Consideration of Dynamic Systems Development Method (DSDM) and eXtreme Programming (XP)

“Holistic approaches to software development embracing the principled of RAD project environment”

“Delivering Agile Business Solutions on Time”

How user involvement can work in practice
Objectives

- Introduce DSDM
- Discuss benefits and issues
- Identify skills and techniques
- Consider DSDM in relation to management and professional issues
- Extend DSDM to consider eXtreme Programming
Agile Methods

- DSDM and XP are "agile methods"
- Other agile methods include Adaptive Software Development (ASD), Crystal, Scrum, and Feature Driven Development (FDD)
- Agile methods are adaptive rather than predictive
  - Unlike other engineering methods, agile methods welcome change.
- Agile methods are people oriented rather than process oriented. Agile methods assert that no processes will ever make up the skill of the development team, so the role of the process is to support the development team in their work
- All agile methods centre around small iterations
Agile Software Development

- **Individuals and Interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan
DSDM

The Dynamic Systems Development Method (DSDM) is a public domain Rapid Application Development method which has been developed through capturing the experience of a large consortium of vendor and user organisations.

It is now considered to be the UK's de-facto standard for RAD.
DSDM History

- 1994 – DSDM consortium formed
- 1995 – DSDM Version 1.0 released
- 1996 – DSDM Version 2.0 released
- Late 1997 – DSDM Version 3.0 released
- Early 2001 – e-DSDM Version 1.0 released
- Autumn 2001 – DSDM Version 4 released
- 2002 – DSDM Version 4.1 released
- Spring 2003 – e-DSDM Version 2.0 released
- Summer 2003 – Version 4.2 released
- Autumn 2004 – 10th Anniversary Conference
Development of eXtreme Programming

- Roots of XP lie in the Smalltalk (programming language) community
- XP evolved as an informal practice in the early 1990s
- 1996 formalised into a methodology
  - (working on a payroll project for Chrysler which went live in 1997)
- 2000 – eXtreme Programming Explained is published
Software Development Environments

- Initiation
- Development
- Testing
- Live
- Maintenance (often used as a parallel to live)
The key to DSDM is to deliver what business needs when it needs it.

- Achieved by using the various techniques in the framework and flexing requirements.
- The aim is always to address the current and imminent needs of the business rather than to attack all the perceived possibilities.

A fundamental assumption of DSDM is that nothing is built perfectly first time, but that a usable and useful 80% of the proposed system can be produced in 20% of the time it would take to produce the total solution.
eXtreme Programming Focus

- Like DSDM seeks to address problems of software development failing to deliver
- Uses development of code as main driver for development
- Examines the way we manage:
  - cost, time, quality and scope
- Incorporates four values
  - communication, simplicity, feedback, courage
- Promoting principles of
  - rapid feedback, assume simplicity, incremental change, embracing change, quality work
Software Development

- Many writers argue that software development fails to deliver product and fails to deliver value
- Failure of software development has huge economic and human impact
- Agile methods seek address the issues of failure
Why systems fail

- The system fails to meet the business requirements for which it was developed. The system is either abandoned or expensive adaptive maintenance is undertaken.
- There are performance shortcomings in the system, which make it inadequate for the users’ needs. Again, it is either abandoned or amended incurring extra costs.
- Errors appear in the developed system causing unexpected problems. Patches have to be applied at extra cost.
- Users reject the imposition of the system, for political reasons, lack of involvement in its development or lack of commitment to it.
- Systems are initially accepted but over time become unmaintainable and so pass into disuse.
Risk: the Basic Problem

Beck (2000) argues that the basic problem of software development is risk and identifies the following examples of risk:

- Schedule slips
- Project cancelled
- Systems go sour – needs to be replaced after a short period in a live environment
- Defect rate – put in to production but never used
- Business misunderstood
- Business change
- False feature rich – from user and developer
- Staff turnover

