**What is Software Life Cycle?**

The system development life cycle is the process of determining how an information system can support business, designing the system, building it, and delivering it to users. Now that you have done a few projects yourself you may think this is a simple process. In the real world, however, it is not so easy.

In 2010, an estimated $2.4 trillion was spent by organizations and governments on IT hardware, software, and services worldwide. This spending level was projected to increase by 3.5% in 2011. 1 Unfortunately, a study conducted in 2008 found success ‘improbable’, in 68% of technology projects. Many of the systems that aren’t totally abandoned are delivered to the users significantly late, cost far more than expected, and have fewer features than originally planned.

Today, both businesses and governments experience embarrassing and costly errors in their information systems. Here is a sample of just a few notable software glitches that occurred in 2010:

* A software error resulted in Toys R Us double billing some shoppers for purchases made on black Friday.
* McAfee’s anti-virus software product caused its users’ computers to lock up
* McAfee offered affected customers a free 2-year subscription and reimbursement for costs incurred to repair the machines.

**System analyst**

The key person in the SDLC is the system analyst, who analyzes the business situation, identifies opportunities for improvements, and designs and information system to implement the improvements. Many system analysts view their profession as one of the most interesting, exciting and challenging jobs around. As a system analyst, you will work as a team with a variety is people, including business and technical experts. You will feel the satisfaction of seeing systems that you designed and developed make a significant positive business impact, while knowing that your unique skills helped make that happen. It is important to remember that the primary objective of the systems analyst is not to create a wonderful system.

The primary goal is to create value for the organization, which for most companies means increasing profits. Many failed systems were abandoned because the analysts tried to build a wonderful system without clearly understanding how the system would support the organization’s goals, improve business processes, and integrate with each other information systems to provide value. An investment in an information system is like any other investment, such as a new machine tool. The goal is not to acquire the tool, because the tool is simply a means to an end; the goal is to enable the organization to perform work better so that it can earn greater profits or serve its constituents more effectively.

Business System Analyst

Gleeson Recruitment Group

West Bromwich

**Job Overview**

Key member of the IT Department, role to manage and lead NAV ERP and other business system customization and effective utilization of the system and reporting tools.

This is a systems analyst role with some BA and report writing responsibilities, looking for an all-rounder.

Central team member of Navision re-implementation project to move from customized MS Dynamics NAV 2017 to current standard version.

Provide business support through the scoping, implementation, and maintenance of business applications.

40k-50k annually

The system analyst role focuses on the IS issues surrounding the system. This person develops ideas and suggestions for ways that IT can support and improve business processes, helps design new business processes supported by IT, designs the new information system, and ensures that all IS standards are maintained. The systems analyst will have significant training and experience in analysis and design and in programming.

The business analyst role focuses on the business issues surrounding the system. This person helps to identify the business value that the system will create, develops ideas for improving the business processes, and helps design new business processes and policies. The business analyst will have business

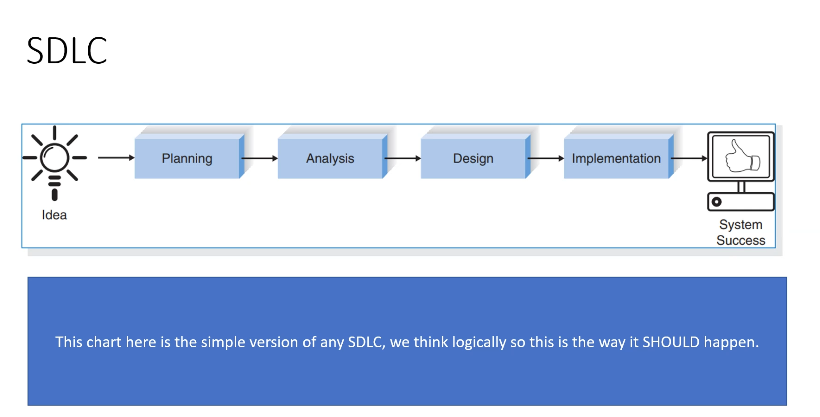
training and experience, plus knowledge of analysis and design.

The requirements analyst role focuses on eliciting the requirements from the stakeholders associated with the new system. As more organizations recognize the critical role that complete and accurate requirements play in the ultimate success of the system, this specialty has gradually evolved. Requirements analysts understand the business well, are excellent communicators, and are highly skilled in an array of requirements elicitation techniques.

The infrastructure analyst role focuses on technical issues surrounding the ways the system will interact with the organization’s technical infrastructure (hardware, software, networks, and databases). This person ensures that the new information system conforms to organizational standards and helps to identify infrastructure changes that will be needed to support the system. The infrastructure analyst will have significant training and experience in networking, database administration, and various hardware and software products.

