CIS1108

Rapid Application Development (RAD)

In this part we look at the agile approach taken to software development. Agile, or rapid, development techniques are an attempt to maintain software quality while making development faster to increase competitiveness in today’s software development market.

Agile Development Principles

1. Priority given to individuals and interaction, rather than to processes and tools (trust, personal communication, quality of work, responsibility, self-management at micro-level)
2. Attention to working parts of solution, rather than to comprehensive documentation (tangible results, client reassurance, evident solution parts)
3. Importance to customer involvement, rather than contractual detachment (stakeholder participation, transparency, collaboration, feedback, client acceptance)
4. Reactive to change, rather than railed into a rigid plan (flexibility, relevance, result effectiveness, client satisfaction)

General points only taken from www.agilealliance.org

RAD Principles

1. All stakeholders
   - must take active part in decision taking
   - can take binding decisions
2. Deliverables
   - are frequent
   - must clearly map on to the business process
   - are bite-sized
3. All development effort is reversible
4. Requirements are agreed upon at a high level
   - the method of their actual implementation is not that important
5. Testing permeates the SDLC
6. Collaboration, not competition between stakeholders is of essence

Main RAD Features

- User involvement
- Strict (and usually fast) delivery times
- Prototyping
- Effective use of modern development tools
- Reduction of the requirements capture and analysis effort
- Incremental development

The next slides will go through each feature

User Involvement

- JAD
- Decision taking
- Sign-offs
- Prototype demos
- Bite size development focus
- Problem detection and correction

Strict Delivery Times

- Timebox approach
  - 2 to 6 weeks
- Prioritisation (MoSCoW rules)
  - Must, Should, Could, Want to have
- User agreement
  - Power of decision

Prototyping

- The main communication tool in RAD
- Requires planning
- Serve as milestones
- Should be measureable
- Can take any form horizontal or vertical
- Is generally evolutionary in nature

Effective use of Modern Techniques

The “educated” use of:

- Models
- Methods
- Techniques
- Technologies
- Environments

Reduction of Requirements Capture and Analysis Effort

- Less time on getting every part right as early as possible at model level, producing a “ready” solution.
- More time on doing smaller bits, asking for feedback, combining the bits to “grow” the solution.
- Shift of some analysis effort from developer to user.

Incremental Development

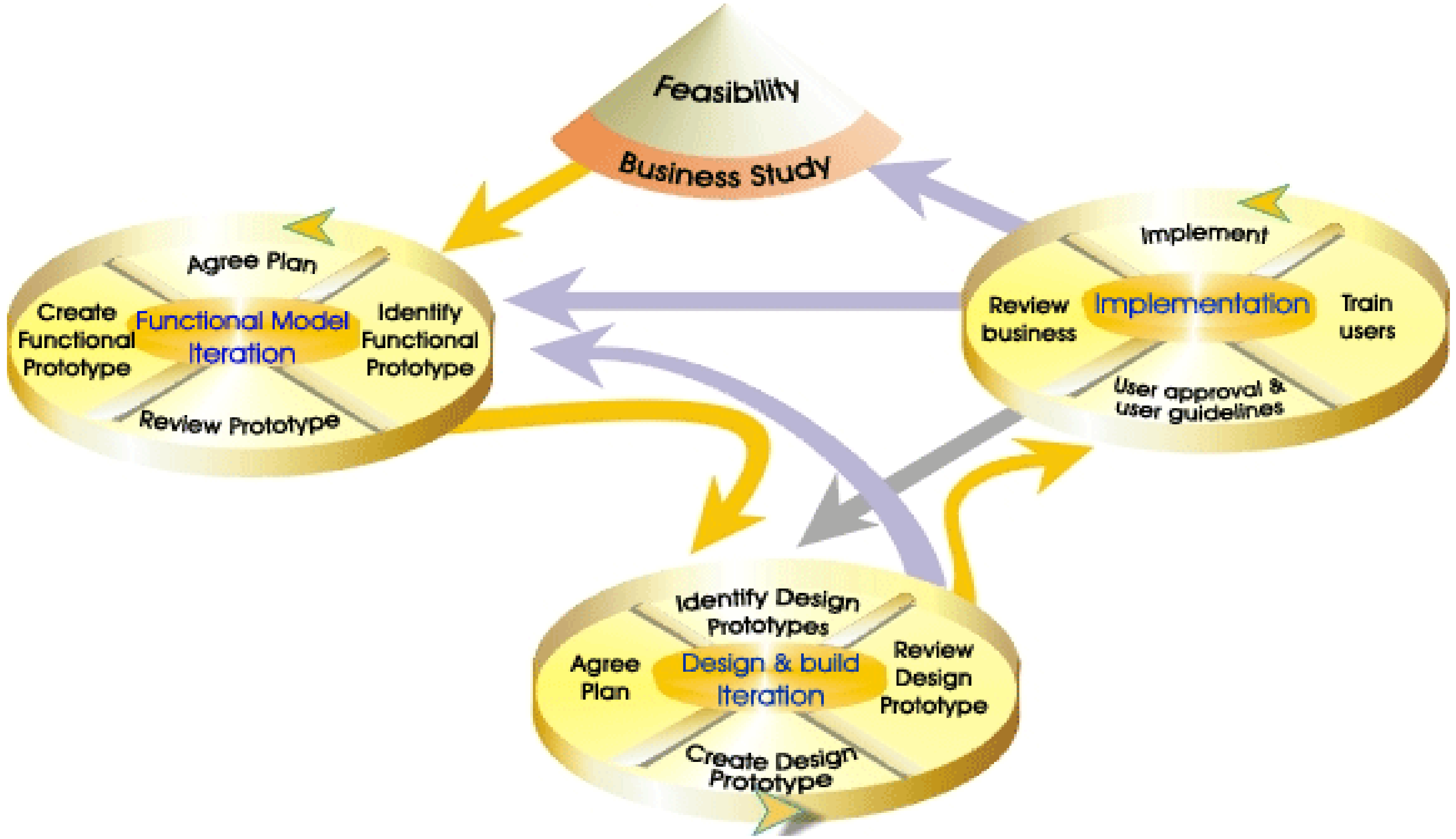
- The current version of evolves from a simpler previous version
- Development can be subdivided into manageable parts
- Requires good configuration management to work
- Small development chunks are a must
- Frequent deliverables are a must