Return to XP addresses these issues later
RAD Lifecycle

- Delivers a fully functional system in 90 days, give or take 30 days
- Phases
  - Requirements Planning
  - User Design
  - Construction
  - Cutover
- Essential components:
  - JAD,
  - Evolutionary Prototyping,
  - Tool Support
Criteria for RAD success

- Management commitment
- Use of evolutionary prototyping
- User involvement throughout
- Appropriate use of tools
- Use of standards
The CASE for RAD

- Reduced development time
  - time-boxing;
  - concurrent development;
  - evolutionary prototyping;
- Lower cost
  - reduced development time;
  - less maintenance
- Higher quality
  - more user involvement;
  - more emphasis on requirements specification;
  - focus on product
The CASE against RAD

- Over-hyped, misunderstood
- Set-up costs often underestimated
- Getting the right people involved
- Need for commitment to the process
- Danger of inappropriate application
- Can reduce quality through lack of rigour
DSDM Ethos

- A fundamental assumption of the DSDM approach is that **nothing is built perfectly first time**, but that 80% of the solution can be produced in 20% of the time it would take to produce the total solution.

- In “traditional” development practice, a lot of time is spent in getting from the 80% solution to the total solution, with the assumption that no step ever needs to be revisited. The result is either projects that are delivered late and over budget or projects that fail to meet the business needs since time is not spent reworking the requirements.

- DSDM assumes that all previous steps can be revisited as part of its iterative approach. Therefore, **the current step need be completed only enough to move to the next step**, since it can be finished in a later iteration.
Benefits of using DSDM

- Using an iterative process based on prototyping, DSDM involves the users throughout the project life cycle

- Gives the benefits of:
  - early implementation to business problems
  - users more likely to accept ownership of the computer system
  - risk of building the wrong computer system is reduced
  - the final system is more likely to meet the users’ real business requirements
  - IT professionals and end users become partners
  - the users will be better trained, since their representatives will define and co-ordinate the training required
  - implementation is more likely to go smoothly, because of the co-operation of all parties concerned in development
  - empowerment
DSDM Organisation

Senior Management Board

Executive Sponsor

Project Steering Committee

User Management

Development Project

End Users, including Advisor Users

Operations

Project Roles
- Project Manager
- Technical Co-ordinator
- Visionary

Team Roles
- Team Leader
- Ambassador User
- Developer, Scribe, Tester
Traditional methods versus DSDM

- **Average time to delivery (in months)**
  - Using traditional approaches: 18-24
  - Using DSDM: 4-6

- **Average project team size**
  - Using traditional approaches: 11
  - Using DSDM: 5

- **% of completed projects rated good to excellent**
  - Using traditional approaches: 77%
  - Using DSDM: 87%

Source: British Airways IM Department, Newcastle
DSDM Principles

1. active user involvement is imperative
2. DSDM teams must be empowered to make decisions
3. focus is on the frequent delivery of products
4. need to measure fitness for business purpose
5. iterative and incremental development is required
6. all changes during development are reversible
7. requirements are base lined at a high level
8. testing is integrated through the lifecycle
9. a collaborative and co-operative approach between all stakeholders is essential

See seminar notes, Stapleton (1997, 2003) and DSDM website (www.dsdm.org) for details on principles
DSDM Process Overview

- **Feasibility**
  - Business Study

- **Design & Build Iteration**
  - Agree Schedule
  - Create Functional Prototype
  - Review Prototype
  - Identify Functional Prototype

- **Implementation**
  - Review Business
  - Implement
  - User Approval & User Guidelines
  - Train Users

- **Review Prototype**
  - Functional Model Iteration

- **Design Prototype**
  - Agree Schedule
  - Design & Build Iteration
  - Create Design Prototype
  - Review Design Prototype
User centred techniques in DSDM

- User analysis
  - identify user population for the proposed system
- Usability analysis
  - determine characteristics of user interface
- Task modelling
  - identify business events (user tasks)
- Task scenario Definition
  - identify instances of task execution for a user
- User conceptual modelling (user object modelling)
  - provide a map of the system from the user’s perspective
- GUI design
  - user interface to support identified tasks
- User interface prototyping
  - provide animated view of proposed system
Introducing DSDM to an organisation

- Questions to raise in the change of culture
  - How are projects currently staffed?
  - What responsibility and authority do project managers have?
  - Current environment one of consensus or control?
  - How will people react to change in working practices?
  - How mobile are staff in an organisation?
  - Can workshops be accommodated?
  - What is the current relationship with users?
Functional Model Iteration

- Produces standard analysis, but also software.

