**Software Engineering | Architectural Design**

Introduction: The software needs the architectural design to represents the design of software. IEEE defines architectural design as “the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.” The software that is built for computer-based systems can exhibit one of these many architectural styles.

Each style will describe a system category that consists of :

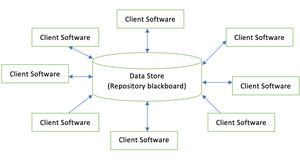
* A set of components (eg: a database, computational modules) that will perform a function required by the system.
* The set of connectors will help in coordination, communication, and cooperation between the components.
* Conditions that how components can be integrated to form the system.
* Semantic models that help the designer to understand the overall properties of the system.

The use of architectural styles is to establish a structure for all the components of the system.

Taxonomy of Architectural styles:

**Data centered architectures:**

* A data store will reside at the center of this architecture and is accessed frequently by the other components that update, add, delete or modify the data present within the store.
* The figure illustrates a typical data centered style. The client software access a central repository. Variation of this approach are used to transform the repository into a blackboard when data related to client or data of interest for the client change the notifications to client software.
* This data-centered architecture will promote integrability. This means that the existing components can be changed and new client components can be added to the architecture without the permission or concern of other clients.
* Data can be passed among clients using blackboard mechanism.
* Repository of data is independent of clients
* Client work independent of each other
* It may be simple to add additional clients.
* Modification can be very easy



**Data flow architectures:**

* This kind of architecture is used when input data to be transformed into output data through a series of computational manipulative components.
* The figure represents pipe-and-filter architecture since it uses both pipe and filter and it has a set of components called filters connected by pipes.
* Pipes are used to transmit data from one component to the next.
* Each filter will work independently and is designed to take data input of a certain form and produces data output to the next filter of a specified form. The filters don’t require any knowledge of the working of neighboring filters.
* If the data flow degenerates into a single line of transforms, then it is termed as batch sequential. This structure accepts the batch of data and then applies a series of sequential components to transform it.

**Advantage of Data Flow architecture**

 It encourages upkeep, repurposing, and modification.

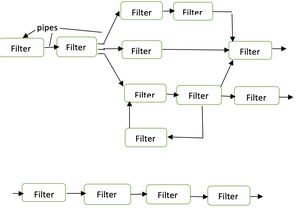
With this design, concurrent execution is supported.

**Disadvantage  of Data Flow architecture**

It frequently degenerates to batch sequential system

Data flow architecture does not allow applications that require greater user engagement.

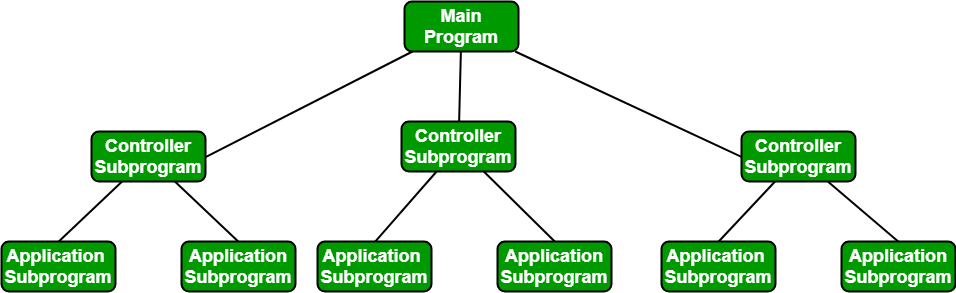
It is not easy to coordinate two different but related streams



**Call and Return architectures:** It is used to create a program that is easy to scale and modify. Many sub-styles exist within this category. Two of them are explained below.

**Remote procedure call architecture:** This components is used to present in a main program or sub program architecture distributed among multiple computers on a network.

**Main program or Subprogram architectures:** The main program structure decomposes into number of subprograms or function into a control hierarchy. Main program contains number of subprograms that can invoke other components. 



**Object Oriented architecture:** The components of a system encapsulate data and the operations that must be applied to manipulate the data. The coordination and communication between the components are established via the message passing.

           Characteristics of  Object Oriented architecture

Object protect the system’s integrity.

An object is unaware of the depiction of other items.

            Advantage of Object Oriented architecture

It enables the designer to separate a challenge into a collection of autonomous objects.

Other objects are aware of the implementation details of the object, allowing changes to be made without having an impact on other objects.

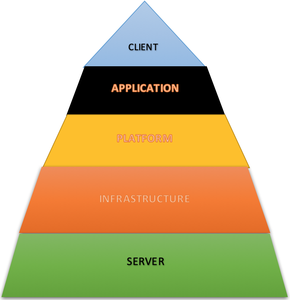
**Layered architecture:**

A number of different layers are defined with each layer performing a well-defined set of operations. Each layer will do some operations that becomes closer to machine instruction set progressively.

At the outer layer, components will receive the user interface operations and at the inner layers, components will perform the operating system interfacing(communication and coordination with OS)

Intermediate layers to utility services and application software functions.

One common example of this architectural style is OSI-ISO (Open Systems Interconnection-International Organisation for Standardisation) communication system.



**Software Engineering | Organizational Structure**

**Organization structure:**

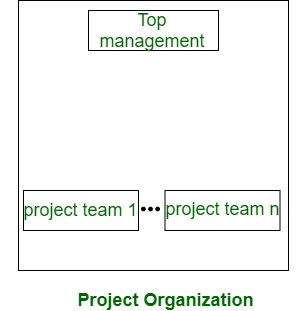
Usually, each software package development organization handles many projects at any time. Software package organizations assign totally different groups of engineers to handle different software projects. every sort of organizational structure has its own advantages and downsides that the issue “How is that the organization as a full structured?” should be taken into thought so each software package project is finished before its point in time.

There are basically 2 broad ways in which a software package development organization is structured: Project format, and Functional format. These are explained as following below.

**Project format:**

The project development workers are divided supported the project that they work (as shown below diagram). In the project format, a group of engineers is appointed to the project at the beginning of the project and that they stay with the project until the completion of the project.

Thus, the identical team carries out all the life cycle activities. Obviously, the functional format needs a lot of communication among groups than the project format, as a result of one team should perceive the work done by the previous groups.

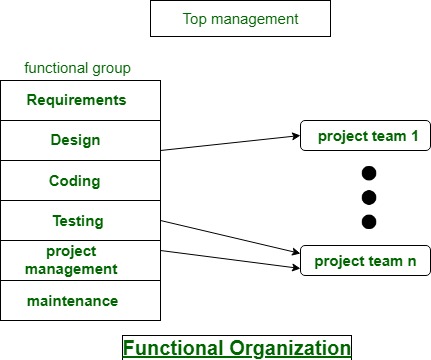


**Functional format:**

The event workers are divided supported the useful cluster to that they belong. the various project borrows engineers from the specified useful teams for specific parts to be undertaken within the project and come back them to the functional cluster upon the completion of the phase.

In the functional format, totally different groups of programmers perform different phases of a project. For example, one team may do the necessities specification, another do the planning, and so on. The partially completed product passes from one team to a different because the project evolves.

Therefore, the useful format needs significant communication among the various groups as a result of the work of 1 team should be clearly understood by the next teams engaged on the project. This needs sensible quality documentation to be made when each activity.



Refrences: -https://www.geeksforgeeks.org/software-development-organizational-structure/

**Software Engineering | Modularity and its Properties**

The module simply means the software components that are been created by dividing the software. The software is divided into various components that work together to form a single functioning item but sometimes they can perform as a complete function if not connected with each other. This process of creating software modules is known as Modularity in software engineering.

It simply measures the degree to which these components are made up than can be combined. Some of the projects or software designs are very complex that it’s not easy to understand its working and functioning. In such cases, modularity is a key weapon that helps in reducing the complexity of such software or projects.

The basic principle of Modularity is that “Systems should be built from cohesive, loosely coupled components (modules)” which means s system should be made up of different components that are united and work together in an efficient way and such components have a well-defined function. To define a modular system, several properties or criteria are there under which we can evaluate a design method while considering its abilities.

