LECTURE 2 - INFORMATION SYSTEM DEVELOPMENT

Learning outcomes

By the end of this topic, the student should be able to;

1. Outline the problems encountered in information system development both from a user perspective, client perspective and developer perspective.
2. Examine why the above problems are encountered, and how they can be avoided.
3. Discuss the different project life cycles in system development.
4. Carry out extensive case studies of real-life systems in their failure or success in development.

2.0.1. Problems in Information Systems Development

Most software developers go straight into coding without first understanding the issues surrounding the problem or consulting the right people who are involved in the problem. Analysis is a way of understanding what needs to be done before we begin trying to do it, and design is a way of checking that the planned action or solution really meets the needs of the situation. An understanding of potential problems is an essential precursor to systems analysis and design.

The Problems

1. End user perspective:

(a) **What system?** I haven't seen a new system: Vaporware describes a software product that is much talked about, but never released to its intended users. An astounding proportion of information systems development projects fail to deliver any product at the end.

(b) It might work but its dreadful to use! This relates to systems that are unpleasant or difficult to use. Systems might fail to meet the criterion of usability in a number of ways, including: poor interface design, inappropriate or illogical sequence of data entry, incomprehensible error messages, unhelpful „help", poor response time and unreliability in operation. The table below illustrates examples:

Systems Characteristic	Example
Poor interface design	A web page with yellow text on a white background
Inappropriate data entry	A system where the backspace key sometimes deletes whole words
Incomprehensible error message	A system message that says 'error # 13452'
Unhelpful 'help'	A system message that says 'wrong date format –try again'
Poor response time	
Unreliability in operation	

Figure 2.1 Information Systems Development Problems

2. A client's perspective

(a) If I had known the real price, I‟ never have agreed‟: it is almost routine in many organizations for information systems to exceed their budget. Any project that overruns its budget can reach a point where the total costs outweigh the benefits that will be provided on completion.

(b) It's no use delivering it now‟: A project that is completed late may no longer be of any use. Many other kinds of projects are time-critical.

(c) A messy installation: Once a new system gets a bad press it can be very difficult to overcome the resistance of the people who are expected to use it, particularly if there is an existing alternative available to them.

(d) I didn't want it in the first place‟: Organizations are complex and political by nature. There are sometimes conflicting ideals and ambition and the play of power within an organization.

(e) Changing needs: Sometimes by the time the system is completed. The requirements are no longer the same as they were thought to be at the beginning

Requirements can change for many reasons:

• Project timescales can be very long, and the business needs may change in the meantime.

• Users naturally tend to ask for more, as they learn more about what is available.

• External events can have a dramatic impact – for example a currency crisis that led to the collapse of a significant market might render a new information system pointless.

3. A developer's perspective

(a) We built what they said they wanted: Changes to the requirements can be difficult for a developer.

(b) There wasn't enough time to do it any better: In every project, there are pressures from outside that limit the ability of the development team to achieve excellence.

(c) Don't blame me – I have never done object-oriented analysis before: In a successful information system development team, the members must possess a harmonious blend of skills that are appropriate to the needs of the project.

(d) How can I fix this, I don't know how it's supposed to work: This complaint is often heard from programmers who have been asked to modify an existing program, and who have then discovered that there is no explanation of what it does or how it works.

(e) We said it was impossible, but no one listened: Just like client managers, system developer can sometimes be overwhelmed by organization politics. At times this means that the project is forced on an unwilling team who don't believe that it is technically feasible to achieve the project's goals.

(f) The system's fine, the users are the problem: A few information systems professionals, usually those who understand least about business and organizations are sometimes prone to blame the user for everything.

2.0.2. Why Things Go Wrong

Quality Problems

The quality of a product is in terms of its „fitness for purpose“. It is therefore necessary to know for what purpose the system is intended and how to measure its fitness. Both parts of this can be problematic at times.

- The wrong problem A primary cause of system failures in general is that some projects are started with no clear idea about exactly what are the nature and goals of the client organization.
- Neglect of the content This can take the form of a system that is too difficult to use, since the designers have taken insufficient account of the environment in which its users work, or the way that they like to work.
- Incorrect requirements analysis Even if the aims are clear at the onset, many pitfalls arise particularly if the development team doesn‟t not have the right skills, the right resources or enough time to do a good job. However, even if none of these present a difficulty, the project can still fail if the techniques being applied by the team are inappropriate to the project.
- Project carried out for the wrong reason

Productivity Problems

Productivity relates to the rate of progress of the project, and the resources that it consumes along the way.

- Requirement’s drift This means that the users‟ requests change over time.
- External events Depending on the environment in which the organization operates, decisive external events can be impossible to anticipate. For example, a project to build a distributed information system that is to operate on new state-of-the-art computers communicating over public telephone circuits may be sensitive to external factors such as the reliability of the telephone network and call pricing.
- Poor project management d. Implementation not feasible Some projects are overambitious is their technical aims, and this may not become evident until an attempt is made to implement the system.