- Cycle
  - Identify what is to be produced
  - Agree how and when to do it
  - Create the product
  - Check that it has been produced correctly

- Software aimed at function
- Testing takes place
Design and Build Iteration

- Computer system is engineered to a suitably high standard
- Major product is a tested SYSTEM
- Includes non-functional requirements
- Cycle
  - Identify what is to be produced
  - Agree how and when to do it
  - Create the product
  - Check that it has been produced correctly
- Only agreed parts due to time constraint
Implementation

- Cutover from development environment to operational environment
- Training of users
- Documentation is completed
Difference between traditional development and DSDM

- **Traditional**
  - Time: Vary
  - Resources: Vary
  - Functionality: Vary

- **DSDM**
  - Time: Fixed
  - Resources: Fixed
  - Functionality: Fixed
Critical success factors in DSDM

- acceptance of DSDM philosophy before starting work
- the decision making powers of the users and developers in the development team
- commitment of senior user management to provide significant end-user involvement
- incremental delivery
- easy access by developers to end-users
- the stability of the team
- development team skills
  - in tools and business knowledge
- size of the development team
- supportive commercial relationship
- development technology
Selecting projects for DSDM

- Care should be taken that the right sort of projects are selected.
- DSDM is particularly well-suited to business applications but has been used with considerable success in engineering system development.
Characteristics of systems where DSDM can be used

- Interactive systems, where functionality is clearly demonstrable at the user interface
- Systems with a clearly defined user group
- In complex system, systems that allow for the complexity to be decomposed or isolated
- Systems that are time constrained
- Systems where requirements can be prioritised
- Systems where the requirements are unclear or subject to frequent change
Characteristics of systems where care is required in applying DSDM

- Process control or real time applications
- Requirements that have to be fully specified before any code can be written
- Safety critical applications
- Systems delivering re-usable components
  - re-use debate: correctness versus high modularity
Inappropriate reasons for DSDM

- Impatience
  - "we want the system now and we don’t care about the rest of the selection criteria".

- Control
  - If traditional controls are applied to DSDM, the project will probably not succeed in delivering quality software to the business when it wants it.
Potential Risks in using DSDM

- Lack of user involvement
- Excessive time spent on decision making
- Irreversible increments are developed
- Team focus on activity rather than delivery of products
- Testing is not integrated throughout the lifecycle
- Users allocated to the project are “not wanted” by the organisation
- Users get too involved in the project
- Data structures get too monolithic and inflexible due to rapid prototyping
Techniques to consider

- Flexibility
- Timeboxing
- MoSCoW Rules
- Prototyping
- Facilitated Workshops
Flexibility

- The flexibility of requirements to be satisfied has significant impact on the development processes and controls, and on acceptance of the system.

- A fundamental assumption of DSDM is that nothing is built perfectly first time.

- Assumes that a usable and useful 80% of the proposed system can be produced in 20% of the time it would take to produce the total system.
80:20 model

\[ \text{Requirements} : \text{Time} = 80\% : 20\% \]
80:20 in more detail

- A fundamental assumption is that nothing is built perfectly first time, but that a usable and useful 80% of the proposed system can be produced in 20% of the time it would take to produce the total solution.
- One of the underlying principles of DSDM is that fitness for business purpose is the essential criterion for the acceptance of deliverables.
- This moves away from the approach of satisfying all the "bells and whistles" in a requirements specification as this approach often loses sight of the fact that the requirements may be inaccurate.
Timeboxing

- This is a very important aspect of DSDM projects. Without effective timeboxing, prototyping teams can lose their focus and run out of control.

