.) support the pragmatic use of systems ideas, while remaining critical of pure pragmatism.

“...Pragmatists, therefore do not worry about ‘artificial’ theoretical distinctions. They concentrate on building up a ‘tool kit’ of techniques that can be used as required of the real-world situation. Proven techniques from different strands ... are employed together in the course of problem solving if the situation warrants it. The choice of techniques and the whole procedure is justified to the extent that it brings results in practice...Systems people should be ‘activist’ seeking out problems that should be tackled using systems ideas. Available theory should be used pragmatically and eclectically...”

Closing Thoughts on Method

• “A mature citizen is not a man who has been instructed in a special ideology, ...and who now carries this ideology with him like a mental tumour, a mature citizen is a person who has learned how to make up his mind and who has then decided in favour of what he thinks suits him best. He is a person who has a certain mental toughness (he does not fall for the first ideological street singer he happens to meet) and who is therefore able consciously to choose the business that seems to be most attractive to him rather than being swallowed by it.”

• “There is no special method that guarantees success or makes it probable. Scientists (for example) do not solve problems because they possess a magic wand - methodology, or a theory of rationality - but because they have studied a problem for a long time, because they know the situation fairly well, because they are not too dumb (though that is rather doubtful nowadays when almost anyone can become a scientist), and because the excesses of one scientific school are almost always balanced by the excesses of some other school. (Besides, scientists only rarely solve their problems, they make lots of mistakes, and many of their solutions are quite useless.)”

Feyerabend, P (1975) Against Method, Humanities Press

Are Software Developers any different?
READING

• Read Martin Fowler’s paper ‘The New Methodology’
  http://martinfowler.com/articles/newMethodology.html and compare and contrast SAD with AGILE through the consideration of:
  
  – Their structure, ethos and applicability.
  – The degree to which they might help to address the top 20 project risk factors. Slide 7
  – The manner in which they help to ameliorate Brooks’ essential (and other important) software development difficulties. Slides 16-23