The change management analyst role focuses on the people and management issues surrounding the system installation. This person ensures that adequate documentation and support are available to users, provides user training on the new system, and develops strategies to overcome resistance to change. The change management analyst will have significant training and experience in organizational behavior and specific expertise in change management.

The project manager's role ensures that the project is completed on time and within budget and that the system delivers the expected value to the organization. The project manager is often a seasoned systems analyst who, through training and experience, has acquired specialized project management knowledge and skills.



SDLC[Plan]

|  |  |  |  |
| --- | --- | --- | --- |
| Phase: | Step: | Technique | Deliverable |
| Focus: Why build this system?  How to structure the projects?  Primary outputs:  System request with feasibility study  Project plan | Identify opportunity  Analyze feasibility  Develop workplan  Staff project  Control and direct project | Project identification  Technical feasibility  Economic feasibility  Organizational feasibility  Time estimation  Task identification  Work breakdown structure  Gantt chart  Scope management  Project Staffing  Project charter  CASE repository  Standards  Documentation  Time boxing  Risk management | System request  Feasibility study  Project plan  Workplan  Staffing plan  Standards list  Risk management |

**SDLC[Analysis]**

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Step | Technique | Deliverable |
| Focus: who, what, where, and when for this system?  Primary output – System proposal | Develop analysis strategy  Determine business requirements  Create use cases  Model processes  Model data | Business process automation  Business process improvement  Business process reengineering  Interview  Questionnaire  Document analysis  Observation  Use case analysis  Data flow diagrams  Entity relationship modelling  Normalization | System proposal  Requirement definition  Use cases  Process models  Data models |

**SDLC[Design]**

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Step | Technique | Deliverable |
| Focus: how will this system work  Primary output: System specification | Design physical system  Design architecture  Design interface  Design programs  Design database and files | Design strategy  Architecture design  Hardware and software selection  Use scenario  Interface structure  Interface standards  Interface prototype  Interface evaluations  Data flow diagrams  Program structure chart  Program specification  Data format selection  Entity relationship diagram  Denormalization  Performance tuning  Size estimation | Alternative matrix  System specification  Architecture report  Hardware and software specification  Interface design  Physical process model  Program design  Database and file specification  Physical data model |

**SDLC[Implementation]**

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Step | Technique | Deliverable |
| Focus: delivery and support of completed system  Primary output: installed system | Construct system  Install system  Maintain system  Post implementation | Programming  Software testing  Performance testing  Conversion strategy selection training  Support selection  System maintenance  Project assessment  Post implementation audit | Test plan  Programs  Documentation  Migration plan  Conversion plan  Business contingency plan  Training plan  Support plan  Problem report  Change request  Post implementation audit report |

**Planning**

The planning phase is the fundamental process of understanding why an information system should be built and determining how the project team will go about building it. It has two steps:

1. During project initiation, the system’s business value to the organization is identified-how will it lower costs or increase revenues? Most ideas for new systems come from outside the IS area (from the marketing department, accounting department, etc.) in the form of a system request. A system request presents a brief summary of a business need, and it explains how a system that supports the need will create business value. The IS department works together with the person or department generating the request (called the project sponsor) to conduct a feasibility analysis. The feasibility analysis examines key aspects of the proposed project:

* The technical feasibility (Can we build it?)
* The economic feasibility (Will it provide business value?)
* The organizational feasibility (If we build it, will it be used?)

The system request and feasibility analysis are presented to an information systems approval committee (sometimes called a steering committee), which decides whether the project should be undertaken.

1. Once the project is approved, it enters management. During project management, the project manager creates a work plan, staffs the project, and puts techniques in place to help the project team control and direct the project through the entire SDLC. The deliverable for project management is a project plan that describes how the project team will go about developing the system.

Identify the business requirements such as the new software and amending/increasing security.

A good first step to identifying all your business requirements is to interview the people who know your business capabilities (what you do) and the processes and systems underlying them (how you do them)

Then, hold follow-up focus groups and workshops with groups of employees to fill in the gaps and identify potential points of friction and additional value. They may even identify new requirements that could improve processes without additional software investment.

Once the employees have identified and documented business requirements, you should share their findings with other stakeholders for review. This process can help indicate which requirements are key and which you can eliminate. You are looking for requirements that:

* Deliver competitive advantage
* Maximize the business outcomes critical to your stakeholders
* Automate and simplify processes

You should also look carefully at requirements that you can meet with the configuration of off-the-shelf solutions rather than customization of more unique solutions.

