Lectures 9 and 10

Alternative Development Strategies

Lecture Objectives

- At the completion of this topic, you should
  - be aware of some alternative approaches to information systems development
  - be aware of the usefulness and limitations of some of these alternatives

Systems Development Strategies

- Traditional SDLC
- Prototyping
- Joint Application Development (JAD)
- Rapid Application Development (RAD)
- Application packages
- Enhancing existing systems

Systems Development Concepts

- Method
  - a prescribed set of tasks that uses specific techniques and tools to complete a systems development activity
- Technique
  - a way of doing a particular task in the systems development process
- Tool
  - automated tools to help systems development

Systems Development Concepts

- Methodology
  - a collection of procedures, techniques, tools and documentation aids which assist systems developers to implement information systems
  - consists of phases which consist of sub-phases
  - helps developers plan, manage, control and evaluate information systems projects

Avison and Fitzgerald (1995)

Traditional SDLC

- Formalised method for building information systems (the oldest one - early 1970s)
- The "waterfall" model:
  - feasibility study
  - system investigation
  - systems analysis
  - systems design
  - implementation
  - review and maintenance
Traditional SDLC

- Has a number of phases, each consisting of a number of sub-phases, activities
- Many variants
- System is generally developed sequentially, but some tasks in earlier phases may be revisited, and some tasks may be done in parallel
- Formal division of labour between users and IS staff and amongst IS staff
- Formal sign-offs required at the completion of each major stage

Traditional SDLC

- Useful for
  - providing a base guideline for systems development which can be modified to suit specific requirements
  - building large transaction processing systems (TPS) and management information systems (MIS) where requirements are highly structured and well defined
  - building complex systems which need rigorous and formal requirements analysis, predefined specs, and tight controls over the systems building process

Traditional SDLC

- Limitations
  - resource intensive
    - takes time to gather information and prepare detailed specifications and sign-off documents
    - could take years to develop a system - requirements may change before the system is operational
  - inflexible and inhibits change
    - time and cost required to repeat activities encourages freezing of specifications early in development .. locks users into something that may no longer be appropriate

Traditional SDLC

- Limitations
  - hard to visualise final system
    - users sign off specification documents without fully understanding their contents or implications
  - ill-suited to decision-oriented applications
    - decision-making is often unstructured .. there are no well-defined models or procedures
    - being forced to develop formal specifications can be very inhibiting

Traditional SDLC

- Limitations
  - not well suited to the small desktop systems and web-based applications that will predominate in the future
  - not well suited to short development life cycles
  - does not encourage user participation
  - management and strategic needs ignored
  - focus on technical aspects

Prototyping

- Prototype
  - a working model of some aspect(s) of an information system

Prototyping

- Prototyping
  - an iterative process of quickly building an experimental system, for demonstration and evaluation so that users can dynamically determine their information requirements and explore and test the design of the
Prototyping

- Can be used in various phases of the SDLC
  - Initiation - to test the feasibility of a particular technology that might be applied for an IS
  - Analysis - to discover users’ requirements by ‘painting’ screens and reports to solicit feedback
  - Design - to simulate the ‘look and feel’ of the system and evaluate how easy it is to use and learn
  - Implementation - prototype evolves directly into the production system, to train users

A prototype is designed with an expectation of change - expect to get it wrong the first time!
- Need appropriate technology
- Types of prototypes
  - features eg external design mock-up
  - throw-away
  - evolutionary

Prototyping

- Useful
  - when there is uncertainty about requirements or design solutions
    - can capture requirements in concrete, rather than verbal or abstract form
    - users are more likely to be able to state their detailed requirements when they see and use a prototype
    - users are more likely to get what they want

- Limitations
  - tends to skip through analysis and design phases too quickly --> lack of thorough understanding of the problems
    - a tendency to avoid creating formal documentation of system requirements which can then make the system more difficult to develop into a production system
    - can discourage consideration of a wide range on alternative design options ... tendency to go with the first one that the user likes

- Limitations
  - often lacks flexibility, technical efficiency and maintainability because of hasty construction
  - not suitable for large applications which have large amounts of data and multiple users - hard to control
  - often built as stand-alone systems, thus ignoring issues of data sharing and interactions with other existing systems
Prototyping

• **Limitations**
  – checks in the SDLC are bypassed so tendency to gloss over essential tasks eg. feasibility, standardisation, documentation, testing, security, etc..
  – can become too specific to the user representative and difficult to adapt to other potential users

Joint Application Development (JAD)

• **Is actually analysis and design**
  – Originated in late 1970s at IBM
• **Brings together key users, managers, systems analysts in a group interview with a specific structure of roles and agenda**
• **Purpose**
  – collect key system requirements
  – develop system design

Joint Application Development (JAD)

• **Group meeting**
  – formal agenda
  – avoid distractions
  – identify areas of agreement and conflict
  – resolve conflicts during the period of sessions
  – focus on rapid delivery of analysis and design specifications

Joint Application Development (JAD)

• **JAD participants**
  – facilitator - organises and runs the sessions
  – scribe(s) - takes notes on PC, CASE tool etc
  – users - understand the system requirements
  – managers - organisational overview
  – systems analysts - technical knowledge, learn about the system
  – sponsor - senior executive who commits and funds the process

Joint Application Development (JAD)

• **Preparing for JAD sessions**
  – JAD leader prepares and distributes agenda and documentation about scope and objectives
  – Agenda specifies issues to be discussed and time allocated to each
  – Ground rules for running the sessions are made clear
  – Ensure users who attend are knowledgeable about their business area

Joint Application Development (JAD)

• **JAD sessions**
  – from one to five days
  – structured meeting room (war room) with white boards, CASE tools etc
  – located away from users' workplace
  – outcome is documents detailing the system - workings of/requirements for the system, system design specifications, prototypes
Joint Application Development (JAD)