- Timeboxing works by concentrating on when a business objective will be met as opposed to the tasks which contribute to its delivery.
Component parts of a timebox

Investigate – a quick pass to see whether the team is taking the right direction
Refinement – to build on the comments resulting from the review at the end of investigation
Consolidation – the final part of the timebox to tie up any loose ends
Timeboxing basics

- time between start and end of an activity
- DSDM uses nested timeboxes, giving a series of fixed deadlines
- ideally 2 - 6 weeks in length
- objective is to have easiest 80% produced in each timebox
- remaining 20% potentially carried forward subsequent timeboxes
- focus on the essentials
- helps in estimating and providing resources
Key Characteristics of Timebox

- Time available dictates work done
- Review at deadline
- Reaffirm scope
- Prevent “drift”
- Potential risk
  - Loss of functionality
  - Failure to meet all objectives
MoSCoW Rules

formalised in DSDM version 3
M must have – fundamental to project success
S should have – important but project does not rely on
C could have – left out without impacting on project
W won't have this time round can be left out this time
Prototyping in DSDM (1)

- Prototypes are necessary in DSDM because
  - facilitated workshops define the high-level requirements and strategy
  - prototypes provide the mechanism through which users can ensure that the detail of the requirements is correct
  - demonstration of a prototype broadens the users' awareness of the possibilities and assists them in giving feedback to the developers
  - speeds up the development process and increases confidence that the right solution will be delivered.
Prototyping in DSDM (2)

- A prototype need not be complete and tested with respect to all its related functional and non-functional requirements.
- DSDM prototypes are intended to be incremental, in other words they evolve.
- Four categories of prototype are recommended:
  - Business for demonstrating the business processes being automated,
  - Usability for investigating aspects of the user interface that do not affect functionality,
  - Performance & Capacity for ensuring that the system will be able to handle full workloads successfully,
  - Capability/Technique for trialling a particular design approach or proving a concept.
Prototype: Potential Issues

- Experience shows prototyping is a potential problem area in DSDM
  - Lack of control
  - Scope creep
  - False expectation of completion
Facilitated Workshops

- Purpose to produce clear outcomes that have been reached by consensus

- Participants
  - workshop sponsor
  - workshop owner
  - facilitator
  - participants
  - scribes
  - observers
  - prototypers
Advantages of Workshops

- Speed
- Involvement / ownership
- Productivity
- Consensus
- Quality of decisions
- Overall perspective / synergy
Types of Workshop

- Business Vision Analysis
- Business Systems Planning
- Technical Systems Options
- Business Process Design
- Information Systems Design
- Business Information Systems Benefits
- Information Systems Requirements Definition/Prioritisation
- Acceptance Test Planning
Linking DSDM to other methods

- Why look at DSDM in isolation?
- When not take the “best” bits of DSDM and combine with other methods?
- Why not use the robustness of more formal methods to strengthen DSDM?
- Why should organisation be constrained by one method?
For example merge UML with DSDM
XP in more detail

Next section of lecture examines the principles and practices of XP.
Addressing Risks in XP

- Schedule slips - short release cycles, release highest priority first
- Project cancelled - customer chooses the smallest release that makes the most business sense
- Systems go sour - XP creates and maintains a comprehensive suite of tests, run and rerun after every change to ensure a quality baseline
- Defect rate - test by function by function (programmer) and program feature by program feature (customer)
- Business misunderstood – customer to be an integral part of the development team
- Business changes - shortens release cycle
- False feature rich - address highest priority tasks
- Staff turnover - give programmers responsibility for estimating and completing their own work
XP Core Values

- Values necessary for an emergent culture and improved productivity
  - Communication
  - Feedback
  - Simplicity
  - Courage

