Solution to the problems of software project development:

1. Start on the right foot
2. Maintain momentum
3. Track progress
4. Make smart decisions
5. Conduct a postmortem analysis

The 90–90 rule

- The first **90** percent of a system absorbs **90** percent of the allotted effort and time.
- The last **10** percent takes another **90** percent of the allotted effort and time.

W5HH Principle (to define key project characteristics)

WHY	Why is the system being developed?	All stakeholders should assess the validity of business reasons for the software work. Does the business purpose justify the expenditure of people, time, and money?
WHAT	What will be done?	The task set required for the project is defined.
WHEN	When will it be done?	The team establishes a project schedule by identifying when project tasks are to be conducted and when milestones are to be reached.
WHO	Who is responsible for a function?	The role and responsibility of each member of the software team is defined.
WHERE	Where are they located organizationally?	Not all roles and responsibilities reside within software practitioners. The customer, users, and other stakeholders also have responsibilities.
HOW	How will the job be done technically and managerially?	Once product scope is established, a management and technical strategy for the project must be defined.
HOW MUCH:	How much of each resource is needed?	The answer to this question is derived by developing estimates based on answers to earlier questions.

- Boehm's W5HH Principle is **applicable regardless of the size or complexity** of a software project.
- The questions noted provide **an excellent planning outline**.

Project planning is undertaken and completed before any development activity starts.

Commitment to unrealistic time and resource estimates → schedule slippage→customer dissatisfaction→adversely affect team morale→project failure.

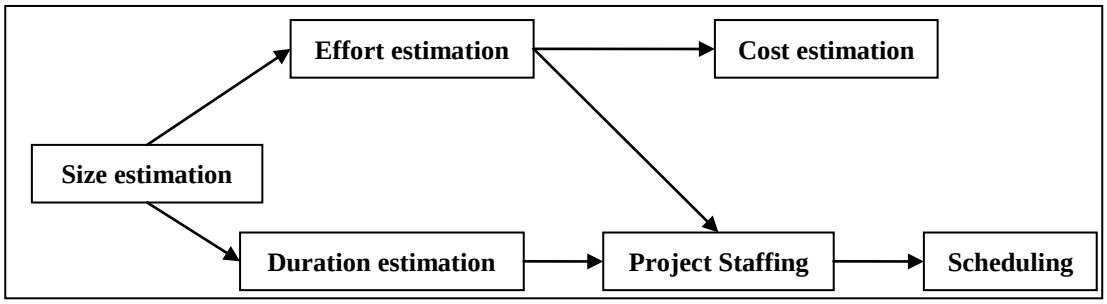
Hence, project planning is undertaken with utmost care and attention.

Planning includes:

1. **Estimation :** The following project attributes are estimated:
   - **Cost:** How much is it going to cost to develop the software product?
   - **Duration:** How long is it going to take to develop the product?
   - **Effort:** How much effort would be necessary to develop the product?

   The **effectiveness** of all later planning activities such as scheduling and staffing is **dependent** on the **accuracy** with which these three estimations have been made.

2. **Scheduling:** The schedules for manpower and other resources are developed.
3. **Staffing:** Staff organization and staffing plans are made.
4. **Risk management:** This includes risk identification, analysis, and abatement planning.
5. **Miscellaneous plans:** This includes making several other plans such as quality assurance plan, and configuration management plan, etc.



The figure shows the **order** in which the **planning activities** are undertaken.

- The **size estimation** is the first activity done by a project manager during project planning.
- Based on the **size** estimation, the **effort** required to complete a project and the **duration** over which the development is to be carried out are estimated.
- Based on the **effort** estimation, the **cost** of the project is computed.
- The estimated **cost** forms the basis for **price negotiations** with the customer.
- Other planning activities such as **staffing, scheduling** etc. are done based on the **effort** and **duration** estimates.

- The size of a project is **not** the number of **bytes** that the **source code** occupies, neither is it the **size** of the **executable code**.
- **The project size is a measure of the problem complexity in terms of the effort and time required to develop the product.**
- Currently, two metrics are popularly being used to measure size:
  1. lines of code (LOC)
  2. function point (FP)

Lines of Code (LOC)

- LOC measures the size of a project by **counting the number of source instructions** in the developed program.
- While counting the number of source instructions, **comment lines and header lines** are ignored.
- Determining the LOC count at the end of a project is very simple.
- **Accurate estimation** of LOC count at the beginning of a project is a **very difficult task**.

Steps (systematic guess work) taken by project manager:
1. Divide the problem into modules
2. Divide the modules into sub-modules until the estimation of the leaf level is possible.

Advantages:
1. LOC is the **simplest** among all metrics available to measure project size.
2. Consequently, this metric is extremely **popular**.