These criteria are defined by Meyer. Some of them are given below:

**Modular Decomposability –**

Decomposability simply means to break down something into smaller pieces. Modular decomposability means to break down the problem into different sub-problems in a systematic manner. Solving a large problem is difficult sometimes, so the decomposition helps in reducing the complexity of the problem, and sub-problems created can be solved independently. This helps in achieving the basic principle of modularity.

**Modular Composability –**

Composability simply means the ability to combine modules that are created. It’s actually the principle of system design that deals with the way in which two or more components are related or connected to each other. Modular composability means to assemble the modules into a new system that means to connect the combine the components into a new system.

**Modular Understandability –**

Understandability simply means the capability of being understood, quality of comprehensible. Modular understandability means to make it easier for the user to understand each module so that it is very easy to develop software and change it as per requirement. Sometimes it’s not easy to understand the process models because of its complexity and its large size in structure. Using modularity understandability, it becomes easier to understand the problem in an efficient way without any issue.

**Modular Continuity –**

Continuity simply means unbroken or consistent or uninterrupted connection for a long period of time without any change or being stopped. Modular continuity means making changes to the system requirements that will cause changes in the modules individually without causing any effect or change in the overall system or software.

**Modular Protection –**

Protection simply means to keep something safe from any harms, to protect against any unpleasant means or damage. Modular protection means to keep safe the other modules from the abnormal condition occurring in a particular module at run time. The abnormal condition can be an error or failure also known as run-time errors. The side effects of these errors are constrained within the module.

**Software Engineering | Reference architecture**

In the world of software engineering, **reference architectures** act as the**guideline for operations**. They provide the framework for processes and documentation. They also continuously enhance the efficiency and performance of your systems and processes. This leads to cost savings, better optimization, and increased productivity.

Discover what reference architecture is in this article, along with best practices and examples. Read on to learn all you need to know about this guideline!

## **What Is Reference Architecture?**

**Reference architecture**, also known as enterprise architecture, is a template or a set of documents. This template makes it easy for software architects and developers to address specific problems. For example, it could contain definitions for commonly-used words, an **architecture blueprint** with elements and relationships, data models, communication standards, recommended processes, etc.

The idea is to **make the job easier for anyone investigating problems** or implementing solutions in that domain. In short, enterprise architecture is a resource you can refer to while designing and implementing complex technology solutions. But you may ask yourself how this helps you and why you should have one in the first place. I’ll talk about this in the next section.

## **Why Is Reference Architecture Important?**

A reference architecture adds value to companies in the following ways:

* **Eliminates confusion** by standardizing language and organizational context
* **Makes problems easier to solve** by implementing clear guidelines
* **Provides resources** for designing an IT architecture, teams, and solutions
* **Saves time, effort, and money** by having the ability to tap into existing resources
* **Optimizes problem-solving** by providing standard best practices
* **Supports interoperability** and reusability of components

But how can you harness these benefits for your company? Let’s see how you can create your own enterprise architecture.

## **5 Steps to Build a Reference Architecture**

Creating your [**reference architecture**](https://techgenix.com/understand-hybrid-network-security-interactive-reference-architecture-diagram/) is challenging as no predefined steps or processes exist. However, below are **5 key steps** to help make the implementation process easier!

### **1. Identify the Purpose**

Firstly, ask yourself some questions to **identify the purpose** of the enterprise architecture:

* What is the **domain and scope** of the reference?
* Who are the **stakeholders**?
* How will you **use**this architecture?
* What are the **constraints**, **assumptions**, and **environments** associated with it?

Exploring the answers can help you create a pragmatic reference document.

### **2. Formulate the Principles**

Once you have the answers to the questions, start writing things you want to include in your enterprise architecture. Ensure you**align items with the** **foundational statements** and your**company’s culture and values**.

### **3. Lay Down the Technical Rules**

Now, you’re all set for the **technical aspects of your document**. The next step is to decide the [**frameworks and patterns**](https://techgenix.com/frameworks-and-standards/) you want to follow within your company. Ideally, include the rationale for these decisions so others can understand them. Add all the rules required to achieve the principles formulated earlier.

### **4. Build the Rules and Standards**

Next, build the **rules and standards** across different departments within the company. Come up with a list of possible challenges and choose the correct standard for each scenario.

### **5. Set the Context**

Finally, **narrow your enterprise architecture down** to include the vocabulary and context for each situation. Your document acts as a comprehensive reference point for everyone within your company. To this end, it starts with a **broad definition and systematically refines terms and actions** for each situation and context. That said, this isn’t the only way to build your enterprise architecture. Most companies follow a**logical process**, making it easy to incorporate industry best practices.

Now let’s discover the best practices you can adopt in your company.

## **Reference Architecture Best Practices**

When you use **best practices** to design your enterprise architecture, you ensure you leverage all its benefits. In particular, you can [**improve efficiency**](https://techgenix.com/dell-reduces-it-complexity-and-improves-energy-efficiency-with-new-data-center-infrastructure-products-507/), meet regulatory requirements, and reduce the chance of errors. Below is a brief look at best practices to optimize your solution:

* **Competitive advantage**; works best on elements within your company, like compliance
* **Benchmark metrics** within your industry; this helps to compare outcomes with competitors
* **Compliance standards** for regulatory mandates
* **Component reuse;**reducing organizational efforts on repetitive tasks
* **Governance** as a part of your reference architecture

Ref:- https://techgenix.com/reference-architecture-guide/

**Software Engineering | User Interface Design**

User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. UI brings together concepts from interaction design, visual design, and information architecture.

Choosing Interface Elements

Users have become familiar with interface elements acting in a certain way, so try to be consistent and predictable in your choices and their layout. Doing so will help with task completion, efficiency, and satisfaction.

Interface elements include but are not limited to:

Input Controls: buttons, text fields, checkboxes, radio buttons, dropdown lists, list boxes, toggles, date field

Navigational Components: breadcrumb, slider, search field, pagination, slider, tags, icons

Informational Components: tooltips, icons, progress bar, notifications, message boxes, modal windows

Containers: accordion

There are times when multiple elements might be appropriate for displaying content. When this happens, it’s important to consider the trade-offs. For example, sometimes elements that can help save you space, put more of a burden on the user mentally by forcing them to guess what is within the dropdown or what the element might be.

Best Practices for Designing an Interface

Everything stems from knowing your users, including understanding their goals, skills, preferences, and tendencies. Once you know about your user, make sure to consider the following when designing your interface:

Keep the interface simple. The best interfaces are almost invisible to the user. They avoid unnecessary elements and are clear in the language they use on labels and in messaging.

Create consistency and use common UI elements. By using common elements in your UI, users feel more comfortable and are able to get things done more quickly. It is also important to create patterns in language, layout and design throughout the site to help facilitate efficiency. Once a user learns how to do something, they should be able to transfer that skill to other parts of the site.

Be purposeful in page layout. Consider the spatial relationships between items on the page and structure the page based on importance. Careful placement of items can help draw attention to the most important pieces of information and can aid scanning and readability.

Strategically use color and texture. You can direct attention toward or redirect attention away from items using color, light, contrast, and texture to your advantage.

