

BATCH 15 - SE LAB - RECORD - SOFT
COPY

20VV1A12-(59, 60, 61, 63)

June 24, 2022

SRS DOCUMENT

Section 1 : Introduction

Purpose :

- This document describes the software requirements for a Ariane 5 Rocket Launching System. It is intended for the designer, developer, the maintainer and the user of the system.

Scope :

- The Rocket Launching System is designed to build a distributed software application that manages the launching of Ariane 5 rocket.
- Illustrate issues with requirements specification, multi organisational working, artificial system validation and some of the problems of software reuse.
- It also shows the organisational complexity of systems of development and how organisational issues can lead to system failure.

Definitions/Abbreviations :

- Ballistic : moving under the force of gravity.
- Terminal : forming or situated at the end or extremity of something.
- Mach : The ratio of the speed of a body to the speed of sound in the surrounding medium.
- Altitude : The height of an object or point in relation to sea level or ground level.
- Flight : The action or process of flying through the air.

References :

- <https://iansommerville.com/software-engineering-book/case-studies/ariane5/>
- <http://www.rvs.uni-bielefeld.de/publications/Incidents/DOCS/Research/Rvs/Misc/Additional>
- www.google.com

Section 2 : General Description

Overall Description :

This section usually consists of six subsections, as follows :

- Product Perspective
- Product Functions
- User Characteristics
- Constraints
- Assumptions and Dependancies
- Requirements

Product Perspective :

The product is a software designed to launch a Rocket. Constitutes Internal Calculators that help in maintaining the rocket in regular parameters of flight. Scientific, Development and Maintenance Team interacts with the software.

Product Functionality :

Product provides calculators like Ballistic Flight, Terminal Velocity, Rocket Altitude, Mach and Speed of Sound. Each has its own importance in launching a rocket.

Design and Implementation Constraints :

- As the product is inherited from the case study of Ariane 5 Launch.Accident, the rocket has blasts in 37 secs into the flight after launch.
- Reason being no proper code reuse.
- The content didn't provided enough information about how we can develop a complete end to end rocket launching software.
- Also many companies keep their source code confidential making it much more difficult to get ideas related to it.
- Understanding how various items are involved and how they can be managed into one so that they can lead to meaningful outcomes it's a long task.

Assumptions and Dependencies :

- Any of the modern operating systems are supported.
- Windows, Linux, Mac are basic OS that are fully supportive.
- Browser should be of the latest generation.
- Browsers like Internet Explorer are not supported.
- An active internet connection is highly recommended.
- No further packages or software plugins are required.
- The specific hardware and software due to which the product will be run.

Section 3: Specific Requirements

External Interface Requirements:

Functional Requirements:

User Interfaces:

1. Home Page:

- For Providing navigation buttons to different features in the software.
- This home page is designed for showing the features of the software designed.
- contains BFC, TVC, RAC, MSSC
- User will input numeric values and output is shown.

2. Ballistic Flight Calculator

- This program calculates the maximum height of a launched ballistic shell or a shell with drag, and the time from launch when the maximum height is reached.
- has attributes Initial Velocity, Mass, Cross Section Area etc.
- User will input numeric values and output is shown. Further user can use RAC and home navigation buttons.

3. Terminal Velocity Calculator

- This Program Calculates the Terminal Velocity of a falling object. Circular Orbit Calculator and velocity of an object in a circular orbit about the earth, moon, or mars.
- has mass, Cross Section Area, Drag Coefficient etc.
- User will input numeric values and output is shown. Further user can use RAC and home navigation buttons.

4. Rocket Altitude Calculator

- This program calculates the terminal velocity of a falling object.
- Angle A, Angle B, Angle C etc.
- User will input numeric values and output is shown. Further user can use RAC and home navigation buttons.

Mach and Speed of Sound Calculator

- This program calculate the speed of sound as a function of temperature and the mach number as fuction of object speed and speed of sound.
- has attributes Altitude, Speed etc.
- User will input numeric values and output is shown. Further user can use RAC and home navigation buttons.