- To support and reinforce the core values, XP recommends a whole range of planning, testing and development practices that can be divided into 3 groups:
  - Programmer practices
  - Team practices
  - Project practices
XP Practices in Projects

- Plan
- Small releases
- Metaphor – e.g., Microsoft use “desktop”
- Simple design
- Testing
- Refactoring
- Pair Programming
- Collective Ownership
- Continuous iteration
- No overtime
- On-site customer
- Coding standards
the XP 'circle of life'

programmer practices

on-site customer
acceptance testing
open workspace
coding standards
release planning

team practices

test-first design
pair programming
simple design
refactoring
sustainable pace
metaphor

project practices

iterations
one team
continuous integration
user stories
small releases
XP challenges assumptions

XP says that analogies between software engineering and other engineering are false:

- software customers’ requirements change more frequently;
- our products can be changed more easily;
- the ratio of design cost:build cost is much higher;
- if we consider coding as “design” and compile-link as “build”:
  - the “build” task is so quick and cheap it should be considered instant and free,
  - almost all software development is “design”.

XP challenges assumptions

- The design meets *known existing requirements*, not all possible future functionality.
- Beck (2000): “If you believe that the future is uncertain, and you believe that you can cheaply change your mind, then putting in functionality on speculation is crazy. Put in what you need when you need it.”
How XP works

- As with RAD and DSDM etc. programmers meet and communicate with customers regularly, and the software gets released incrementally.
- Programmers always work in pairs (considered more productive).
Pair Programming

- At first glance seems expensive and wasteful use of labour
- Two programmers working together on one programme on one machine,
  - First programmer writes code,
  - Second engages in strategic thinking, suggesting better alternatives, correcting mistakes (syntax and semantics), identifying unit tests
- After a time pair swap roles
- Pairing is dynamic
  - ie people in team move between pairs
- Helps in testing and following standards
- At least two people in organisation will understand the code!
How XP works

- Testing is the start point, not the end:
  - For each user *story*, the customer first writes an acceptance test.
  - For each unit the programmer writes a set of unit tests.
  - Then each unit in a story is coded.
  - When a unit is ready, its tests are run automatically.

- Customers are *allowed* to suggest improvements.

- Redesigns are common - what they call *refactoring* - and handled easily.
The Limits of XP

Technical limitations
- Programming language - possibly
- Legacy code
- Where rapid change is not facilitated

Cultural limitations
- Team size – big teams can be problematic
- Colocation – example in distributed projects
- Situations where users / customers are distrustful
- Product development
- Regulated industries
- Competitive tender / fixed price contracts
XP and DSDM

- DSDM and XP aim to solve the same problem: delivering good systems in short timescales
- Argue that they are complementary – activity for you to think about in seminar
- XP focuses on the act of programming which is treated very lightly in DSDM
- DSDM provides a controlling framework into which XP can be plugged
XP and DSDM

XP focuses on link between FMI and D and BI
DSDM and XP are not homes for hackers
Opportunity for practitioner certification
Developers work in teams whose focus is not only on technological problems
Practitioners are expected to be quality conscious and manage their work effectively
Quality Issues!

- Agile methods aim to remove the “quick and dirty” image of RAD
- Agile methods address maintainability
  - Options
    - maintainable from day 1
    - maintainable at a late date
    - quick fix which will be withdrawn later
- Agile methods develop solution that is fit for business purpose
- Testing happens throughout development
- DSDM linked to TickIT by the BSI
- Quality expectations of right “first time every time” need to change
Measuring success of Agile methods

- Reduce the number of systems developments which fail
- Increase user satisfaction
- Improve productivity of developers
- Get business solutions to live environment in time
Conclusion

- DSDM and XP can potentially be great benefits to systems development in business
- Great care should be taken in selecting projects to make use of DSDM (suitability matrix)
- DSDM and XP are neither cheap nor easy options
- DSDM and XP both require combination of technical and interpersonal skills – in both approaches people are key