COM201: System Analysis and Design

Lecture 2: Building Information Systems

Dr. Fayza A. Nada

1. **Title**
2. **Overview**
3. **Purpose:** Provides key definitions for systems development, projects and main aims.

- **Notes:**
    - Analogous to engineering and architectural projects. We have just looked at a few
    - Analogy with architecture. Don't build until designs complete and signed off.
        - Why? Hard to start again or fix up a house once building has begun.
    - Rigorous, systematic, robust method for constructing an information system.
    - Why, let's have a look at some problems related to poor development.
    - What is the product in this case? → Product is obviously an information system consistent with the goals of an enterprise
    - **Goals:**
        - Deliver on budget
        - on time
        - meets the client's requirements (i.e. what they want)

4. **Purpose:** Explains what constitutes project failure on top of failure to meet key aims.

Notes:

- Unreliable (characteristics of successful systems).
- No system delivered at all for whatever reason. Hard-pressed to get a job if that happened.
- Let's look at some spectacular failures

5. **Purpose:** A quick note on characteristics we might expect from successful systems.

Notes:

- Rather than projects, how about the key output of a systems development project – the system.

- **Maintainability:** It should be possible for the software to evolve to meet changing requirements.
- **Dependability:** The software should not cause physical or economic damage in the event of failure.
- **Efficiency:** The software should not make wasteful use of system resources.
- **Usability:** Software should have an appropriate user interface and documentation. → People can actually use it.
- **Correctness:** A program is functionally correct if it behaves according to the specification of the functions it should provide. → Supports requirements.
- **Reliability:** Software is reliable if the user can depend on it or the software will operate as expected over a specified time interval (MTTF).
- **Robustness:** A program is robust if it behaves "reasonably," even in circumstances that were not anticipated in the requirements specification.
- **Performance:** A software system is efficient if it uses computing resources economically.
- **User Friendliness:** A software system is user friendly if its human users find it easy to use. Subjective interpretations. One test is - can a novice user use it?
- **Evolvability:** Software products are modified over time to provide new functions or to change existing functions.

6. **Purpose:** Introduces **SDLC** framework for the first time.

Notes:

- Basis of many projects since the 1970s.
- Almost always discussed in conjunction with systems development.
- Variations such as planning, programming, information gathering, deployment, management.
- Note that these phases will be covered in more detail in subsequent slides and future lectures.
- Not unusual to include `feasibility study' or `planning' as separate phase.
- Conventional systems development.
- Sometimes mistakenly called the waterfall model (see later).
- Important to notice arrows indicating orderly progress through phases.
- In the past the arrows went one way, now there are iterations included.
- Complete phase and then proceed to next one.

- If problems are not picked up in early stages it much harder to fix later on (e.g. house → one extra bedroom).

7. **Purpose:** Introduces the **<u>first phase - project planning</u>**

Notes:

- Designed to assess feasibility of implementing solutions for business problems.
- Incorporates `planning' and 'feasibility' in here. Important before proceeding.
- <u>Focus on this course is software rather than hardware</u> but we will talk architectures later.
- Initial examination of problem domain
- <u>Might ask the following questions</u>:
  - What primary problems might a new or enhanced system solve?
  - What opportunities might a new or enhanced system provide?
  - What new hardware, software, databases, or procedures will improve an existing system?
  - What are the potential costs (variable and fixed)?
  - What are the associated risks?
  - ***You will practice the planning phase in <u>Tutorial 1.</u>***

8. **Purpose: <u>Introduces the second phase - analysis</u>**

Notes:

- Examining of existing systems, which begins once approval for further study is received from management
- Requirements analysis identifies user, stakeholder, and organizational needs for the new or modified system. This involves studying problems they are having with the current system and improvements they suggest.
- **<u>Things of interest</u>**:
  - Functional requirements (see next lecture)
  - Constraints
  - Types of information and relationships.
  - Exceptional conditions
  - Problems with current manual or computer methods.

- When a system is clear-cut and users clearly understand their needs for the new system, asking them to tell you works well. However, when needs aren't so clear, the systems analyst must find other ways to elicit requirements.
- The IS plan addresses long-term IS requirements. If this is referred to when identifying requirements for a specific system, it is more likely that the system will fit into the long-term plan.
- **Outputs of analysis**:
  - Strengths and weaknesses of the existing system.
  - The user/stakeholder requirements for the new system.
  - The organisational requirements for the new system.
  - A description of what the new information system should do.
- What the system needs to do not how?
- ***You will practice the design phase in Tutorial 2.***

9. **Purpose: Introduces the third phase – design**.