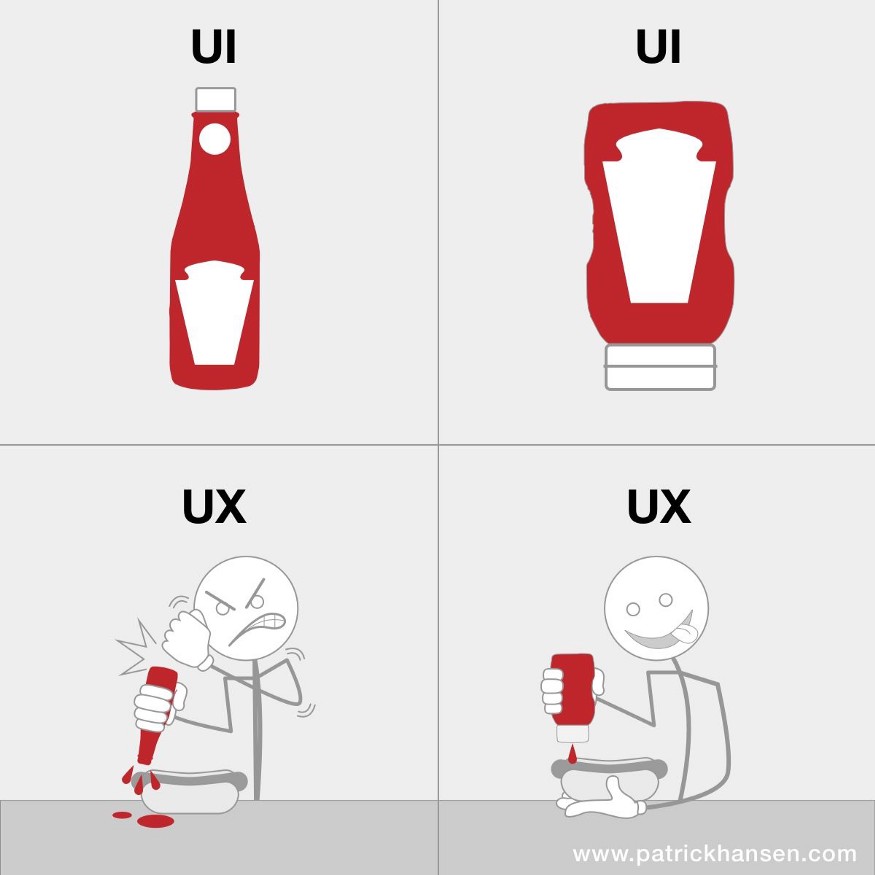
Use typography to create hierarchy and clarity. Carefully consider how you use typeface. Different sizes, fonts, and arrangement of the text to help increase scanability, legibility and readability.

Make sure that the system communicates what’s happening. Always inform your users of location, actions, changes in state, or errors. The use of various UI elements to communicate status and, if necessary, next steps can reduce frustration for your user.

Think about the defaults. By carefully thinking about and anticipating the goals people bring to your site, you can create defaults that reduce the burden on the user. This becomes particularly important when it comes to form design where you might have an opportunity to have some fields pre-chosen or filled out.

**Design issues**

Beautiful, attractive, and astonishing UI designs and implementations are not the only aspects of creating web applications. It’s vital to make sure your application is great to use as well (after all, UI and UX go hand-in-hand). Therefore, you must also think about accessibility and usability.



What matters in the applications we build is the ease of usability of the features created. Many UI problems found in applications defeats the ease of usability, thereby giving the applications poor UX. In this article, we’ll learn about some common UI problems and how to solve them, hence building better web applications. As the title suggests, the best way to fix these problems is through UI or front-end testing. We’ll go further into this later in the article.

# Common UI problems

Here are some UI problems you may find when you visit untested web applications.

**1. Inaccessible links and buttons**

Inaccessible here means difficult to click. You’ll often find links and buttons having small font sizes or very enclosed environments on many platforms. For the small sizes, they become difficult to click. The enclosed environments become difficult to click as a user can unintentionally click on something else.

**2. Poor color contrast**

In light and dark themes, you may find many color combinations that contrast terribly. This awful contrast makes the content hard to read and, in general, creates a bad user experience. Additionally, poor color schemes can make it hard for colorblind or color-sensitive users to use your platform.

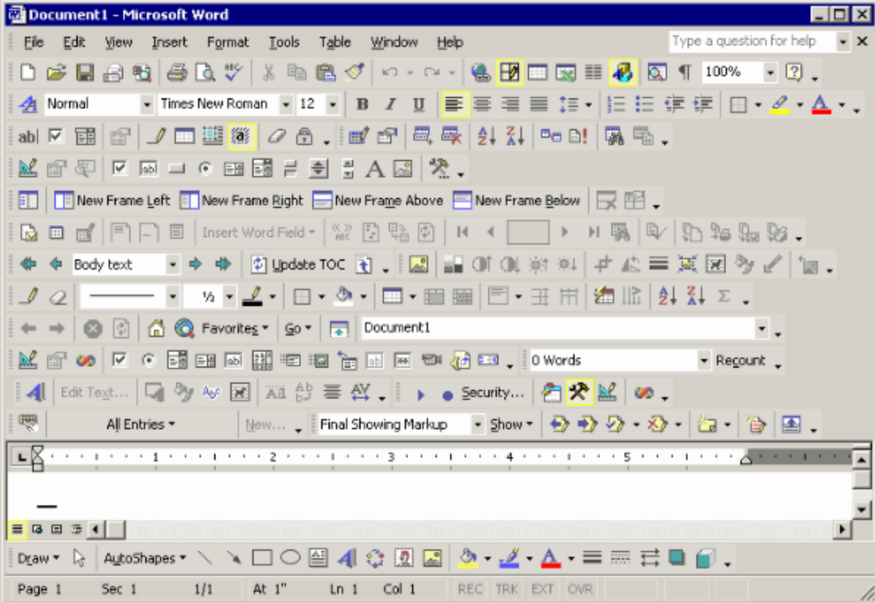
**3. Non-Responsive Design**

Users can visit applications using different devices, and sometimes, developers/designers fail to give attention to those other types of devices. That becomes a UI problem for a user viewing a site on a mobile phone, and they have to slide left and right to view the entire content of the website.

**4. Poor Typographical Hierarchy**

[Typographical Hierarchy](https://www.toptal.com/designers/typography/typographic-hierarchy) is a system that coordinates how each section of a page relates to another. Good use of this makes content easy to consume — for example, grouping content with headings and subheadings. However, poor use of this system makes content hard to consume, as users may find it hard knowing how sections are mapped to each other.

**5. Too many things in a place at a time**



See how crowded this interface is? When your application becomes cluttered with too many features, it becomes overwhelming for your users. This also affects the usability of your application, as a cluttered screen is a recipe for a bad user experience. It’s important to know what aspects of your platform should be prioritized over the other.While there are countless other [UI problems](https://www.google.com/url?sa=i&url=https%3A%2F%2Fcareerfoundry.com%2Fen%2Fblog%2Fui-design%2Fcommon-ui-design-mistakes%2F&psig=AOvVaw3UPN-vvjAtrE7GIvQkjf9f&ust=1610532532692000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCJjtoq-Tlu4CFQAAAAAdAAAAABAt), these 5 are the most common. Now that we’ve established the problems, let’s look at some solutions.

**1. Running End-to-End Tests**

Some of these problems can be hard to find. Some may not exist in previous versions of the applications but can occur in the updated ones. This encourages the need to run automated tests on applications regularly — especially after every update. These tests can be a group of small unit tests for different parts of the application or you can conduct [end to end testing](https://www.testcraft.io/end-to-end-testing/) to test the application’s flow from the beginning of a page to the end as an actual user would.

With these tests, you can confidently write your applications and get notified if there are problems. Tests like these help you discover unreachable elements in a page that you may have missed.

**2. Using Accessibility Testing Tools**

Some testing tools do more than ensuring all elements are reachable, clickable, and usable. Some may also offer accessibility insights on your buttons, links, and many other elements. However, there are specialized accessibility tools you can find online. You can find a lot of [browser extensions](https://www.digitala11y.com/accessibility-plug-ins-ie-chrome-firefox-browsers/), plugins, and even integrations to existing testing tools.

These tools already have default configurations on what to test for, and some support custom configurations too. The accessibility testing tools give insight on better color contrast, responsive designs, and sizes of elements as they affect accessibility.

**3. Proofreading and Editing Text**

Some tools may not do this out of the box for you. The order in which text is arranged on an application dramatically impacts readability. After writing content, the development team should have a thorough editing process, ensuring that each section is correctly mapped to the next and use correct headings, sub-headings, line breaks, and paragraphs. For a text-heavy application, just simple proofreading and editing will make a huge positive difference.

**4. Aim for Simplicity**

UI problems arise in designs when many things are happening at once — when there are many features in one place at a time. It can be challenging to place elements perfectly on a page, but arranging things becomes more comfortable when you aim for simplicity, thereby improving usability. If you’re building a web application, some hosting providers have their own page layouts that are already well-designed. So it may be prudent to adopt an existing design for your purposes to maintain simplicity and usability.

