An introduction to Rapid Application Development

(Agile concepts)

RAD – Where it all started

In this lecture we will introduce Rapid Application Development as an approach to systems development.

Over the next few weeks we will look at methods/frameworks that support this approach.

What is RAD (1)?

- It is a systems development approach suggested by James Martin in 1990
  - He published a *seminal text*:
    - *Martin J. (1990), Rapid Application Devt, Macmillian (USA).*
  - He developed the concept of RAD at IBM in the 1980s before publishing in 1990.
  - His book discussed the *concept* of Rapid Application Development
    - He did not propose a methodology – this came later from other sources

What is RAD (2)?

- It is a systems development concept that products can be developed *faster* and to a *higher quality standard* through:
  - Harnessing the knowledge of the user throughout the development life cycle
    - The user is an integral part of the development team

What is RAD (3)?

- Agreeing *high level requirements* at the beginning of the project **but:**
  - Using workshops or focus groups to gather them initially
  - Refining them during the development cycle with the user
- Using the concept of *prototyping* to work with users and develop/revise low level requirements

What is RAD (4)?

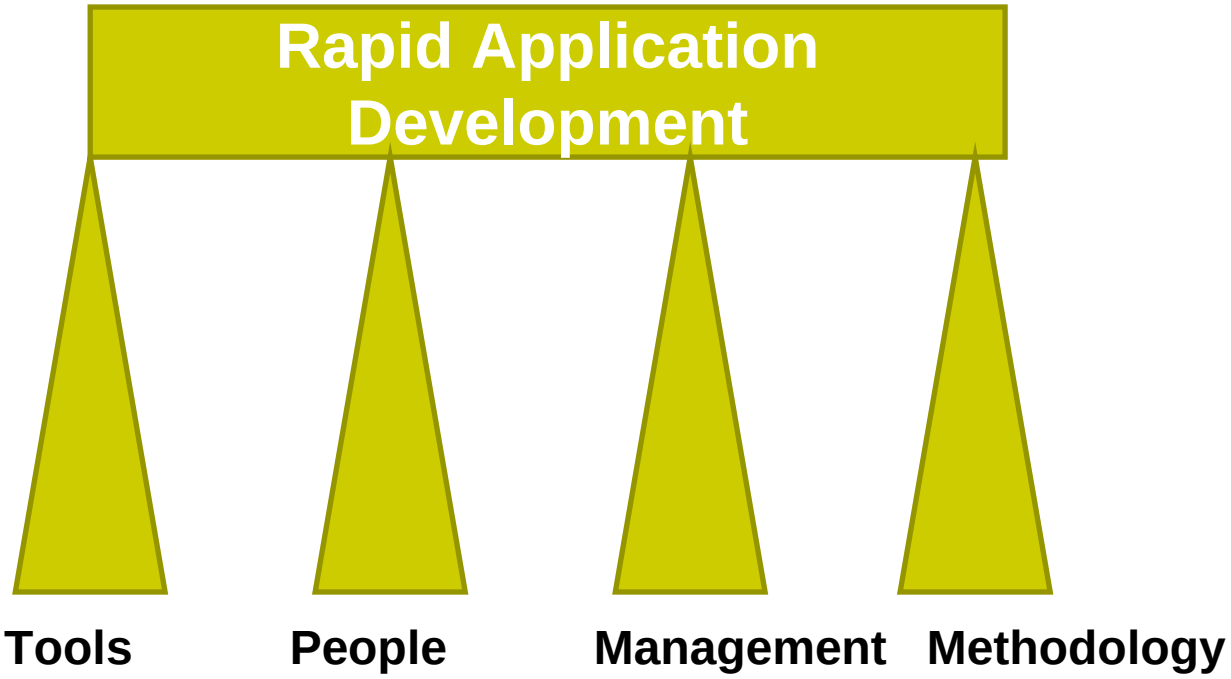
- Using a *rigidly paced, tightly defined* schedule to ensure that something is always delivered by any given deadline.
  - This is called *timeboxing.*
- Using software development tools such as visual building environments, reuse of s/w components to *reduce* the time spent on coding.