Notes:

- Prepare the detailed design needs for a new system or make modifications to an existing one.
- How the system is going to do it (support business activities, implement requirements).
- Effectively the blueprints that coders (programmers) will work from of the new system or modifications of existing one (recall architecture analogy)
- Represented as models, specifications and so on (designs frozen) ← One INCIS problem was ever-changing requirements.
- **Once these are correct and signed off, the construction can proceed**.
- **Details can include: inputs, outputs, user interfaces, HW, SW, database, telecommunications, etc**.
- **We are interested in SW related activities**.
- **Conceptual:** Abstracting real world phenomena (vanilla specification, no sw or hw selected).
- **Logical:** Planning the purpose of each system element (tying it to SW and HW). What system will do?
- **Physical:** How the tasks are accomplished, including how the components work together and what each component does.

- System interfaces – integrate functions and modules
- ***You will practice the design phase in <u>Tutorial 3</u>***

10. **Purpose: <u>Introduces the fourth phase - implementation.</u>**

Notes:

- Install system and make everything, including users, ready for its operation.
- Coding or construction takes place here (builders building a house).
- Make system operational.
- Deploy the system.
- Different methods of installation.
- Other activities:
    - Hardware acquisition
    - Software acquisition
    - Site and data preparation
    - Start up
- ***You will practice the design phase in <u>Tutorial 4</u>***

11. **Purpose: Introduces the last phase – Support (or maintenance)**

Notes:

- Checking, changing, and enhancing the system to make it more useful in obtaining user and organizational goals.
- Adds to the useful life of a system:
- Benefits from same rigorous methods and project management techniques applied to systems development.
- Checking, changing, and enhancing system
- Can consume large amounts of resources
- Resource = developer time mostly.
- Should continue to be a rigorous process and well documented.
- By far the largest chunk of the development process is devoted to maintenance because it continues after project is complete for the lifetime of the system.

- We will discuss some reasons for this phase shortly and you can see why it takes a lot of resource.
- Related to 'software evolution'..
- Environmental effects: change in law, improvements in the way they do things, more customers, expansion and so on.
- New requests can be activated by some of the other items (e.g., Gov. Regulations, mergers).
- Bug fixes. Problems only identified after system operational.

12. **Purpose:** This slide discusses the really important aspect of all development projects – **<u>the people involved.</u>**

Notes:

- Recall jigsaw model from first lecture.
- One component is people and our earlier discussion of building projects
- People are vitally important. Development approaches in recent times try to include clients, users and so on wherever possible.
- Analogy (architecture):
    - Architect (+ team)
    - Client (wanting house)
    - Builders
    - Plumber
    - Electrician
    - Council representative (building consent, codes of compliance)
    - Project manager
- **Stakeholders:** Individuals that benefit or are affected in some way by the results of a SD project
    - Example: Clients → pay for project.
    - All could be called stakeholders and certainly are
- **Users:** interact with the system regularly. As name implies use system
    - Specialists on both sides of the fence (domain experts, database guru).
- **Managers:** Management, of CFO, CIO, in charge of functional area like Accounts, Sales. Paying for development

- **Systems development specialists:** system analysts, programmers, requirements engineers, testers, documents, database designers, etc. you.
- **Project managers:** coordinate development process, run project, organise development specialists) → next week's lecture discusses project management.
- **Support personnel:** technical support, run the system when it is implemented
- **External:** Shareholders, councils, standards bodies, etc.

13. **Purpose:** Provides categories of different software tools used in the development process.

Notes:

- Support phases of SDLC
- Techniques are effectively the main activates associated with an ordered and robust development project
- Image: **Gantt chart** used by project managers to group various activities and schedule them

14. **Purpose:** Provides definitions of CASE.

Notes:

- Other acronyms (Computer assisted SW engineering, …).
- Comprehensive.
- Collection of programs.
- Support or automate design, analysis, translation (transformation).
- Think of CAD (Computer Aided Design) for architecture → same thing.
- Support engineering-type discipline through:
    - Common techniques
        - Standard methodologies
        - Automated tools
- Facilitate design and construction of system specifications
- Generate forms, reports, and code
- Typically based around a particular design paradigm