**Software Engineering | User Interface Design**

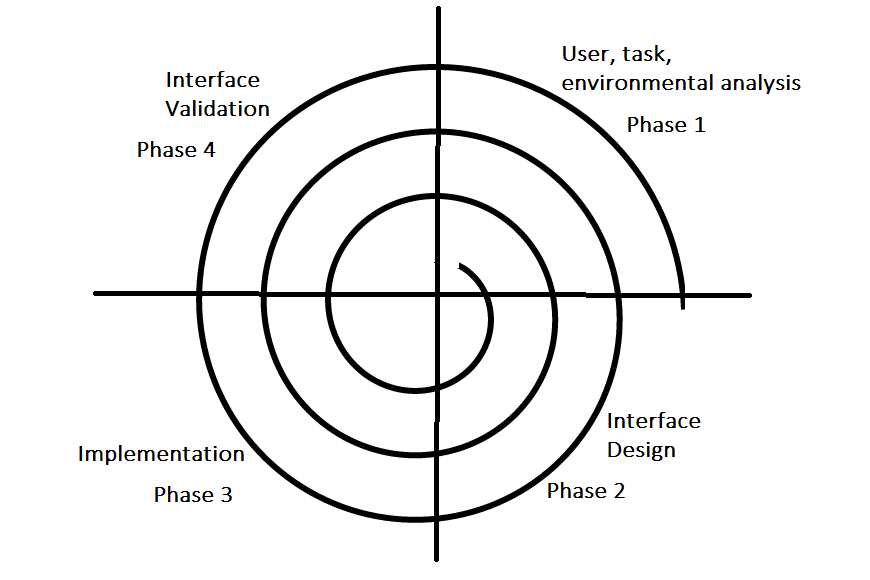
User interface is the front-end application view to which user interacts in order to use the software. The software becomes more popular if its user interface is:

* Attractive
* Simple to use
* Responsive in short time
* Clear to understand
* Consistent on all interface screens

There are two types of User Interface:

Command Line Interface: Command Line Interface provides a command prompt, where the user types the command and feeds to the system. The user needs to remember the syntax of the command and its use.

Graphical User Interface: Graphical User Interface provides the simple interactive interface to interact with the system. GUI can be a combination of both hardware and software. Using GUI, user interprets the software.



**User Interface Prototyping**

User interface (UI) prototyping is an iterative analysis technique in which users are actively involved in the mocking-up of the UI for a system. UI prototypes have several purposes:

As an analysis artifact that enables you to explore the problem space with your stakeholders.

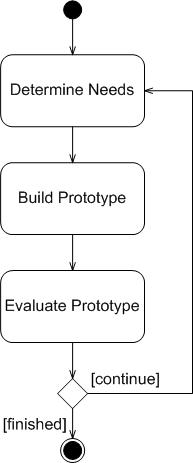
As a requirements artifact to initially envision the system.

As a design artifact that enables you to explore the solution space of your system.

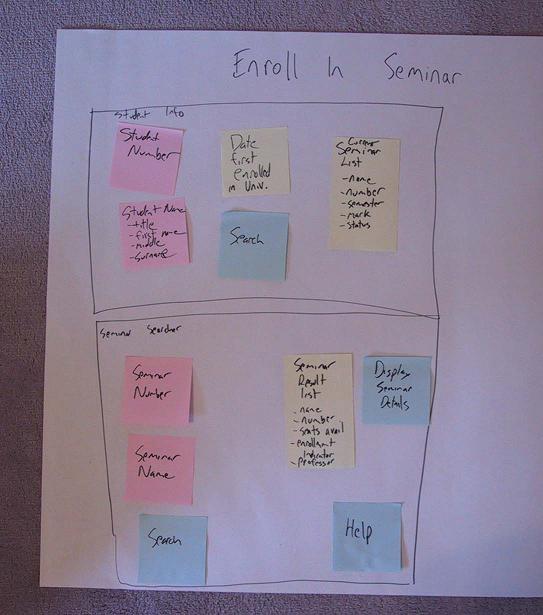
A vehicle for you to communicate the possible UI design(s) of your system.

A potential foundation from which to continue developing the system (if you intend to throw the prototype away and start over from scratch then you don't need to invest the time writing quality code for your prototype).

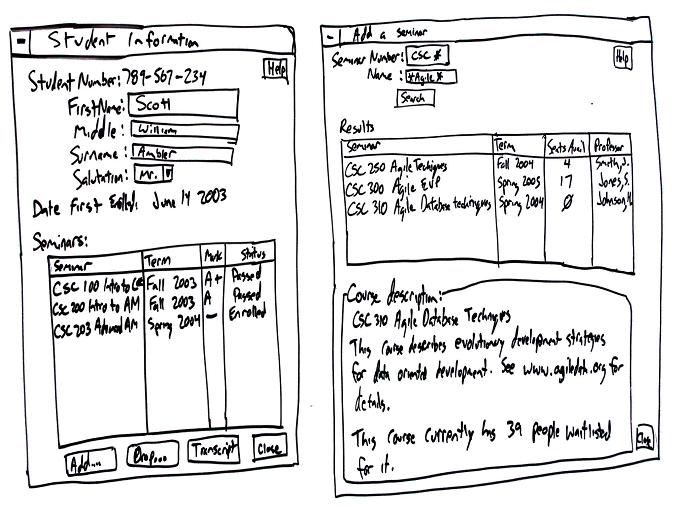
As you see in the activity diagram depicted in Figure 1 there are four high-level steps in the UI prototyping process. The first step is to analyze the user interface needs of your users. User interface modeling moves from requirements definition into analysis at the point you decide to evolve all or part of your essential user interface prototype into a traditional UI prototype. This implies you convert your hand-drawings, flip-chart paper, and sticky notes into something a little more substantial. You begin this process by making platform decisions which in effect is an architectural decision. For example, do you intend to deploy your system so it runs in an Internet browser, as an application with a windows-based graphical user interface (GUI), as a cross-platform Java application, or as a mainframe-based set of "green screens?" Different platforms lead to different prototyping tools, for a browser-based application, you need to use an HTML-development tool, whereas a Java-based application would require a Java development tool and a different approach to the user interface design.



While you're determining the needs of your stakeholders you may decide to transform your essential user interface prototypes, if you created them to begin with, with sketches. Figure 2 depicts an essential UI and Figure 3 a sketch of two potential screens or HTML pages based on that prototype. Transform really isn't the right word here seeing as I'm using a completely different modeling technology now (a whiteboard instead of paper) so in effect I'm replacing the essential UI prototype with the sketches.

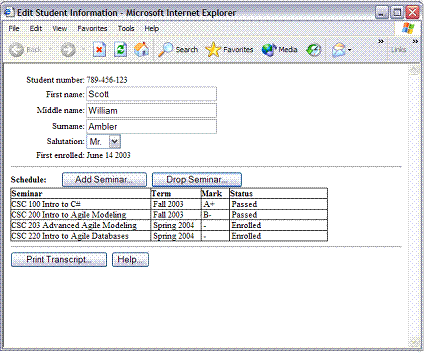
 `

I chose to split the prototype in two for cohesion issues - I prefer screens that fulfill a single purpose, in this case capturing basic student information and enrolling a student in seminars respectively. This is arguably a design issue (there's a fine line between analysis and design which you'll cross all the time). The sketches provide a final level of detail than the paper prototypes do, it is much easier to get a feel for how the screen will be implemented from than it is from [Figure 2](http://agilemodeling.com/artifacts/uiPrototype.htm#Figure2EssentialUI), although the sketch isn't as flexible because it's hard to shift widgets from one part of the diagram to another whereas with the paper it is very easy.



As you iterate through UI prototyping you will often discover information better captured in other artifacts. That's okay, you can follow the AM practice [Iterate to Another Artifact](http://agilemodeling.com/practices.htm#IterateToAnotherArtifact) and capture that information in the proper place. It also points to the importance of the AM practice [Create Several Models in Parallel](http://agilemodeling.com/practices.htm#CreateSeveralModelsInParallel) - you often need to work on several things at once to get the job done. Agile software development is an evolutionary process, so this is normal.

Once you understand the UI needs of your stakeholders the next step is to actually build a prototype. Using a prototyping tool or high-level language you develop the screens, pages, and reports needed by your users. With the user interface platform selected, you can begin converting individual aspects of your essential UI prototype into your traditional UI prototype. You may want to create sketches such as you see in [Figure 3](http://agilemodeling.com/artifacts/uiPrototype.htm#Figure3ScreenSketches) or go straight to a concrete implementation, such as the HTML page depicted in [Figure 4](http://agilemodeling.com/artifacts/uiPrototype.htm#Figure4HTMLPage). The sketches are more inclusive, your stakeholders can be actively involved in creating them, although the actual HTML page is much closer to working code (your primary goal).



