

PROJECT MANAGEMENT



PROJECT MANAGEMENT

- Project management involves the planning, monitoring, and control of the people, process, and events that occur as software evolves from a preliminary concept to full operational deployment.

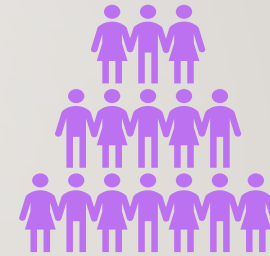
THE MANAGEMENT SPECTRUM

- 4 P's
 - People
 - Product
 - Process
 - Project

PEOPLE



People Capability Maturity Model (People-CMM)



The Stakeholders

Senior managers

Project (technical) managers

Practitioners

Customers

End users

-
- Team Leader
 - The Software Team depends on following factors:
 - Difficulty of the problem to be solved
 - Size
 - Team lifetime
 - Degree to which the problem can be modularized
 - Quality and reliability
 - Rigidity of the delivery date
 - Communication required for the project

ORGANIZATIONAL PARADIGMS



Closed
paradigm

Random
paradigm

Open
paradigm

Synchronous
paradigm

PRODUCT

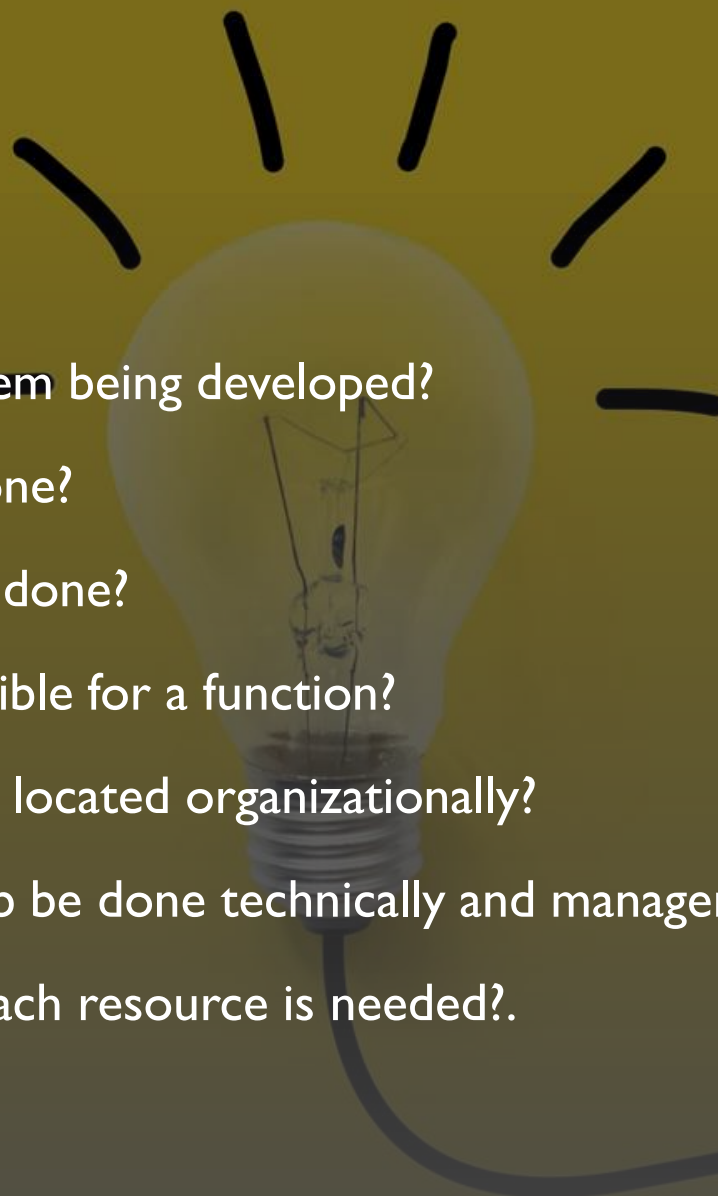
- Product objectives and scope should be established
- Software Scope:
 - **Context.**
 - **Information objectives**
 - **Function and performance**
- Problem Decomposition