Non Functional Requirements

Performance Requirements

1. The Rocket Altitude Calculator (RAC) will calculate the Altitude of the rocket when needed by other interfaces.
2. The Terminal Velocity of Rocket when needed by other interfaces.
3. All calculators should work in coordination with each other.

Safety and Security Requirements:

1. It's advised to ensure the right usernames and password for authentication of scientist or Launch Engineer.
2. Product may not allowed to use by general users
3. Systems running the software must satisfy the hardware and feature requirements for obtaining the maximum accurate and less error prone output from the product.
4. Systems running the software should maintain antivirus and firewall protection for reliable performance of the product.

Software Quality Attributes:

- Portability
- Interoperability
- Modifiability
- Performance
- Availability
- Re-usability
- Scalability
- verifiability
- Manage Complexity
- Security
- Openness

USE CASE DIAGRAM

Description:

Name:	Ariane 5 Rocket Launch Calculator
Actors:	Launch Engineer, Test Engineer and Software Development Engineer.
Triggers:	User opens application and initiates software.
Pre-Condition:	1. The Hardware components must be in good working condition.
	2. The Calculators must take in accurate input and accurate output.
Post-Condition:	1. All performed operations and their results must be stored.
	2. Repeated Testing must be conducted to identify errors.
Actor Actions	System Actions
1)The Launch Engineer launches the rocket.	3)Request sent to Terminal Velocity Calculator to provide terminal velocity value
2)Launch Engineer provides input to the Ballistic Flight Calculator	5)Request sent to Rocket Altitude Calculator to provide altitude value
4) Launch Engineer provides input to the Terminal Velocity Calculator	6) Final output given by Ballistic Flight Calculator
7)Launch Engineer provides input to the Mach and Speed of Sound Calculator	8) Output given by Mach and Speed of Sound Calculator

Alternative Flow of Events:	1)In case errors are encountered by the software, the Test Engineer refers to the Log Report and reports the errors to the Software Development Engineer to fix the errors.
	2)User must ensure that they have entered the correct values. If they have not they must re enter the values to get the correct

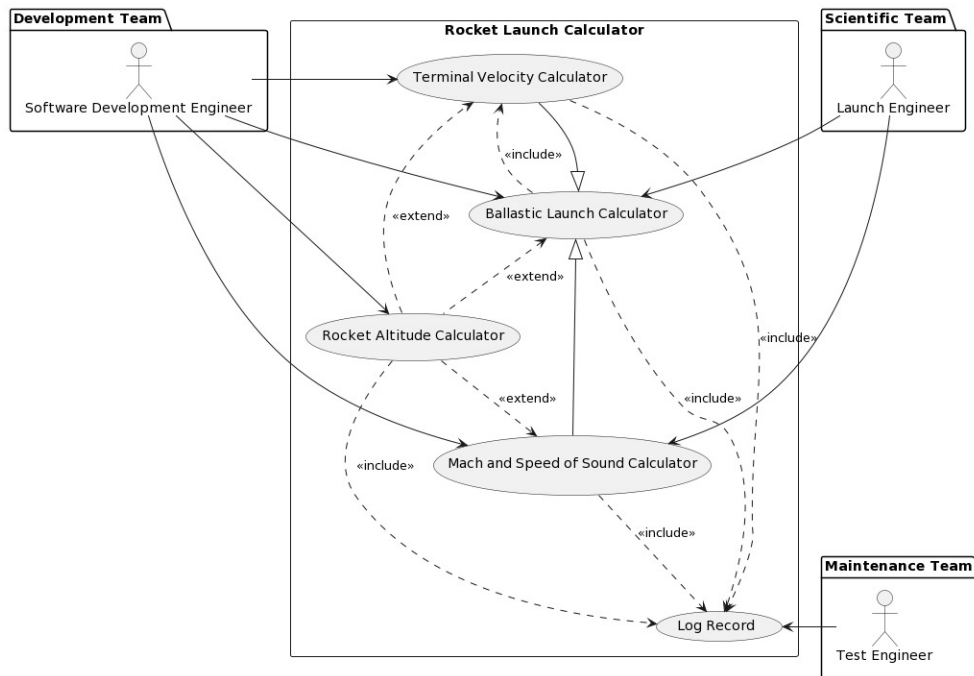


Figure 1:

Usecase Diagram for Ariane 5

Usecase Diagram code:

```
@startuml
    package "Development Team"
    actor "Software Development Engineer" as SDE
    package "Maintenance Team"
    actor "Test Engineer" as TE
    package "Scientific Team"
    actor "Launch Engineer" as LE
    rectangle "Rocket Launch Calculator"
    usecase "Ballistic Launch Calculator" as BFC
    usecase "Terminal Velocity Calculator" as TVC
    usecase "Rocket Altitude Calculator" as RAC
    usecase "Mach and Speed of Sound Calculator" as MSSC
    usecase "Log Record" as LR
    BFC .down> TVC : << include >>
    RAC .up> BFC : << extend >>
    RAC ..> TVC : << extend >>
    RAC ..> MSSC : << extend >>
    BFC -down-> LR : << include >>
    TVC .down> LR : << include >>
    RAC .down> LR : << include >>
    MSSC .down> LR : << include >>
    TVC — right—> BFC
    MSSC -right—> BFC
    LE -> BFC
    LE -> MSSC
    SDE -right-> TVC
    SDE -right-> BFC
    SDE -right-> RAC
    SDE -right-> MSSC
    TE -left-> LR
@enduml
```

Class Diagram

Aim: To draw class diagram for notes and file management

Description:

- Class Diagram : In software engineering,a class diagram in the Unified Modelling Language(UML) is a type of static structure diagram that describes the structure of a system by showing the system classes ,their attributes,operations and the rectangle among objects.
- Class : A description of a group of objects with similar roles in the system.Simply user can say that class is a combination of data numbers and member functions.
- Class Notation : A Class notation consists of three parts.
 1. Class Name :
 - The name pf the class appears in the first position
 2. Class Attribute :
 - Attributes are shown in the second partition .
 - The attribute type is shown after the colon.Attributes map on to member variables in code.
 3. Class Operations(Methods) :
 - Operations are shown in the third partition .
 - The return type of a method shown after the colon at the end of the method signature.

Description of class:

1. Ballastic Flight Calculator(BFC):
 - BFC is the class name and it is one of the key actors of the rocket launch system.
 - Initial velocity,mass,cross section area,altitude,drag coefficient,terminal velocity,max height are attributes.
 - Compute BFC() is for computing ballastic flight.
 - home(),calcu;ate rocket altitude() are supporting ,methods.

2. Terminal Velocity Calculator(TVC):
 - TVC inherits attributes methods, from BFC.
 - Compute TVC(), is method for calculating terminal velocity.
3. Rocket Altitude Calculator(RAC):
 - Rocket Altitude Calculator is a supporting class and also a autonomous class in the system.
 - angle A,angle B, angle C ,angle D , reference length.
 - Compute RAC() is method for calculating RAC.
4. Mach and Speed of Sound Calculator(MSSC):
 - Speed , Speed of sound ,mach are attributes.
 - inherits BFC class
 - Compute MSSC() is individual method.
5. Log Record(LR):
 - Log Report class is used to store data of all.
 - Signal is an attribute.
 - Calculate BFC(),TVC(),RAC(),MSSC() all methods of the class.

Relationship used in the class diagram as:

1. Inheritance:A solid line with a hollow arrow head that point from the child to the parent class.
2. Simple Association : A line connectivity between two classes.
3. Composition : A special type of aggregation which denotes strong ownership between two classes when one class is part of another class.
4. Aggregation : Aggregation relationship shows a class as if it's a part of or subordinate to another class free.