Disadvantages:
1. LOC is a measure of **coding activity alone**.
   - LOC does **not consider** the total **efforts for coding**.
   - So, it is not a good metric for project size.
2. LOC count depends on **the choice of specific instructions**.
   - This can be overcome by **counting the language tokens** rather than the lines of code.
   - But, e.g. if in a C prog. **switch** case is used instead of **else if** ladder, **tokens** will be **different**.
3. LOC measure **correlates poorly** with the **quality** and **efficiency** of the code.
   - Larger code size does not necessarily imply better quality of code or higher efficiency.
4. LOC metric **penalizes** use of **higher-level programming languages** and **code reuse**.
   - If library routines are used LOC will be small.
   - If effort is calculated based on this, programmers will be discouraged to reuse code.
   - Also, object-oriented programming makes relationship between code and LOC less precise.
5. LOC metric measures the **lexical complexity** of a program and does not address the more important issues of **logical and structural complexities**.
6. It is very **difficult to accurately estimate** LOC of the final program from **problem specification**.

Function Point Metrics

FP Metric overcomes many of the short comings of the LOC Metric. So it is steadily gaining popularity. Main advantage is that it can be **easily computed** from the problem specification itself.

Conceptual idea for FP metric is:
The size of a software product is directly dependent on the number of different high level functions or features it supports.

Each feature may take very different amount of efforts to develop. An implicit assumption can be that more the number of data items that a function reads from the user and the outputs and the more the number of files accessed, the higher is the complexity of the function.

Each feature when invoked typically reads some input data and then transforms those to the required output data. The computation of the number of input and output data items would give a more accurate indication of the code size compared to simply counting the number of high level functions supported by the system.

Steps for Function point Metric Computation

1. Compute the unadjusted function point (UFP) using a heuristic expression.
2. Refine UPF to reflect the actual complexities of the different parameters used in UFP computation.
3. Compute FP by further refining UFP to account for the specific characteristics of the project that can influence the entire development effort.

STEP 1: UFP Computation

The unadjusted function points (UFP) is computed as the weighted sum of five characteristics of a product as shown in the following expression.
The associated weights are empirical-experimental.

UFP = (Number of inputs) * 4 + (Number of outputs) * 5 + (Number of inquiries) * 4 + (Number of files) * 10 + (Number of interfaces) * 10

1. **Number of inputs:** Each data item input by the user is counted. Inquires are user commands that require no input data value from user. These inquires are counted separately in point no. 3. Related data items are not counted separately but are grouped and considered as a single input.
   e.g. name, age, address, etc of a person are together considered as a single input as they describe single person.

2. **Number of outputs:** Reports printed, screen outputs, error messages produced are all different outputs of the system.
   While computing the number of outputs, the individual data items within a report are not considered; but a set of related data items is counted just as a single output.
3. **Number of inquiries:** An inquiry is a user command (without any data input) and only requires some actions to be performed by the system. So these are distinct interactive queries without data input that can be made by the user.
4. **Number of files:** The files here are the logical files. A logical file represents a group of logically related data. Logical files include data structures as well as physical files.
5. **Number of interfaces:** The interfaces denote the different mechanisms that are used to exchange information with other systems. Data files on tapes, disks, communication links with other systems, etc are example of interfaces.

STEP 2: Refine parameters

Each parameter in calculation of UFP has been implicitly assumed to be of average complexity.
UFP is refined by taking into account the complexities of the parameters of UFP computation. The complexity of each parameter is graded into three broad categories- simple, average or complex.

The weights for different parameters are as follows:

Type	Simple	Average	Complex
Input(I)	**3**	**4**	**6**
Output(O)	**4**	**5**	**7**
Inquiry(E)	**3**	**4**	**6**
Number of files(F)	**7**	**10**	**15**
Number of interfaces	**5**	**7**	**10**

e.g. rather than each input being computed as 4 FPs, very simple inputs are computed as 3 FPs and very complex inputs as 6 FPs.

STEP 3: Refine UFP based on complexity of the overall project.

There are 14 different parameters that can influence the development effort.
Each of these 14 parameters is assigned a value from 0 (not present or no influence) to 6 (strong influence).

Resulting numbers are summed to give the total **degree of influence (DI)**.
A **technical complexity factor (TCF)** is computed as **0.65 + 0.01*DI.**

As DI can vary from 0 to 84, TCF can vary from 0.65 to 1.49.
Finally, FP is computed as **FP = UFP * TCF.**

14 Function Point Relative Complexity Adjustment Factors

1. Requirement for reliable backup and recovery
2. Requirement for data communication
3. Extent of distributed processing
4. Performance requirements
5. Expected operational environment
6. Extent of online data entries
7. Extent of multi-screen or multi-operation online data input
8. Extent of online updating of master files
9. Extent of complex inputs, outputs, online queries and files
10. Extent of complex data processing
11. Extent that currently developed code can be designed for reuse
12. Extent of conversion and installation included in the design
13. Extent of multiple installations in an organization and variety of customer organizations
14. Extent of change and focus on ease of use

Short comings:

FP metric does not take into account the algorithmic complexity of a function.

FP metric implicitly assumes that the effort required to design and develop any two different functionalities of the system is the same. But this is not true.

To overcome this problem, an extension to the function point metric called "feature point" metric has been proposed.