What is RAD (5)

- Taking an:
  - **Iterative approach** and an
  - **Incremental approach** to achieve all of this

- *Note that all of this things are concepts suggested by the Agile approach much later – there are no new concepts in the Agile approach they are based on RAD*

The four essential ingredients of RAD (From James Martin (1990))



The four essential ingredients of RAD

- James Martin describes four essential aspects of RAD:

- **RAD and Tools**
  - RAD life cycles depend on automated tools
  - Development process should have maximum degree of automation

RAD and Tools

- Tools required include CASE tools, prototyping tools and integrated environments which can be used by whole life cycle.
- Note that many of the tools on the market have, until recently, been **inadequate** for fast development life cycles

RAD and People

- RAD is highly dependent upon people
- People with specialist RAD skills are required to be part of RAD team.
- These people include:
  - Users
  - Management representatives
  - System developers
  - Project management staff

RAD and Management

- Management of a RAD project is crucial to its success.
- Tempting to ignore management in a RAD environment:
  - Isn’t prototyping all about an ‘evolving’ environment without constraints such as time scales, predefined requirements etc…?

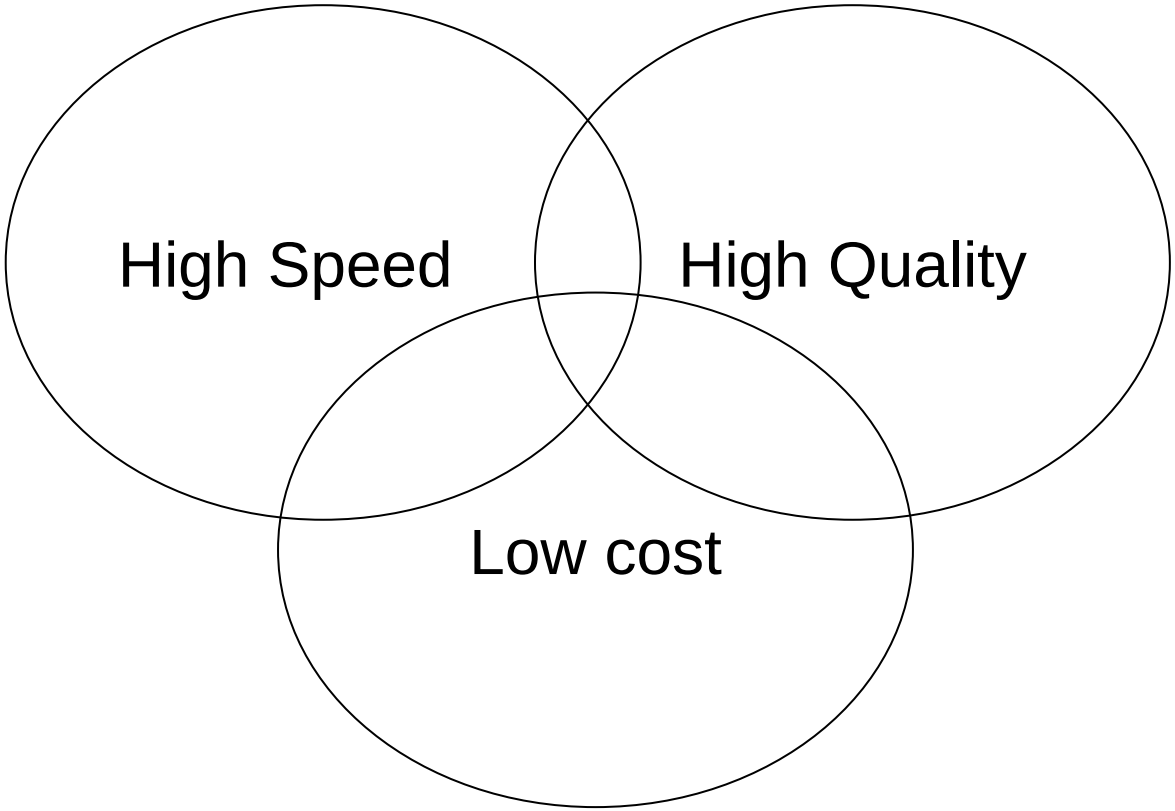
RAD and Methodology

- To achieve good results, an I.S organisation needs a methodology that suits both its **environment** and its **tool set.**
  - When James Martin first published his thoughts, there was no method to support RAD.
  - DSDM developed in 1994 and has been refined many times since then.