It is critical to understand that you don't need to create a prototype for the entire system. It is very common to prototype a small portion of the user interface, perhaps a single screen or HTML page, before moving on to implementing it. Remember, agile developers work in an evolutionary manner - the don't need to define everything up front before moving on. Sometimes you will need to prototype a large portion of your system, perhaps as part of an envisioning exercise or perhaps even to help define the system scope so that you can obtain funding.

After a version of the UI prototype is built, it needs to be evaluated by your stakeholder's to verify that it meets their needs. Sometimes this is as easy as asking someone to spend a few minutes to look at what you've built and other times it's as complicated as scheduling a meeting so that you can demonstrate the software to a group of people. I prefer the first approach. When evaluating a UI prototype I've always found that the following questions provide me with significant feedback:

* What is good about the UI prototype?
* What is bad about the UI prototype?
* What is missing from the UI prototype?

After evaluating the prototype, you may find you need to scrap parts of it, modify parts, and even add brand-new parts. You want to stop the UI prototyping process when you find the evaluation process is no longer generating any new ideas or it is generating a small number of not-so-important ideas. Otherwise, back to exploring your stakeholder's UI needs.

**Software Engineering | Project Management**

What is Project?

A project is a group of tasks that need to complete to reach a clear result. A project also defines as a set of inputs and outputs which are required to achieve a goal. Projects can vary from simple to difficult and can be operated by one person or a hundred.

Projects usually described and approved by a project manager or team executive. They go beyond their expectations and objects, and it's up to the team to handle logistics and complete the project on time. For good project development, some teams split the project into specific tasks so they can manage responsibility and utilize team strengths.

What is software project management?

Software project management is an art and discipline of planning and supervising software projects. It is a sub-discipline of software project management in which software projects planned, implemented, monitored and controlled.

It is a procedure of managing, allocating and timing resources to develop computer software that fulfills requirements.

In software Project Management, the client and the developers need to know the length, period and cost of the project.

Prerequisite of software project management?

There are three needs for software project management. These are:

* Time
* Cost
* Quality

It is an essential part of the software organization to deliver a quality product, keeping the cost within the client?s budget and deliver the project as per schedule. There are various factors, both external and internal, which may impact this triple factor. Any of three-factor can severely affect the other two.

Project Manager

A project manager is a character who has the overall responsibility for the planning, design, execution, monitoring, controlling and closure of a project. A project manager represents an essential role in the achievement of the projects.

A project manager is a character who is responsible for giving decisions, both large and small projects. The project manager is used to manage the risk and minimize uncertainty. Every decision the project manager makes must directly profit their project.

Role of a Project Manager:

1. Leader

A project manager must lead his team and should provide them direction to make them understand what is expected from all of them.

2. Medium:

The Project manager is a medium between his clients and his team. He must coordinate and transfer all the appropriate information from the clients to his team and report to the senior management.

3. Mentor:

He should be there to guide his team at each step and make sure that the team has an attachment. He provides a recommendation to his team and points them in the right direction.

Responsibilities of a Project Manager:

Managing risks and issues.

Create the project team and assigns tasks to several team members.

Activity planning and sequencing.

Monitoring and reporting progress.

Modifies the project plan to deal with the situation.

**Management activities**

Activities

Software Project Management consists of many activities, that includes planning of the project, deciding the scope of product, estimation of cost in different terms, scheduling of tasks, etc.

**The list of activities are as follows:**

* Project planning and Tracking
* Project Resource Management
* Scope Management
* Estimation Management
* Project Risk Management
* Scheduling Management
* Project Communication Management
* Configuration Management

1. Project Planning: It is a set of multiple processes, or we can say that it a task that performed before the construction of the product starts.

2. Scope Management: It describes the scope of the project. Scope management is important because it clearly defines what would do and what would not. Scope Management create the project to contain restricted and quantitative tasks, which may merely be documented and successively avoids price and time overrun.

3. Estimation management: This is not only about cost estimation because whenever we start to develop software, but we also figure out their size(line of code), efforts, time as well as cost.

If we talk about the size, then Line of code depends upon user or software requirement.

If we talk about effort, we should know about the size of the software, because based on the size we can quickly estimate how big team required to produce the software.

If we talk about time, when size and efforts are estimated, the time required to develop the software can easily determine.

And if we talk about cost, it includes all the elements such as:

* Size of software
* Quality
* Hardware
* Communication
* Training
* Additional Software and tools
* Skilled manpower

4. Scheduling Management: Scheduling Management in software refers to all the activities to complete in the specified order and within time slotted to each activity. Project managers define multiple tasks and arrange them keeping various factors in mind.

For scheduling, it is compulsory -

* Find out multiple tasks and correlate them.
* Divide time into units.
* Assign the respective number of work-units for every job.
* Calculate the total time from start to finish.
* Break down the project into modules.

5. Project Resource Management: In software Development, all the elements are referred to as resources for the project. It can be a human resource, productive tools, and libraries.

Resource management includes:

* Create a project team and assign responsibilities to every team member
* Developing a resource plan is derived from the project plan.
* Adjustment of resources.

6. Project Risk Management: Risk management consists of all the activities like identification, analyzing and preparing the plan for predictable and unpredictable risk in the project.

Several points show the risks in the project:

* The Experienced team leaves the project, and the new team joins it.
* Changes in requirement.
* Change in technologies and the environment.
* Market competition.

7. Project Communication Management: Communication is an essential factor in the success of the project. It is a bridge between client, organization, team members and as well as other stakeholders of the project such as hardware suppliers.

AD

From the planning to closure, communication plays a vital role. In all the phases, communication must be clear and understood. Miscommunication can create a big blunder in the project.

8. Project Configuration Management: Configuration management is about to control the changes in software like requirements, design, and development of the product.

The Primary goal is to increase productivity with fewer errors.

Some reasons show the need for configuration management:

* Several people work on software that is continually update.
* Help to build coordination among suppliers.
* Changes in requirement, budget, schedule need to accommodate.
* Software should run on multiple systems.

Tasks perform in Configuration management:

AD

* Identification
* Baseline
* Change Control
* Configuration Status Accounting
* Configuration Audits and Reviews

People involved in Configuration Management:

**Project Planning**

Software Project Planning

A Software Project is the complete methodology of programming advancement from requirement gathering to testing and support, completed by the execution procedures, in a specified period to achieve intended software product.

Need of Software Project Management

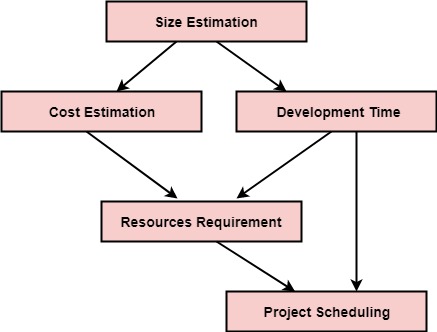
Software development is a sort of all new streams in world business, and there's next to no involvement in structure programming items. Most programming items are customized to accommodate customer's necessities. The most significant is that the underlying technology changes and advances so generally and rapidly that experience of one element may not be connected to the other one. All such business and ecological imperatives bring risk in software development; hence, it is fundamental to manage software projects efficiently.

Software Project Manager

Software manager is responsible for planning and scheduling project development. They manage the work to ensure that it is completed to the required standard. They monitor the progress to check that the event is on time and within budget. The project planning must incorporate the major issues like size & cost estimation scheduling, project monitoring, personnel selection evaluation & risk management. To plan a successful software project, we must understand:

* Scope of work to be completed
* Risk analysis
* The resources mandatory
* The project to be accomplished
* Record of being followed

Software Project planning starts before technical work start. The various steps of planning activities are:



The size is the crucial parameter for the estimation of other activities. Resources requirement are required based on cost and development time. Project schedule may prove to be very useful for controlling and monitoring the progress of the project. This is dependent on resources & development time.

**Risk management**

What is Risk?