Feature point metric incorporates algorithmic complexity as an extra parameter.
This parameter ensures that the computed size using the feature point metric reflects the fact that higher the complexity of a function, the greater the effort required to develop it – therefore it should have larger size compared to a simpler function.

Though these metrics are language independent and can be computed easily from the SRS document during project planning stage itself, because these metrics are subjective in nature, they leave sufficient scope for debate.

Grouping of data items (used in equation for UFP) can be subjective and hence different project managers may arrive at different function point measure for the same problem.

EXAMPLE 1

Determine the function point measure of the size of the following supermarket software.
A supermarket needs to develop the following software to encourage regular customers. For this, the customer needs to supply his/her residence address, telephone number, and the driving license number. Each customer who registers for this scheme is assigned a unique customer number (CN) by the computer. Based on the generated CN, a clerk manually prepares a customer identity card after getting the market manager's signature on it. A customer can present his customer identity card to the check out staff when he makes any purchase. In this case, the value of his purchase is credited against his CN. At the end of each year, the supermarket intends to award surprise gifts to 10 customers who make the highest total purchase over the year. Also, it intends to award a 22 carat gold coin to every customer whose purchase exceeded Rs. 10,000. The entries against the CN are reset on the last day of every year after the prize winners' lists are generated. Assume that various project characteristics determining the complexity of software development to be average.

INPUTS: 1. Customer details 2. check out card on purchase
OUTPUTS: 1. CN number (from registration) 2. 10 customers with highest total purchase. 3. List of customers with purchase more than 10K.
Inquiries: reset entries after winner list is generated
FILE: 1. Customer details 2. Purchases
Interfaces : none

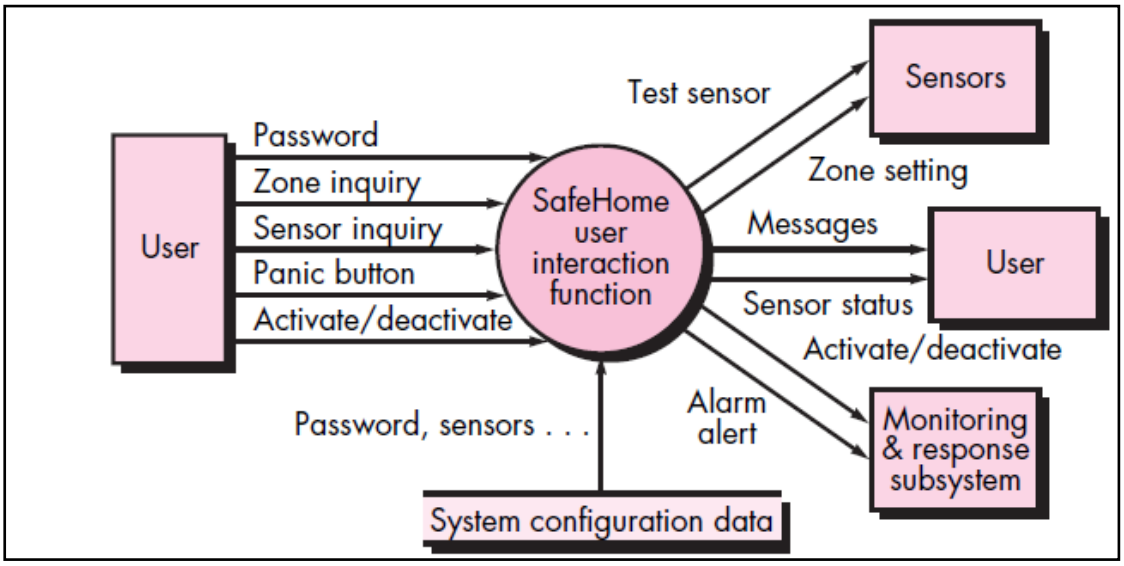
Calculate FP
STEP 1:
UFP = (Number of inputs) * 4 + (Number of outputs) * 5 + (Number of inquiries) * 4 + (Number of files) * 10 + (Number of interfaces) * 10

UPF = (2) * 4 + (3) * 5 + (1) * 4 + (2) * 10 + (0) * 10 = 47

STEP 2:
As registration has only one output CN so it is considered to have simple complexity.
Refined UFP = (2) * 4 + [(1) *4 + (2) * 5] + (1) * 4 + (2) * 10 + (0) * 10 = 46

STEP 3:
DI = 14 * 4 = 56 [because all adjustment factors have average values]
TCF = 0.65 + 0.01(DI) = 0.65 + 0.01(56) = 1.21
FP = UFP * TCF = 46 * 1.21 = 55.66

EXAMPLE 2



Three external inputs—**password, panic button**, and **activate/deactivate**
Two external inquiries—**zone inquiry** and **sensor inquiry**.
One ILF (**system configuration file**)
Two external outputs (**messages** and **sensor status**)
four EIFs (**test sensor, zone setting, activate/deactivate,** and **alarm alert**)

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

DI is assumed to be 46 as moderately complex.

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