RAD and Quality

- Traditional view is that if you develop something *quickly* you sacrifice some of the *quality.*
- *Quality* is key to the success of any IT development project (as we have already discussed)
- *James Martin* suggested that:
  - *High quality, lower cost and rapid development* go hand-in-hand if an appropriate development methodology is used.

RAD and Quality



Why RAD is increasingly important to industry (1)

- **Response to users**
  - History of IS development cluttered with failure of IS projects that *did not do what the user wanted them to do.*
  - Use participation is now recognised as an important part in problem definition and functionality design.
    - A key part of the RAD philosophy is user participation throughout the development cycle.

Why RAD is increasingly important (2)

- Response to users (cont..)
  - ‘empowering the client’ is fundamental principle of RAD
  - prototyping is built upon user-centred activities & enables feedback from user at all stages of development.
  - Development activities such as JAD, reviews, walk throughs etc. all include user in complete development cycle

Why RAD is increasingly important (3)

- **The modern business climate**
  - All organisations operate within an extremely competitive environment where business processes are changing **rapidly** to maintain competitive edge
  - Development projects with long life cycles are unacceptable
  - Also, unacceptable to deliver systems that don’t meet needs of the user.

What projects are suitable for RAD environment? (1)

- RAD is not suitable for all types of projects (DSDM.org)
- There are several factors seen as useful for deciding on the suitability of a given IS project to RAD development.
- Type of system to which RAD is especially seen as suitable are the following applications…….

What projects are suitable for RAD environment? (2)

- The application is interactive
- the functionality is clearly visible at the user interface
- the user group is clearly defined
- the functionality of the system is not computationally complex

What projects are suitable for RAD environment? (3)

- The type of organisation is also important. RAD is most suitable where the following factors are taken into account:
  - **Management Involvement**.

    RAD needs the commitment of senior user management to provide significant end-user involvement. Without such a commitment more traditional waterfall methods should be considered.

What projects are suitable for RAD environment? (4)

- **End-user involvement**.
  - Easy access by developers to end-users is imperative. The developers and users should reside in the same location.
- **The development team**.
  - The development team should be committed and trained in RAD.

What projects are suitable for RAD environment? (5)

- **Project management**.
  - One of the problems with prototyping is that it can run out of control. Good project control must be in place
- **Empowering the team**
  - The RAD team must be given authority to make design decisions on a day-to-day basis without the need for consultation with their superiors.

What projects are suitable for RAD environment? (6)

- **The size of the project**
  - RAD is good for small to medium scale projects.
  - Large projects should only be considered for a RAD approach if the project is capable of being split up into a number of smaller projects.
  - It should be possible to deliver each smaller project independently.

What projects are suitable for RAD environment? (7)

- **Team size.**
  - The RAD development team must be reasonably small in order to
    - reduce management concerns
    - maintain clear lines of communication
    - increasing commitment to development and ownership of the project.

When not to use RAD

- RAD should not be used for:
  - real-time or safety-critical systems
  - very large organisational-wide systems
  - computationally complex systems
  - any application where the functional requirements must be fully understood before any programs are written

Managing Requirements in an Agile Environment (1)

- In an Agile project management environment, while high-level requirements are also captured upfront, it is understood that requirements may evolve over the course of the effort.

Managing Requirements in an Agile Environment (2)

- The effort begins with a high-level scope agreement where initial business requirements are translated into user stories. Those user stories specify the needs of the product based on the information at the time given.

Managing Requirements in an Agile Environment (3)

- The goal is no longer focused on eliciting the “as-is” in order to the define the “to-be,” but to clarify and ensure understanding of the business need for all users.

Managing Requirements in an Agile Environment (4)

- As more information becomes known over time, the team is better able to adjust and make changes accordingly.