System Analysis & Design

CSCI 2783

The Systems Development Life Cycle or SDLC

The Systems Development Life Cycle (SDLC) is a software engineering framework that is used to describe the various phases used to develop an information system. These phases include planning, analysis, design, development, testing, and implementation. SDLC environments describe the activities and tools required to perform a particular process within the SDLC. They are also defined as controlled points where software engineers can carry out activities related to development, testing, installation, and configuration. These environments are associated with the different phases that make up the SDLC.

The Systems Development Life Cycle or SDLC

The main SDLC environments include:

- The analysis and design environment
- The development environment
- The common build environment
- The testing environment, which has two components:
  - ✓ The systems integration testing environment
  - ✓ The user acceptance testing environment
- The production environment

Analysis & Design Environment

The analysis and design environment is aligned to the planning and analysis phases of the SDLC. In this environment, the main processes that take place include carrying out an in-depth examination of the current system and the proposed system. The system architecture is also defined and includes developing the design of the hardware, software, and network requirements for the system. Within this environment, systems and business analysts work closely with software engineers.

Development & Common Build Environment

The development environment is aligned to the development phase of the SDLC. This is where processes related to software development are carried out. The development environment contains a set of different processes and tools for programming. These are used to develop the final software.

The development environment can also be a physical space where development takes place and where software engineers interact. Another example of the development environment is the integrated development environment (IDE). The IDE provides a platform where tools and development processes are coordinated in order to provide software engineers a convenient way of accessing the resources they require during the development process.

Development & Common Build Environment

The common build environment is closely aligned to the development phase of the SDLC. In this environment, software engineers merge the work done in the development environment. Within this environment, software engineers build systems. These are used to automate the process of software compilation.

Testing Environment

The testing environment is closely aligned to the testing phase of the SDLC. The testing environment comprises the following components: the System Integration Testing Environment and the User Acceptance Testing Environment.

The system integration testing environment includes the testing of the entire system being developed. This includes a complete test of the modules making up the software. This environment controls processes involved in assembling parts of the system in a manner that is cost-effective and logical and then comprehensively checking the manner in which the system executes. It involves testing all functionalities of the system.

The Origin of Software

Software emerged in 1948 in England as more engineers and developers sought to design systems that could address large scale operations for companies. However, the theory of software was credited much earlier to a mathematician, computer scientist, and logician, Alan Turing, in 1935. Tom Kilburn, a computer scientist, wrote the world's first software piece for a computer he had built with his friend Freddie Williams, called the Manchester small scale experimental machine, aka Baby.

The Origin of Software

The software created by Tom was programmed to perform mathematics based on code instruction and data. On 21st June 1948, at 11 am, Tom ran the first software piece at the University of Manchester in England. It reportedly took 52 minutes for the machine to compute the initial mathematical request, which was to conjure the greatest divisor of 2 to the power of 18.

The Origin of Software

After this software emerged, numerous advancements followed, such as;

- Punch cards used to program computers and denote instructions.
- Fortran, a programming language, was published in 1957.
- Other distinctive programming languages developed over the years, including BASIC, Pascal, Cobol, and C.

The Origin of Software (PCs)

Once personal computers launched in the 70s and 80s, software improved dramatically. For instance, in 1977, the Apple 2 computer was released to the public with a popular app called VisiCalc. VisiCalc was the first spreadsheet for PCs, and it was very popular. Over time other businesses created their own PCS. For instance, the IBM PC was launched in 1981. At this time, software was primarily created with business functions in mind. Popular applications, including AutoCAD, Microsoft Excel, and Microsoft Word, launched in the mid-80s.

The Origin of Software (Mobile Phones)

The first mobile phone took place on 3rd April 1973. Later on, in 1993, IBM released the first smartphone to the public, and then in 1996, the Personal Digital Assistant emerged in the market. Other notable mobile phone releases include the Blackberry 850 devices in 1999 and 2007 when Apple released their first mobile – the iPhone. At this point, mobile application software took the world by storm.

The Origin of Software (Present)

Today, every business is using software to some degree. Be it a small furniture restoration company using accounting software to keep their books afloat. Or a gaming company is incorporating poker software for their audience. Retail, finance, transportation, healthcare, and so forth all require software to some extent.

The Origin of Software (Present)

Here are a few reasons why;

- Software can enhance productivity by completing arduous, time-consuming, routine administrative duties and accounting tasks.
- Cyber security software is advantageous for companies with important intel stored on their business devices. Enabling users to detect and eliminate any threats that could jeopardize the livelihood of their company.
- Spreadsheets like Microsoft Excel and OpenOffice enable businesses to collect, record, edit, organize and put data into a readable format.