As for security it’s a must that the company policy is followed and regular updates to the user groups, access authorizations, password changes and admin privileges are checked and amended over time.

Assess the measurable value for the proposed solution in relation to:

User

The client/business

* Scale – What's measures (units)
* Meter – How it’s measured (method)
* Targets – Levels aiming to achieve
* Constraints – levels trying to avoid
* Benchmark – current or past performance levels
* Qualifiers – dates, places or events useful for clarification
* Sources – origin of information for transparency and credibility

Functional – What it does and meant to do

Non-function – How it does the thing it’s meant to do

KPI – Key Performance Indicator

If you cannot measure it, you cannot improve it, KPI’s assist you in measuring the success of your progress in measuring your team. So, it’s easier to link your team objectives to the business strategic goals. There are many useful KPI’s to measure a successful software development process. Such as:

Cycle time – You can use this KPI to measure the team's speed and productivity. This is the KPI that takes the time it’s needed for a bug, feature or task to move from the first commit to production release. It tells you when you have a bottleneck, blocker, or breakdowns in your project. For example, the project may be blocked for lack of QA staff so cycle time helps find these issues early in the process.

Sprint Burndowns – This KPI measures the rate at which the work is completed and how much of it remains to be done. A burn out chart is typically used to keep track of this so you can realize whether the team is running on schedule or not. It may also help plan the next sprint and delegate the work.

Velocity – You can determine how much value is delivered by the team by measuring the velocity KPI. It monitors your team's ability to accomplish each sprint or moving things from TO DO to DONE.

The team will need to define the meaning of done for each task this way it’s not ambiguous. This KPI will help set realistic sprints, set delivery expectations and spot challenges.

* Identify the performance constraints in digital projects – Think of time, cost, resources and scope of the project. You may have issues with target markets' availability to certain technology or platforms.
* Create user acceptance criteria
* Schedule projects (tasks, subtasks, milestones)
* Allocating appropriate resources to digital projects
* Estimate costs of digital projects

Schedule projects (tasks, subtasks, milestones) – time estimation – task management.

Allocate appropriate resources to digital projects – rights, licenses, skill work, digital assets, source code, documentation, bug log and others.

Estimate costs of digital projects – Remember this is the estimation not the actual cost of the project yet. Think what this project cost would IF worked, teams, cost of resources and any other work that the client requires. This is not about showing you can do it for cheap and charge more later. You need to be able to negotiate the best price for the project and deliver accurately to maintain business relation. Clients can always pull out if they are unhappy with the price or the work so professionalism and transparency is absolute in this process.

* Choose programming languages for digital projects based on key criteria.
* Suitability for the proposed task
* Organizational policy
* Scalability – can you add more features to it?
* Security
* Availability
* Costs
* Reliability
* Identify risks and explore ways to migrate the risks

**Analysis**

An analysis strategy is developed to guide the project team’s efforts. Such a strategy usually includes a study of the current system (called the as-is system) and its problems, and envisioning ways to design a new system (called the to-be system)

The next step is requirements gathering (e.g., through interviews, group workshops, or questionnaires). The analysis of this information – in conjunction with input from the project sponsor and many other people. Leads to the development of a concept for a new system. The system concept is then used as a basis to develop a set of business analysis.

The analyses, system concept, and models are combined into a document called the system proposal, which is presented to the project sponsor and other key decision makers

**Design**

The design phase decides how the system will operate in terms of the hardware, software, and the network infrastructure that will be in place; the user interface, forms, and reports that will be used; and the specific programs. Databases, and files that will be needed. Although most of the strategic decisions about the system are made in the development of the system concept during the analysis phase, the steps in the design phase determine exactly how the system will operate. The design phase has four steps.

The design strategy must be determined. This clarifies whether the system will be developed by the company’s own programmers, whether its development will be outsourced to another firm, or whether the company will buy an existing software package.

This leads to the development of the basic architecture design for the system that describes the hardware, software, and network infrastructure that will be used. In most cases, the system will add to or change the infrastructure that already exists in the organization. The interface design specifies how the users will move thorough the system (e.g., by navigation methods such as menus and on-screen buttons) and the forms and reports that the system will use.

The database and file specifications are developed. These define exactly what data will be stored and where they will be stored.

The analyst team develops the program design, which defines the program that need to be written and exactly what each program will do.

This collection of deliverables (architecture design, interface design, database and file specification, and program design) is the system specification that is sued by the programming team for implementation. At the end of the design phase, the feasibility analysis and project plan are re-examined and revised, and another decision is made by the project sponsor and approval committee about whether to terminate the project or continue

**Implementation**

The final phase in the SDLC is the implementation phase, during which the system is actually built (or purchased, in the case of a packaged software design and installed). This is the phase that usually gets the most attention, because for most systems it is the longest and most expensive single part of the development process. This phase has three steps:

System construction is the first step. The system is built and tested to ensure that it performs as designed. Since the cost of fixing bugs can be immense, testing is one of the most critical steps in implementation. Most organizations spend more time and attention on testing than on writing the programs in the first place.