• Conducting JAD sessions:
  – Avoid deviating from the agenda
  – Keep to schedule (time for topics)
  – Ensure scribe takes adequate notes
    > use formal minutes
  – Avoid using technical jargon
    > involve all participants
  – Use conflict resolution strategies

Joint Application Development (JAD)

• Conducting JAD sessions:
  – Allow ample breaks
    > keep everyone at peak efficiency
  – Encourage group consensus
  – Encourage participation vs individuals dominating
  – Ensure ground rules are adhered to

Joint Application Development (JAD)

• Benefits
  – Reduced time to move requirements/design forward (group vs one-on-one, details worked on between meetings)
  – Key people work together to make important decisions
  – Commitment is focused and intensive, not dissipated over time
  – Conflicts and differences can be understood and resolved

Rapid Application Development (RAD)

• Rapid Application Development (RAD)
  – A systems development methodology created to radically decrease the time needed to design and implement information systems
  – James Martin (1991) - RAD methodology

Rapid Application Development (RAD)

• RAD claims to offer
  – A development lifecycle for much faster systems development
  – Better and cheaper systems
  – More rapid deployment of systems as developers and users work together in real time

Rapid Application Development (RAD)

• RAD relies on
  – Extensive user involvement
  – JAD sessions
  – Prototyping
  – I-CASE tools (integrated CASE tools)
  – Code generators
Rapid Application Development (RAD)

- **Evolution of RAD**
  - Pressures for businesses to speed up and compete in a changing, global environment
  - Shorter development lifecycles
  - Dissatisfaction with IT department
  - Diffusion of high-powered prototyping and CASE tools

  Why wait 2 or 3 years to develop systems likely to be obsolete upon completion?

- **James Martin’s four pillars of RAD**
  > Tools
  > People
  > Methodology
  > Management

- **Tools**
  - I-CASE tools with prototyping and code generation facilities
  - Visual development environments

- **People**
  - Manager and user participation in JAD type workshops
  - Developer roles - workshop leader, project leader, scribe, repository manager, construction or SWAT (Skilled With Advanced Tools) team

- **Methodology**
  - to guide and control the use of RAD techniques
  - Should be automated for ease of use - adaptability and flexibility

- **Management**
  - Executive sponsor
  - Facilities and support for the RAD team

- **RAD lifecycle**
  - Is evolutionary
  - Uses “timeboxing”
  - Avoids “feature creep”
  - Avoids requirements “gold plating”
Martin’s (1991) RAD lifecycle

- Requirements planning phase
  - managers, executives, key users determine requirements in terms of business areas and business problems
  - JRP workshops to agree requirements, overall planning

- User design phase
  - end users and IS personnel use I-CASE for rapid prototyping of system design
  - JAD sessions to develop basis for physical design
  - users sign off on CASE-based design (no paper-based spec)

- Construction phase
  - IS personnel now generate code using I-CASE tool
  - end users validate screens, design, etc.

- Cutover phase
  - delivery of new system to users: testing, training, implementation
  - can be combined with construction in small systems

- Timeboxing vs traditional approach
  - Traditional approach - every possible requirement is implemented together leading to increased complexity and long delays
  - Martin claims RAD can produce a system in 6 months that would take 24 months using traditional development methods
  - Small development teams are essential for RAD to work

- Uses timebox approach:
  - system to be developed divided into components that can be developed separately
  - the easiest and most important 75% of the system functionality produced in first timebox (90 day cycle)
  - forces users to focus on the necessary and most well-defined aspects
  - users experience this component first and other component requirements may then change
  - functionality is trimmed: “gold plating” is avoided
  - avoids “feature creep” - more and more requirements creep in during development than originally specified

- Advantages
  - quick development
  - cost savings
  - higher quality/improved performance as easier and most important functions targeted first
  - avoids feature creep
  - aligned with business changes
Rapid Application Development (RAD)

- Disadvantages
  > detailed business models/understanding neglected --> inconsistencies, misunderstandings
  > programming standards, scalability, system administration issues neglected e.g. database maintenance, database reorganisation, backup/recovery, distribution of system updates, etc.

Application Packages

- Purchasing or leasing set of pre-written application software programs that are commercially available

- May range from simple PC systems to complex mainframe or client-server systems

Application Packages

- Useful
  - when you need an information system for a common company function eg. payroll
  - when information systems resources for in-house development are in short supply
  - when the application software package is more cost effective than in-house development
  - because the most of the design and implementation tasks are done - significant time saving
  - because the system and documentation are usually maintained by the vendor

Application Packages

- Useful
  - because the design spec is fixed - no endless reworking - users have to accept it politically because:
    > external work is often perceived as being superior to an in-house effort - easier to get new systems into the company
    > easier to get management support because of fixed costs
    > problems can be attributed to the package rather than internal sources - ends endless source of internal conflict

Application Packages

- Limitations
  - very rare to find a package that can do everything well that a user wants
  - often need to develop specialised package additions because multi-purpose packages do not handle certain functions well
  - conversion and integration costs can sometimes be so significant as to render the project infeasible

Application Packages

- Limitations
  - some vendors refuse to support packages which have been customised by the users - and most packages require some customisation
  - customisation can be so extensive that it would have been cheaper to develop the system in-house
Enhancing Existing Systems

- Can use any development approach - most organisations have a maintenance - development cycle
- The main issues are:
  - urgency
  - integration
  - updating documentation
- Tendency to jump in and code, with little thought to surrounding development tasks

Alternative Development Approaches

- So... as developers we must realise
  - There are many different ways of developing information systems
  - The difficulty is finding the right blend of approaches and techniques to suit the organisation's business, social and political systems development environment
  - There generally is a methodology that is 'more appropriate'

References