The Origin of Software (Present)

Moreover, software’s total capacity is yet to be reached, with operating systems experiencing continuous improvement in writing software and other apps.

The Origin of Software (Future)

As such, with software penetrating almost every industry imaginable, it’s expected that more operations will become mobile-based.

Thus, anyone who has a new digital concept will likely source app software development services to bring their idea to life.

Another burning question regarding software development today is whether AI could eventually take over the role of software engineers.

The Origin of Software (Future)

Another burning question regarding software development today is whether AI could eventually take over the role of software engineers.

The answer so far is while more routine tasks will be passed over to computers over the years. There will still be room for engineers to implement solutions, research, and new software concepts.

“**One machine can do the work of fifty ordinary men. No machine can do the work of one extraordinary man**.” – Elbert Hubbard (Author)

The Origin of Software (Future)

Moreover, there won’t necessarily be a dip in need for software developers, but a change in demand for experience and qualifications in the following trending software services;

- Cloud-based services
- Voice software
- Blockchain technology
- Artificial intelligence
- Machine learning
- Virtual and augmented reality
- New programming languages

The Origin of Software (Future)

Today, software is often taken for granted by personal and business users alike. Because as technology advances, our expectations of devices rises.

And continuously improving software is paramount for inventors and customers alike. To ensure demand for faster, more powerful, intelligent software that can take care of the intricate work or personal tasks people need to do can be completed without little input on their behalf.

Managing the Information System

A critical success factor for effective time management is timely communication for standards and expectations. Setting milestones provides appropriate reference points for the project manager and team members. The most widely used tools for scheduling, monitoring, and communicating time aspects of projects are the Program Evaluation Review Technique (PERT) and the Critical Path Method (CPM).

Managing the Information System

Potential for PERT / CPM

- Estimate minimum time required for competing the entire project.
- Identify critical activities that must be completed in time for the entire project to be completed as scheduled.
- Show progress status for critical activities.
- Show progress status for noncritical activities.
- Estimate the length of time that these noncritical activities can be delayed.
- Estimate the likelihood for completing the entire project on schedule.

Managing the Information System

Time is a resource if it is managed effectively; otherwise it will be a constraint. Timely delivery of information systems projects has been one of the biggest challenges for information systems project managers. Managing time effectively is therefore a critical component for project success. Time management relates not only to the anticipated planned activities but also to unexpected events – last minute changes, personal issues, etc.

Managing the Information System

A successful project is the one that is on time, within budget, and delivers what is expected. Project managers should set the standard for a timely outcome by example. If project managers cannot control their time, then they will have difficulties controlling team members, and consequently the entire project is likely to be late.

Managing the Information System

Project managers often work on tight deadlines and feel they have no time to think about time and its effective use. To be effective, a project manage must be organized and prioritize work. Depending on work habits, this could be done in different ways and may or may not be very formal.

Managing the Information System

Questions to Ask when Assessing Your Time Management

- Do you spend a lot of time responding to email messages?
- Do you spend a lot of time returning calls?
- How often do you work overtime?
- How often do you miss social events?
- How often do you reschedule your appointments?
- How often do you feel you need a large block of time to finish a task?
- Do you have a gatekeeper for unexpected visitors who take up your time?
- Do you prioritize your work? Based on what?
- Do you plan your vacation?

Managing the Information System

Daily Activity Form

Activity tracking for project .....n a m e.....				
Activity	Date Required	Duration	Start Date	Status

Managing the Information System

Sample Activity Form for Developing a Personal Web Page

Activity	Activity Description	Duration (Days)	Preceding Activities
A	Determine user needs	2	–
B	Review software and languages	2	–
C	Purchase software	1	B
D	Design format and style	3	A, C
E	Write programs	5	D
F	Review outcome product with user	1	E
G	Make revisions	2	F
H	Select server site	1	–
I	Install on server and test	2	G

Managing the Information System

Managers, including project managers, are said to spend most of their time in meetings. Many managers would argue that they attend too many meetings and most meetings are too time-consuming. What is important is not so much the number of meetings one attends or the amount of time spent in meetings but what is accomplished in relation to time spent.

Managing the Information System

Principles About Meeting Management

- Longer meetings do not necessarily produce better results
- The need for an agenda that is communicated to all
- The need for continued focus and control
- Opportunity for participation by all
- Summation of outcome and closure