PROJECT



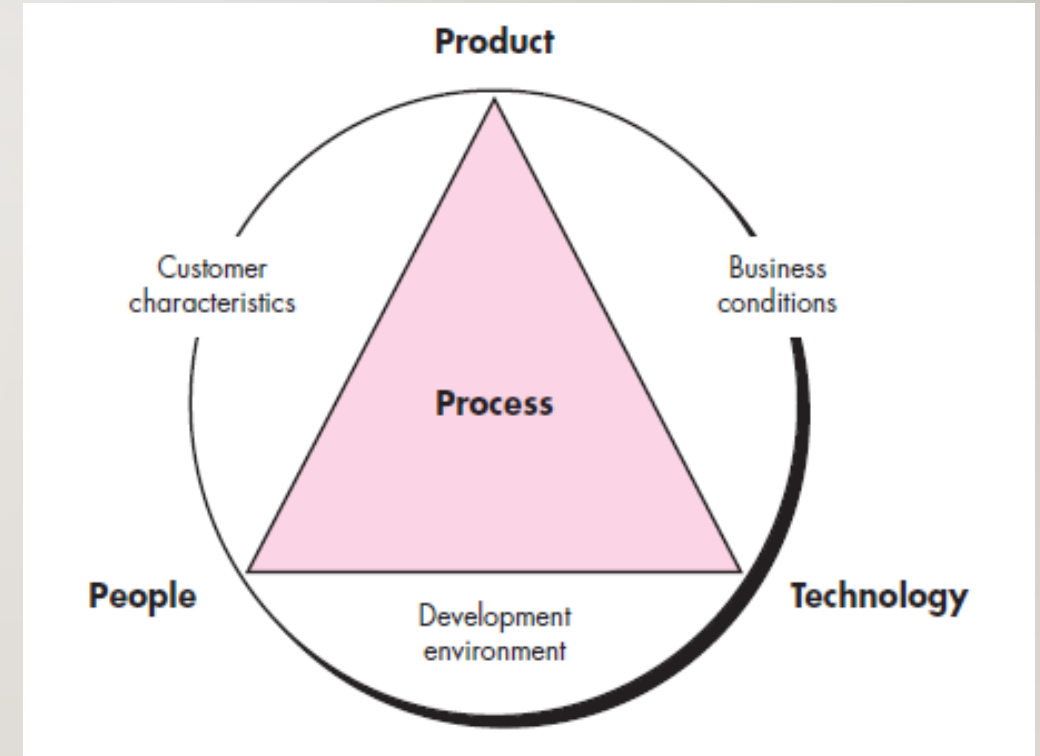
- Approach for planning, monitoring, and controlling the project.
- Five-part approach to software projects:
 - Start on the right foot
 - Maintain momentum.
 - Track progress
 - Make smart decisions
 - Conduct a postmortem analysis

THE W5HH PRINCIPLE

- 
- Why is the system being developed?
 - What will be done?
 - When will it be done?
 - Who is responsible for a function?
 - Where are they located organizationally?
 - How will the job be done technically and managerially?
 - How much of each resource is needed?.

PROCESS METRICS

- *Process metrics* are collected across all projects and over long periods of time.
- lead to long-term software process improvement.
- **Private Metrics**
- **Public metrics**



PROJECT METRICS

- Metrics collected from past projects are used as a basis from which effort and time estimates are made for current software work
- To minimize the development schedule by making the adjustments necessary to avoid delays and, mitigate potential problems and risks.
- project metrics are used to assess product quality on an ongoing basis and, when necessary, modify the technical approach to improve quality.

MEASUREMENT

- Direct Measure
 - Cost
 - Effort
- Indirect Measure
 - Functionality,
 - Quality
 - Complexity,
 - Efficiency, Reliability, Maintainability etc.

SOFTWARE MEASUREMENT

- process, project, and product metrics
- Product metrics that are private to an individual are often combined to develop project metrics that are public to a software team.
- Project metrics are then consolidated to create process metrics that are public to the software organization as a whole.

PRODUCT METRICS

- METRICS FOR THE REQUIREMENTS MODEL
 - Function based metrics
 - Metrics for Specification Quality
- METRICS FOR THE DESIGN MODEL
 - Architectural Design Metrics
- METRICS FOR SOURCE CODE-Halstead's theory
- METRICS FOR TESTING
- METRICS FOR MAINTENANCE

METRICS FOR THE REQUIREMENTS MODEL

FUNCTION BASED METRICS

- Predicting the “size” of the resultant system
- A means for measuring the functionality delivered by a system.
- Information domain values are defined as:
 - **Number of external inputs (Eis)**- originates from a user or is transmitted from another application and provides distinct application-oriented data or control information.
 - **Number of external outputs (EOs)**. Each *external output* is derived data within the application that provides information to the user. In this context external output refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.
 - **Number of external inquiries (EQs)**. An *external inquiry* is defined as an online input that results in the generation of some immediate software response in the form of an online output (often retrieved from an ILF)
 - **Number of internal logical files (ILFs)**. Each *internal logical file* is a logical grouping of data that resides within the application’s boundary and is maintained via external inputs.
 - **Number of external interface files (EIFs)**. Each *external interface file* is a logical grouping of data that resides external to the application but provides information that may be of use to the application.