DSDM- An SDLC based on RAD

- Dynamic Systems Development Method (DSDM)
- Is a standard used throughout the UK and a number of other countries. Yet others, use it sparingly
- Is an "open" method – i.e. non-commercialised
- Is supported by a consortium also called the "DSDM Consortium"

see site: dsdm.org (registration might be required)

The DSDM Framework



DSDM

- Does not specify a particular method of development to use in its phases
- Just offers a framework upon which development may proceed and at which points and what sort of milestone to produce
- Really only covers analysis, design and implementation phases

RAD Principles (9 in all) *(1/2)*

1. Win-Win (my addition)
2. All stakeholders must take active part
3. All stakeholders can take binding decisions
4. Deliverables are frequent
5. Deliverables must clearly map on to the business process
6. Deliverables are bite-sized

RAD Principles (9 in all) *(2/2)*

7. All development effort is reversible
8. Requirements are agreed upon at a high level – i.e. the method of their actual implementation is not that important
9. Testing permeates the SDLC
10. Collaboration, not competition between stakeholders is of essence

The DSDM Filter

When to use DSDM? Answer these…

- Is the functionality visible through the UIs?
- Can all end user classes be identified?
- Is the system heavy on computation?
- Is the system “large” and can it be sensibly split up?
- How time-constrained is the project?
- Are requirements abstract enough and can they change within limits?

Systems that can benefit from DSDM

- Interactive
- Based on UIs (functionally)
- User groups are clearly defined
- Not too complex
- Can be incrementally developed
- Time-wise guaranteed
- Requirements can be prioritised
- Requirements can undergo changes

Examples of Agile Methodologies

- XP (Taken to be generically representative of all the “agile” methodologies)
  This method was outlined in the previous slides
- Scrum (first created in 1994, commercially used in 2002/3)
  This method is based on iterative (usually 30-day) segments called “sprints”. Autonomous teams work in parallel on pre-prioritised requirements and synchronise through short daily meetings called “Scrums”.
- Crystal (2002)
  This method is developer-centric. It assumes that the main project quality driver is the quality of people. Also relies on frequent communication and less intermediate deliverables.
- Adaptive Software Development (2000)
  This is a feature-oriented method that is driven by the development of features as understood by the client. Change is expected and risk is tackled “head-on” by tackling the most complex issues first.

Summary

- The concepts behind agile development methods
- Rapid Application Development (RAD)
- Examples of some agile methodologies