The system is installed. Installation is the process by which the old system is turned off and the new one is turned on. There are several approaches that may be used to convert from the old to the new system. One of the most important aspects of conversion is the training plan, used to teach users how to use the new system and help manage the changes caused by the new system.

The analyst team establishes a support plan for the system. This plan usually includes a formal or informal post-implementation review, as well as a systematic way for identifying major and minor changes needed for the system.

Abstraction – Moving unnecessary data

Encapsulation – keep private things private and public things public

Inheritance – One base class has all the properties you need for an object and other classes can inherit something from that one class

Polymorphism – a class can be implemented in its own way

**Revision:**

Test driven development – It is a software development practice that focuses on creating unit test cases before developing the actual code. It is an iterative approach that combines programming, the creation of unit tests, and refactoring.

The 5 steps of test-driven development are:

* Add a test after reading and understanding and process the feature or bug request.
* Write a unit test and run all tests and see if any new test fails.
* Write the code that fulfills the requirements. Run all the tests and if they pass, they don’t need to repeat this step.
* Run tests, clean up and Refactor code.
* Rinse, lather and Repeat.

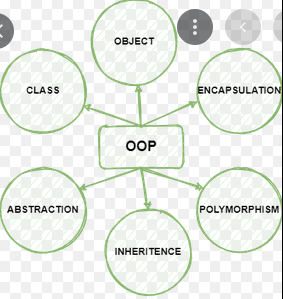
Object oriented programming - What is object-oriented programming Python?

In Python, object-oriented Programming (OOP) is a programming paradigm that uses objects and classes in programming. The four pillars of the OOP are:

* Abstraction – Abstraction is the concept of wrapping up complex actions in simple verbs. Describe each thing you’ve abstracted clearly and hide the complexity. Someone can always check for the details whenever they wish.
* Encapsulation – Encapsulation means that each object in your code should control its own state. State is the current "snapshot" of your object. The keys, the methods on your object, Boolean properties and so on. If you were to reset a Boolean or delete a key from the object, they're all changes to your state.
* Polymorphism – Polymorphism means "the condition of occurring in several different forms." That's exactly what the fourth and final pillar is concerned with – types in the same inheritance chains being able to do different things.
* Inheritance – It lets lone object acquire the properties and methods of another object. It is most beneficial for its reusability as sometimes the same code can be used in other parts.

The OOPs concepts are as follows:

* Class
* Object
* Method
* Method passing



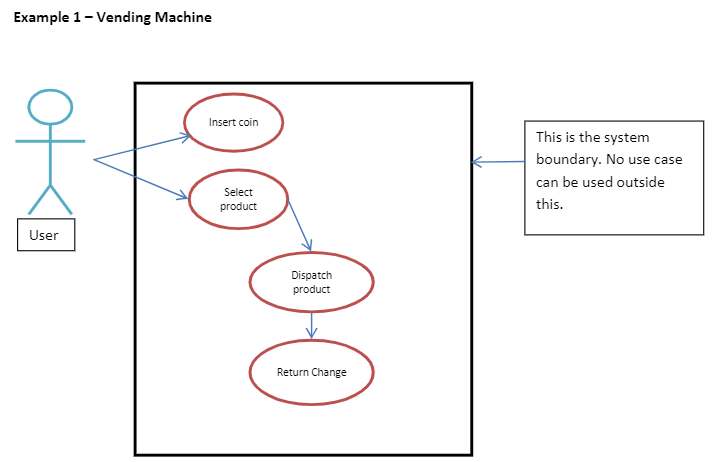
UML use case diagrams –

Actor – is the person that will potentially use the application. Could be the user, admin or employee.

Use case – This is the circle. It represents the use case of any software. This being the interaction or process which is happening between actor and the system. Any kind of interaction can pe pointed in this shape.

System – The rectangle/box. Basically, all the use case will be taken inside this box and actors are placed outside of it.

Package – Package a box with a small box on top of it letting you know it is a package. It is used to group classes together and show where it will be used. For example, if we have classes for easy, medium and hard enemies in a game we will group them using the package inside the system symbol.



Class diagrams -

A class diagram is a UML diagram type that describes a system by visualizing the different types of objects within a system and the kinds of static relationships that exist among them. It also illustrates the operations and attributes of the classes.