"Tomorrow problems are today's risk." Hence, a clear definition of a "risk" is a problem that could cause some loss or threaten the progress of the project, but which has not happened yet.

These potential issues might harm cost, schedule or technical success of the project and the quality of our software device, or project team morale.

Risk Management is the system of identifying addressing and eliminating these problems before they can damage the project.

We need to differentiate risks, as potential issues, from the current problems of the project.

Different methods are required to address these two kinds of issues.

For example, staff storage, because we have not been able to select people with the right technical skills is a current problem, but the threat of our technical persons being hired away by the competition is a risk.

Risk Management

A software project can be concerned with a large variety of risks. In order to be adept to systematically identify the significant risks which might affect a software project, it is essential to classify risks into different classes. The project manager can then check which risks from each class are relevant to the project.

There are three main classifications of risks which can affect a software project:

1. Project risks
2. Technical risks
3. Business risks

**1. Project risks:** Project risks concern differ forms of budgetary, schedule, personnel, resource, and customer-related problems. A vital project risk is schedule slippage. Since the software is intangible, it is very tough to monitor and control a software project. It is very tough to control something which cannot be identified. For any manufacturing program, such as the manufacturing of cars, the plan executive can recognize the product taking shape.

**2. Technical risks:** Technical risks concern potential method, implementation, interfacing, testing, and maintenance issue. It also consists of an ambiguous specification, incomplete specification, changing specification, technical uncertainty, and technical obsolescence. Most technical risks appear due to the development team's insufficient knowledge about the project.

**3. Business risks:** This type of risks contain risks of building an excellent product that no one need, losing budgetary or personnel commitments, etc.

**Other risk categories**

1. **1. Known risks:** Those risks that can be uncovered after careful assessment of the project program, the business and technical environment in which the plan is being developed, and more reliable data sources (e.g., unrealistic delivery date)
2. **2. Predictable risks:** Those risks that are hypothesized from previous project experience (e.g., past turnover)
3. **3. Unpredictable risks:** Those risks that can and do occur, but are extremely tough to identify in advance.

Principle of Risk Management

1. **Global Perspective:** In this, we review the bigger system description, design, and implementation. We look at the chance and the impact the risk is going to have.
2. **Take a forward-looking view:** Consider the threat which may appear in the future and create future plans for directing the next events.
3. **Open Communication:** This is to allow the free flow of communications between the client and the team members so that they have certainty about the risks.
4. **Integrated management:** In this method risk management is made an integral part of project management.
5. **Continuous process:** In this phase, the risks are tracked continuously throughout the risk management paradigm.

**Software Engineering | Quality Management**

Software Quality

Software quality product is defined in term of its fitness of purpose. That is, a quality product does precisely what the users want it to do. For software products, the fitness of use is generally explained in terms of satisfaction of the requirements laid down in the SRS document. Although "fitness of purpose" is a satisfactory interpretation of quality for many devices such as a car, a table fan, a grinding machine, etc.for software products, "fitness of purpose" is not a wholly satisfactory definition of quality.

Example: Consider a functionally correct software product. That is, it performs all tasks as specified in the SRS document. But, has an almost unusable user interface. Even though it may be functionally right, we cannot consider it to be a quality product.

The modern view of a quality associated with a software product several quality methods such as the following:

Portability: A software device is said to be portable, if it can be freely made to work in various operating system environments, in multiple machines, with other software products, etc.

Usability: A software product has better usability if various categories of users can easily invoke the functions of the product.

Reusability: A software product has excellent reusability if different modules of the product can quickly be reused to develop new products.

Correctness: A software product is correct if various requirements as specified in the SRS document have been correctly implemented.

Maintainability: A software product is maintainable if bugs can be easily corrected as and when they show up, new tasks can be easily added to the product, and the functionalities of the product can be easily modified, etc.

Software Quality Management System

A quality management system is the principal methods used by organizations to provide that the products they develop have the desired quality.

A quality system subsists of the following:

Managerial Structure and Individual Responsibilities: A quality system is the responsibility of the organization as a whole. However, every organization has a sever quality department to perform various quality system activities. The quality system of an arrangement should have the support of the top management. Without help for the quality system at a high level in a company, some members of staff will take the quality system seriously.

Quality System Activities: The quality system activities encompass the following:

Auditing of projects

Review of the quality system

Development of standards, methods, and guidelines, etc.

Production of documents for the top management summarizing the effectiveness of the quality system in the organization.

Evolution of Quality Management System

Quality systems have increasingly evolved over the last five decades. Before World War II, the usual function to produce quality products was to inspect the finished products to remove defective devices. Since that time, quality systems of organizations have undergone through four steps of evolution, as shown in the fig. The first product inspection task gave method to quality control (QC).

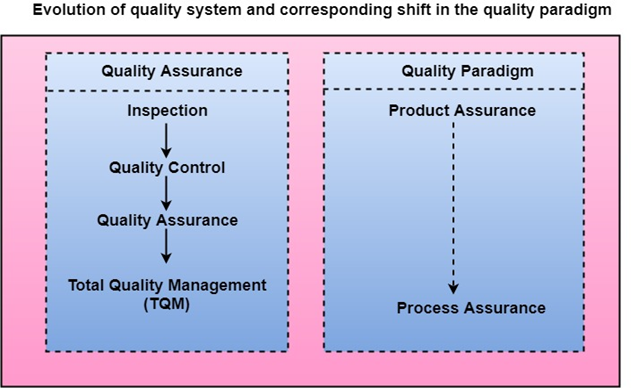
AD

Quality control target not only on detecting the defective devices and removes them but also on determining the causes behind the defects. Thus, quality control aims at correcting the reasons for bugs and not just rejecting the products. The next breakthrough in quality methods was the development of quality assurance methods.

The primary premise of modern quality assurance is that if an organization's processes are proper and are followed rigorously, then the products are obligated to be of good quality. The new quality functions include guidance for recognizing, defining, analyzing, and improving the production process.

Total quality management (TQM) advocates that the procedure followed by an organization must be continuously improved through process measurements. TQM goes stages further than quality assurance and aims at frequently process improvement. TQM goes beyond documenting steps to optimizing them through a redesign. A term linked to TQM is Business Process Reengineering (BPR).

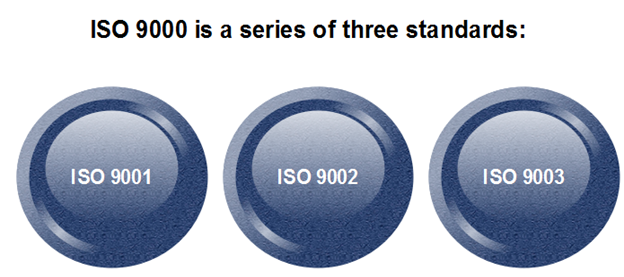
BPR aims at reengineering the method business is carried out in an organization. From the above conversation, it can be stated that over the years, the quality paradigm has changed from product assurance to process assurance, as shown in fig.



**ISO 9000 Certification**

ISO (International Standards Organization) is a group or consortium of 63 countries established to plan and fosters standardization. ISO declared its 9000 series of standards in 1987. It serves as a reference for the contract between independent parties. The ISO 9000 standard determines the guidelines for maintaining a quality system. The ISO standard mainly addresses operational methods and organizational methods such as responsibilities, reporting, etc. ISO 9000 defines a set of guidelines for the production process and is not directly concerned about the product itself.

Types of ISO 9000 Quality Standards



The ISO 9000 series of standards is based on the assumption that if a proper stage is followed for production, then good quality products are bound to follow automatically. The types of industries to which the various ISO standards apply are as follows.

1. ISO 9001: This standard applies to the organizations engaged in design, development, production, and servicing of goods. This is the standard that applies to most software development organizations.
2. ISO 9002: This standard applies to those organizations which do not design products but are only involved in the production. Examples of these category industries contain steel and car manufacturing industries that buy the product and plants designs from external sources and are engaged in only manufacturing those products. Therefore, ISO 9002 does not apply to software development organizations.
3. ISO 9003: This standard applies to organizations that are involved only in the installation and testing of the products. For example, Gas companies.

How to get ISO 9000 Certification?