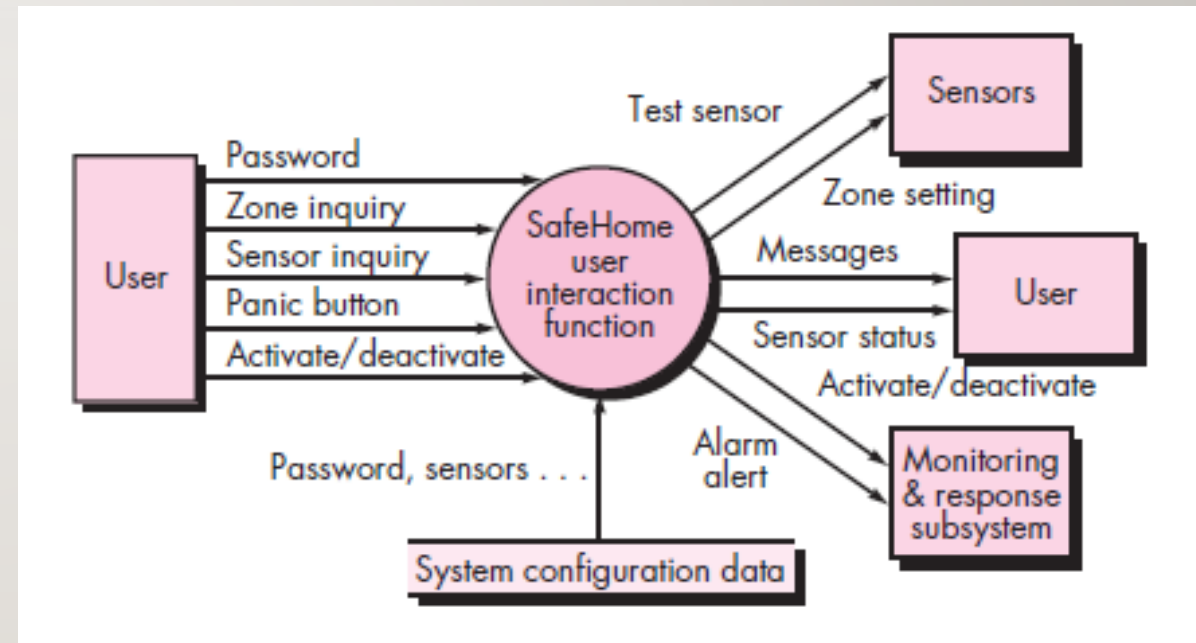
- $$FP = \text{count total} * [0.65 + 0.01 * \sum F_i]$$

$$FP = \text{count total} * [0.65 + 0.01 * \sum F_i]$$

- On a scale of ranges from 0 (not important or applicable) to 5 (absolutely essential

Information Domain Value	Count		Weighting factor			
			Simple	Average	Complex	
External Inputs (EIs)		×	3	4	6	=
External Outputs (EOs)		×	4	5	7	=
External Inquiries (EQs)		×	3	4	6	=
Internal Logical Files (ILFs)		×	7	10	15	=
External Interface Files (EIFs)		×	5	7	10	=
Count total						

Information Domain Value	Count		Weighting factor				
			Simple	Average	Complex		
External Inputs (EIs)	3	×	3	4	6	=	9
External Outputs (EOs)	2	×	4	5	7	=	8
External Inquiries (EQs)	2	×	3	4	6	=	6
Internal Logical Files (ILFs)	1	×	7	10	15	=	7
External Interface Files (EIFs)	4	×	5	7	10	=	20
Count total							50



$$FP = 50 * [0.65 + (0.01 * 46)] = 56$$

	average
User Input = 50	4
User Output = 40	5
User Inquiries = 35	4
User Files = 6	10
External Interface = 4	7

F=42

- Function point=????

	average
User Input = 50	4
User Output = 40	5
User Inquiries = 35	4
User Files = 6	10
External Interface = 4	7

F=42

- Function point= 671.96

METRICS FOR SPECIFICATION QUALITY

$$n_r = n_f + n_{nf}$$

- *Specificity*

$$Q_1 = n_{UI} / n_n$$

- *Completeness*

$$Q_2 = n_c / (n_c + n_{nv})$$

METRICS FOR THE DESIGN MODEL

- **Structural complexity** of a module i is defined in the following manner:

$$S_i = f_{out}^2(i)$$

- **Data complexity** provides an indication of the complexity in the internal interface for a module i and is defined as:

$$D_i = v(i)/(f_{out}(i)+1)$$

- **System complexity** is defined as the sum of structural and data complexity, specified as

$$C(i) = S(i) + D(i)$$

METRICS FOR SOURCE CODE- HALSTEAD'S THEORY

n_1 = number of distinct operators that appear in a program

n_2 = number of distinct operands that appear in a program

N_1 = total number of operator occurrences

N_2 = total number of operand occurrences

- **Length**

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

- **Volume**

$$V = N \log_2 (n_1 + n_2)$$

- **Level**

$$L = \frac{2}{n_1} * \frac{n_2}{N_2}$$

- **Difficulty**

$$D = 1/L$$

- **Effort**

$$E = V * D$$

```
voidsort( int*a, intn )
{ int i, j, t;
    if( n <2 ) return;
    for( i=0 ; i <n-1; i++)
    { for( j=i+1 ; j <n ; j++)
        {if( a[i] > a[j])
            { t=a[i];
              a[i] =a[j];
              a[j]=t;
            }
        }
    }
}
```

```
voidsort( int*a, intn )
```

```
{ int i, j, t;
```

```
    if( n <2 ) return;
```

```
    for( i=0 ; i <n-1; i++)
```

```
    { for( j=i+1 ; j <n ; j++)
```

```
        {if( a[i] > a[j])
```

```
            { t=a[i];
```

```
              a[i] =a[j];
```

```
              a[j]=t;
```

```
        } } }
```

Operators N1 = 50

Operands N2 = 30

n1 = 17

n2 = 7

V=392

D=36