2.0.3. Avoiding The Problems

There are two major areas of focus if we intend to produce information systems within budget, on time and providing the required functionality: Project Life Cycles and Management of Information systems.

2.0.4. Project Life Cycles

Subdividing the process of software development produces what is known as a life cycle model.

- **Traditional Life Cycle**

This model is also known as the waterfall life cycle model (You should have learnt this in system analysis and design, software engineering – please research further). The traditional life cycle has been used for years but is subject to several criticisms:

- ✓ Real project rarely follows such a simple sequential life cycle. Project phases overlap and activities may have to be repeated.
- ✓ Iterations are almost inevitable, because inadequacies in the requirements analysis may become more evident during design, construction and testing.
- ✓ A great deal of time may elapse between the initial systems engineering and the final installation.
- ✓ The model tends to be unresponsive to changes in client requirements or technology during the project.

- **Prototyping**

A prototype is a partially complete system that is built quickly to explore some aspect of the system requirements that is not intended as the final working system.

It is differentiated from the final production system by some initial incompleteness and perhaps a less resilient construction. A prototype may be constructed with various objectives in mind:

- ✓ Used to investigate user requirements.
- ✓ To investigate the most suitable form of interface.
- ✓ To determine whether a particular implementation platform can support certain processing requirements.

- ✓ To determine the efficacy of a particular language, a database management system or a communications infrastructure.

Prototyping has the following advantages:

1. Early demonstration of a system functionality helps in identifying any misunderstandings between the developer and the client.
2. Client requirements that have been missed are identified.
3. Difficulties in the interface can be identified.
4. The feasibility and usefulness of the system can be tested, even though, by its very nature, a prototype is incomplete.

Prototypes disadvantages:

1. The client may perceive the prototype as part of the final system, may not understand the effort that will be required to produce a working production system and expect it delivery soon.
2. The prototype may divert the attention from functional to solely interface issues.
3. Prototyping required significant user involvement.
4. Managing the prototyping life cycle required careful decision making.

- **Incremental Development**

One way to overcome the problems of the traditional model is to view successful large systems as starting out as successful small systems that grow incrementally.

- **Unified Software Development Process**

This process reflects the current emphasis on iterative and incremental life cycles.

This incorporated the UML and comprises four phases:

1. **Inception**: Determining the scope and purpose of the project.
2. **Elaboration**: Requirements capture and determining the structure of the system.
3. **Construction:** to build the software system.
4. **Transition:** product installation and roll out.

2.0.5. Managing Information Systems Development

1. User involvement A key factor in maximizing the chances of success is ensuring that there is continued and effective user involvement throughout the project.
2. Methodological approaches One of the major influences on the quality of systems developed is the software development approach adopted. If the approach used is not appropriate for a particular type of application, then it may limit the quality of the system being produced.
3. CASE (Computer Aided Software Engineering Tools). These provide support for many of the tasks the software developer must perform.

CASE STUDY: The London Ambulance Service Computer Aided Dispatch System (LASCAD)

LASCAD is being examined because it is an example of a requirements problem. In 1992, the London Ambulance Service's (LAS) entire system was managed from a central location at the LAS headquarters in Waterloo. On average between 2000 and 2500 calls were received daily, 60% of which were requests for emergency services. The LAS devoted 55% of its staff and 76% of its budget specifically to emergency response. The original design of the LASCAD system was first proposed in 1987. It was modified in 1989 and then, due to gross overspending by about 300%, the project was cancelled in 1990. However, the national mandate to reduce emergency response time pushed the LAS to look into a computerized system once again (Dalcher, 1999). According to Wikipedia , LAS remains the „largest ambulance service in the world that does not directly charge its patients for its services". Before 1992, The LAS emergency system was run completely manually and consisted mainly of the following issues and tasks:

1. The call taker receiving large number of calls and then records all the status details on a paper form. He then locates the incident on a map and puts the form on a conveyor belt.
2. Control staffs select the next job from the forms on the conveyor belt then identify a regional allocator.
3. A resource allocator analyzes the locations and status of the ambulances and allocates the call to an available unit and records the assignment on the form.
4. A dispatcher informs the ambulance to which the call was allocated and provided the details of the call.

Proposed Improved System

The government-imposed stipulations that calls be responded to within three minutes, in addition to the obvious deficiencies of the manual dispatch system, this led those in charge of LAS seek out a computer-based alternative. LAS management decided to create a new one and proceeded to gather requirements without receiving input from ambulance crews or dispatchers (Dalcher, 1999).

Rather than simply assisting dispatchers, this completely computerized system would do nearly everything automatically. The simple sequence of steps performed by a person would be to answer the phone, enter incident data into a computer terminal, and then respond if the system displayed exception messages resulting from no ambulances being available for longer than 11 minutes. The software would map the location of calls. The system would then use this map data and the location status details provided by an automatic vehicle locating system (AVLS) to find and dispatch the available ambulance closest to the incident‟s location. This system would be implemented not incrementally, but all at once (Dalcher, 1999).