An organization determines to obtain ISO 9000 certification applies to ISO registrar office for registration. The process consists of the following stages:



1. Application: Once an organization decided to go for ISO certification, it applies to the registrar for registration.
2. Pre-Assessment: During this stage, the registrar makes a rough assessment of the organization.
3. Document review and Adequacy of Audit: During this stage, the registrar reviews the document submitted by the organization and suggest an improvement.
4. Compliance Audit: During this stage, the registrar checks whether the organization has compiled the suggestion made by it during the review or not.
5. Registration: The Registrar awards the ISO certification after the successful completion of all the phases.
6. Continued Inspection: The registrar continued to monitor the organization time by time.

Software Engineering Institute Capability Maturity Model (SEICMM)

The Capability Maturity Model (CMM) is a procedure used to develop and refine an organization's software development process.

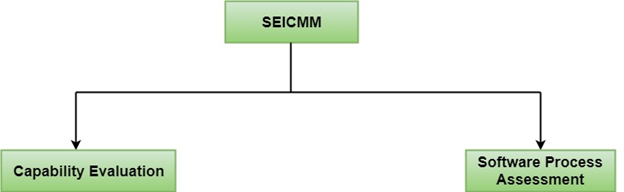
The model defines a five-level evolutionary stage of increasingly organized and consistently more mature processes.

CMM was developed and is promoted by the Software Engineering Institute (SEI), a research and development center promote by the U.S. Department of Defense (DOD).

Capability Maturity Model is used as a benchmark to measure the maturity of an organization's software process.

Methods of SEICMM

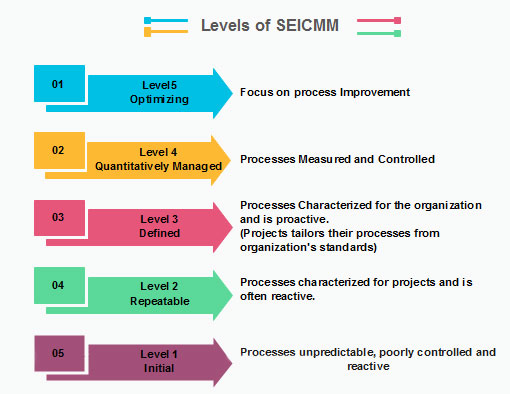
There are two methods of SEICMM:



**Capability Evaluation:** Capability evaluation provides a way to assess the software process capability of an organization. The results of capability evaluation indicate the likely contractor performance if the contractor is awarded a work. Therefore, the results of the software process capability assessment can be used to select a contractor.

**Software Process Assessment:** Software process assessment is used by an organization to improve its process capability. Thus, this type of evaluation is for purely internal use.

SEI CMM categorized software development industries into the following five maturity levels. The various levels of SEI CMM have been designed so that it is easy for an organization to build its quality system starting from scratch slowly.



Level 1: Initial

Ad hoc activities characterize a software development organization at this level. Very few or no processes are described and followed. Since software production processes are not limited, different engineers follow their process and as a result, development efforts become chaotic. Therefore, it is also called a chaotic level.

Level 2: Repeatable

At this level, the fundamental project management practices like tracking cost and schedule are established. Size and cost estimation methods, like function point analysis, COCOMO, etc. are used.

Level 3: Defined

At this level, the methods for both management and development activities are defined and documented. There is a common organization-wide understanding of operations, roles, and responsibilities. The ways through defined, the process and product qualities are not measured. ISO 9000 goals at achieving this level.

Level 4: Managed

At this level, the focus is on software metrics. Two kinds of metrics are composed.

**Product metrics** measure the features of the product being developed, such as its size, reliability, time complexity, understandability, etc.

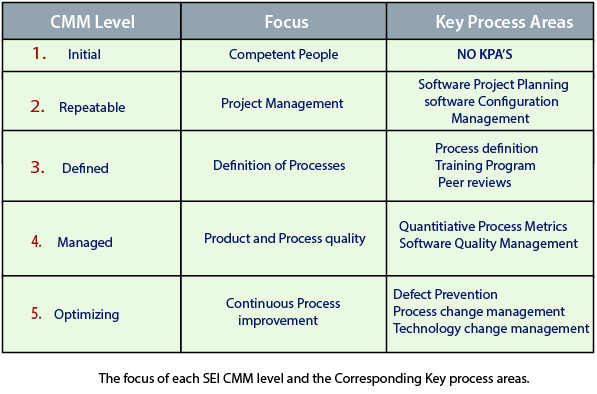
**Process metrics** follow the effectiveness of the process being used, such as average defect correction time, productivity, the average number of defects found per hour inspection, the average number of failures detected during testing per LOC, etc. The software process and product quality are measured, and quantitative quality requirements for the product are met. Various tools like Pareto charts, fishbone diagrams, etc. are used to measure the product and process quality. The process metrics are used to analyze if a project performed satisfactorily. Thus, the outcome of process measurements is used to calculate project performance rather than improve the process.

Level 5: Optimizing

At this phase, process and product metrics are collected. Process and product measurement data are evaluated for continuous process improvement.

Key Process Areas (KPA) of a software organization

Except for SEI CMM level 1, each maturity level is featured by several Key Process Areas (KPAs) that contains the areas an organization should focus on improving its software process to the next level. The focus of each level and the corresponding key process areas are shown in the fig.



SEI CMM provides a series of key areas on which to focus to take an organization from one level of maturity to the next. Thus, it provides a method for gradual quality improvement over various stages. Each step has been carefully designed such that one step enhances the capability already built up.

People Capability Maturity Model (PCMM)

PCMM is a maturity structure that focuses on continuously improving the management and development of the human assets of an organization.

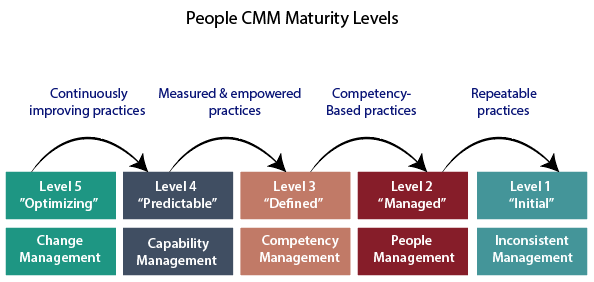
It defines an evolutionary improvement path from Adhoc, inconsistently performed practices, to a mature, disciplined, and continuously improving the development of the knowledge, skills, and motivation of the workforce that enhances strategic business performance.

The People Capability Maturity Model (PCMM) is a framework that helps the organization successfully address their critical people issues. Based on the best current study in fields such as human resources, knowledge management, and organizational development, the PCMM guides organizations in improving their steps for managing and developing their workforces.

The People CMM defines an evolutionary improvement path from Adhoc, inconsistently performed workforce practices, to a mature infrastructure of practices for continuously elevating workforce capability.

The PCMM subsists of five maturity levels that lay successive foundations for continuously improving talent, developing effective methods, and successfully directing the people assets of the organization. Each maturity level is a well-defined evolutionary plateau that institutionalizes a level of capability for developing the talent within the organization

**The five steps of the People CMM framework are:**



Initial Level: Maturity Level 1

The Initial Level of maturity includes no process areas. Although workforce practices implement in Maturity Level, 1 organization tend to be inconsistent or ritualistic, virtually all of these organizations perform processes that are defined in the Maturity Level 2 process areas.

Managed Level: Maturity Level 2

To achieve the Managed Level, Maturity Level 2, managers starts to perform necessary people management practices such as staffing, operating performance, and adjusting compensation as a repeatable management discipline. The organization establishes a culture focused at the unit level for ensuring that person can meet their work commitments. In achieving Maturity Level 2, the organization develops the capability to handle skills and performance at the unit level. The process areas at Maturity Level 2 are Staffing, Communication and Coordination, Work Environment, Performance Management, Training and Development, and Compensation.

Defined Level: Maturity Level 3

The fundamental objective of the defined level is to help an organization gain a competitive benefit from developing the different competencies that must be combined in its workforce to accomplish its business activities. These workforce competencies represent critical pillars supporting the strategic workforce competencies to current and future business objectives; the improved workforce practices for implemented at Maturity Level 3 become crucial enablers of business strategy.