The Problems

As part of the attempt to save money, the LAS decided to reuse some of the hardware that had already been purchases when working on the failed project instead of purchasing hardware that was either more up-to-date or more suitable for the new system. All companies that submitted bids for the project, greater than £1.5 million were immediately thrown out. This was an unreasonably low price, especially considering the fact that even after £7.5 million had been poured into the previous attempt at a CAD system, the project had failed (Dalcher, 1999). A further hard constraint was placed that the project be completed in 11 months. Any bids not meeting this constraint were not considered.

The problems faced included:

1. The AVLS was unable to keep track of the ambulances and their statuses in the system. It began sending multiple units to some locations and no units to other locations.
2. The system began generating a great quantity of exception messages on the dispatchers‟ terminals such that calls got lost.
3. As more and more incidents were entered into the system, it became increasingly clogged.

4. The software system did not function well when given invalid or incomplete data regarding the positions and statuses of ambulances.
5. A memory leak in a small portion of the code caused memory that held incident information on the file server, to be retained even after it was no longer needed. Eventually when the memory filled up, it caused the system to fail.

The LAS team – not a software development team - went through the process of gathering requirements and specification, without consulting ambulance operators, dispatchers and other key users of the system. The requirements document was highly detailed and extremely prescriptive", very much a direct contradiction of the requirements process" focus on the „what" as opposed to the „how" (Dalche, 1999). The software development process was flawed. No quality assurance was performed, configuration management was absent, agreed-upon changes were not tracked and test plans were not written. The LAS also failed because of the inappropriate algorithms chosen for scheduling and dispatch. They utilized an algorithm that sent the nearest ambulance first, leading to ambulances getting farther and farther out of their „home" regions.

The Turn-around While it is true that the CAD system"s software errors demonstrate „carelessness and lack of quality assurance of program code changes," had the LAS paid more attention to selection process of the developing organization and imposed more reasonable and realistic expectations, the CAD system likely would not have failed as it did (Page et al , 1993). In 1996, a new Management of LAS adopted a new approach to the CAD system. A participative approach utilizing prototyping was adopted to help involve the users and instil ownership and acceptance of the system.

A very slow and deliberate approach was adopted and provided time for participation and iteration. A great deal of attention was paid to testing and training and the system was only implemented when it was felt to be ready, not just on a technical sense but when the users were convinced about the its capabilities (Fitzgerald and Russo, 2005).

In relation to technology, a new hardware platform was chosen, as the old system was essentially a PC architecture, which was not thought to be adequate for a command and control system (Fitzgerald and Russo, 2005). The new system went live on 17th January 1996. The initial system was a very basic system enabling the operators to receive a call and enter the details of an incident directly into the system. The computerized Gazetter, using a Postal Address File, looked up the

location and provided a map reference within one second. The system provided the controller with information on which of the 150 or so deployment points around London was the nearest to the incident and the controller would then dispatch an appropriate ambulance. The system additionally provided the location of the nearest hospital for the controller to pass to the ambulance crew. The new CAD system was implemented with few problems and it provided immediate benefits (Fitzgerald and Russo, 2005). A significant enhancement was introduced in September 1996, when „early call viewing“ was introduced. Once the call-takers had established the address of the incident that information was immediately made available to the controllers to begin the dispatch process, that is, before the call had finished. The result was that the annual performance rates improved significantly (Fitzgerald and Russo, 2005).

Learning Activities

We shall use a class timetabling system as our case study for this unit. The new class timetabling system is aimed at minimizing timetable class clashing while maximizing the resources of the university in a minimal amount of time. The system should be implemented to run on the university LAN and each university member should use their login ID and password to access the system; in case the user login details are wrong, the system should indicate to the user an error message. The system should be online 24 hours a day. A user should also be able to print or save the timetable to an external disk such as a flash drive. The system should be able to indicate an error message if the specified printer or disk is not found. The actions that the users can perform include:

1. Querying the times that a class is.
2. Querying the room that a class is in.
3. Free times at a particular day or period.
4. Free rooms at a particular day or period.
5. Query the number of classes taking place at a particular time.

Any of these queries should return either a positive or a negative response, for example, there may be no free rooms for the period of time the user specified. The class timetabling system should be updated only by the timetable master. Other users should not be able to make any changes to the timetable. The timetabling master can update the timetable by inputting new data or modifying existing data.

The system should display an alert if data has been input or modified but no update has been called to be done on the timetable. While updating or modifying the timetabling, there might be some conflicting data required for the timetable to be scheduled properly, for example, when two instances of the room Lab6 have been entered. The system should be able to display an alert when this happens.

Revision Questions

1. Explain the main stages of systems development from an object-oriented point of view.
2. Explain the significance of each of the following views in the development of a system:
    a) Use case view
    b) Analysis view
    c) Design view
3. Describe the principal characteristics of an object-oriented software system