Predictable Level: Maturity Level 4

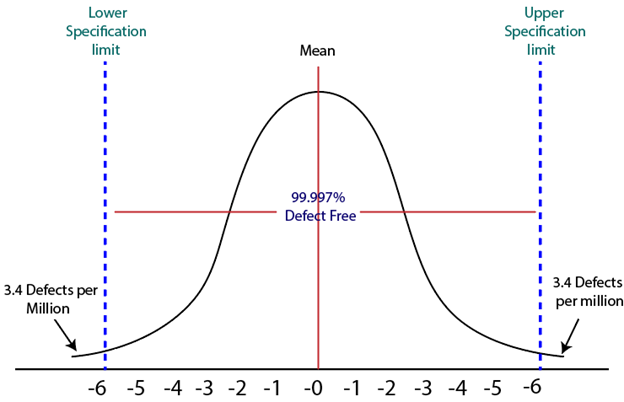
At the Predictable Level, the organization handles and exploits the capability developed by its framework of workforce competencies. The organization is now able to handle its capacity and performance quantitatively. The organization can predict its capability for performing work because it can quantify the ability of its workforce and of the competency-based methods they use performing in their assignments.

Optimizing Level: Maturity Level 5

At the Optimizing Level, the integrated organization is focused on continual improvement. These improvements are made to the efficiency of individuals and workgroups, to the act of competency-based processes, and workforce practices and activities.

Six Sigma

Six Sigma is the process of improving the quality of the output by identifying and eliminating the cause of defects and reduce variability in manufacturing and business processes. The maturity of a manufacturing process can be defined by a sigma rating indicating its percentage of defect-free products it creates. A six sigma method is one in which **99.99966%** of all the opportunities to produce some features of a component are statistically expected to be free of defects (**3.4 defective features per million opportunities**).



History of Six Sigma

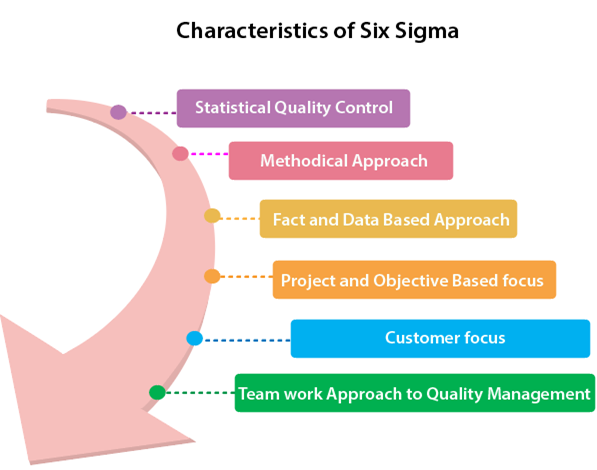
Six-Sigma is a set of methods and tools for process improvement. It was introduced by Engineer **Sir Bill Smith** while working at **Motorola** in 1986. In the 1980s, **Motorola** was developing Quasar televisions which were famous, but the time there was lots of defects which came up on that due to picture quality and sound variations.

By using the same raw material, machinery and workforce a Japanese form took over Quasar television production, and within a few months, they produce Quasar TV's sets which have fewer errors. This was obtained by improving management techniques.

Six Sigma was adopted by Bob Galvin, the CEO of Motorola in 1986 and registered as a Motorola Trademark on December 28, 1993, then it became a quality leader.

Characteristics of Six Sigma

The Characteristics of Six Sigma are as follows:

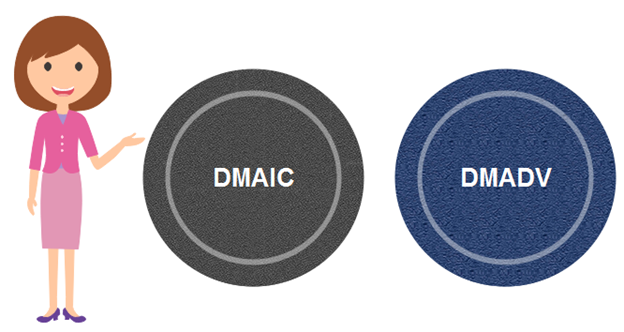


1. **Statistical Quality Control:** Six Sigma is derived from the Greek Letter σ (Sigma) from the Greek alphabet, which is used to denote Standard Deviation in statistics. Standard Deviation is used to measure variance, which is an essential tool for measuring non-conformance as far as the quality of output is concerned.
2. **Methodical Approach:** The Six Sigma is not a merely quality improvement strategy in theory, as it features a well defined systematic approach of application in DMAIC and DMADV which can be used to improve the quality of production. DMAIC is an acronym for Design-Measure- Analyze-Improve-Control. The alternative method DMADV stands for Design-Measure- Analyze-Design-Verify.
3. **Fact and Data-Based Approach:** The statistical and methodical aspect of Six Sigma shows the scientific basis of the technique. This accentuates essential elements of the Six Sigma that is a fact and data-based.
4. **Project and Objective-Based Focus:** The Six Sigma process is implemented for an organization's project tailored to its specification and requirements. The process is flexed to suits the requirements and conditions in which the projects are operating to get the best results.
5. **Customer Focus:** The customer focus is fundamental to the Six Sigma approach. The quality improvement and control standards are based on specific customer requirements.
6. **Teamwork Approach to Quality Management:** The Six Sigma process requires organizations to get organized when it comes to controlling and improving quality. Six Sigma involving a lot of training depending on the role of an individual in the Quality Management team.

Six Sigma Methodologies

Six Sigma projects follow two project methodologies:

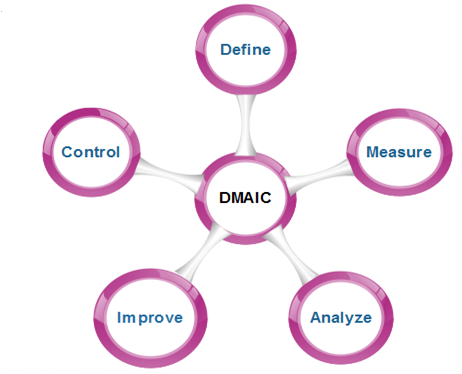
1. DMAIC
2. DMADV



DMAIC

It specifies a data-driven quality strategy for improving processes. This methodology is used to enhance an existing business process.

**The DMAIC project methodology has five phases:**

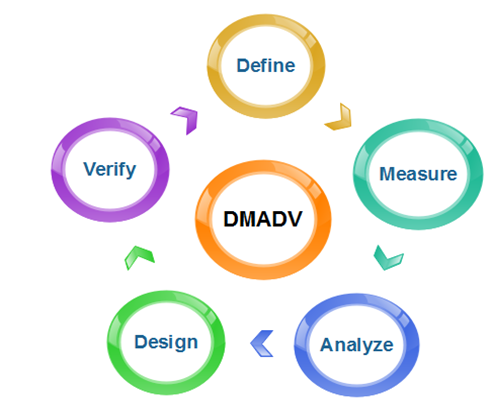


1. **Define:** It covers the process mapping and flow-charting, project charter development, problem-solving tools, and so-called 7-M tools.
2. **Measure:** It includes the principles of measurement, continuous and discrete data, and scales of measurement, an overview of the principle of variations and repeatability and reproducibility (RR) studies for continuous and discrete data.
3. **Analyze:** It covers establishing a process baseline, how to determine process improvement goals, knowledge discovery, including descriptive and exploratory data analysis and data mining tools, the basic principle of Statistical Process Control (SPC), specialized control charts, process capability analysis, correlation and regression analysis, analysis of categorical data, and non-parametric statistical methods.
4. **Improve:** It covers project management, risk assessment, process simulation, and design of experiments (DOE), robust design concepts, and process optimization.
5. **Control:** It covers process control planning, using SPC for operational control and PRE-Control.

DMADV

It specifies a data-driven quality strategy for designing products and processes. This method is used to create new product designs or process designs in such a way that it results in a more predictable, mature, and detect free performance.

**The DMADV project methodology has five phases:**



1. **Define:** It defines the problem or project goal that needs to be addressed.
2. **Measure:** It measures and determines the customer's needs and specifications.
3. **Analyze:** It analyzes the process to meet customer needs.
4. **Design:** It can design a process that will meet customer needs.
5. **Verify:** It can verify the design performance and ability to meet customer needs.