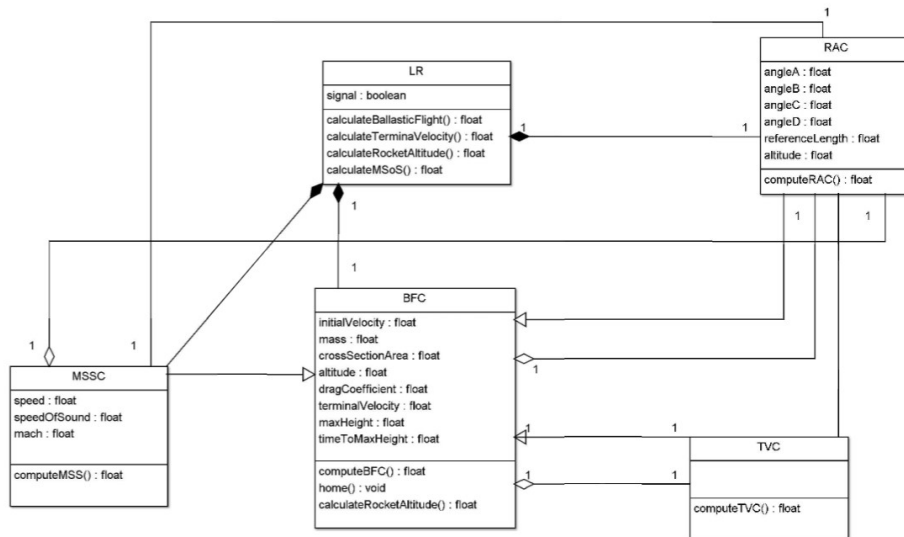


Figure 2: Class Diagram

Class Diagram for Ariane 5

Java code for Ballastic Flight Calculator

```
import java.util.Vector;
public class BFC
{
    private float initialVelocity;
    public float mass;
    public float crossSectionArea;
    public float altitude;
    public float dragCoefficient;
    public float terminalVelocity;
    public float maxHeight;
    public float timeToMaxHeight;
    public LR myLR;
    public Vector myTVC;
    public TVC myTVC;
    public TVC myTVC;
    public RAC myRAC;
    public float computeBFC()
    {
        return 0.0;
    }
    public void home() {
    }
    public float calculateRocketAltitude() {
        return 0.0;
    }
}
```

Java code for Terminal Velocity Calculator

```
import java.util.Vector;
public class TVC extends BFC {
    public Vector myBFC;
    public BFC myBFC;
    public Vector myRAC;
    public BFC myBFC;
    public float computeTVC()
    {
        return 0.0;
    }
}
```


Java code for Rocket Altitude Calculator

```
import java.util.Vector;
public class RAC
{
    private float angleA;
    private float angleB;
    private float angleC;
    private float angleD;
    private float referenceLength;
    public float altitude;
    public LR myLR;
    public Vector myTVC;
    public MSSC myMSSC;
    public MSSC myMSSC;
    public Vector 1;
    public float computeRAC()
    {
        return 0.0;
    }
}
```

Java code for Mach and speed of sound calculator

```
import java.util.Vector;
public class MSSC extends BFC {
    private float speed;
    public float speedOfSound;
    public float mach;
    public Vector myLR;
    public RAC myRAC;
    public RAC myRAC;
    public float computeMSS()
    {
        return 0.0;
    }
}
```

Java code for Log Report

```
import java.util.Vector;
public class LR
{
    private boolean signal;
    public BFC myBFC;
    public Vector myMSSC;
    public RAC myRAC;
    public float calculateBallasticFlight()
    {
        return 0.0;
    }
    public float calculateTerminaVelocity()
    {
        return 0.0;
    }
    public float calculateRocketAltitude() {
        return 0.0;
    }
    public float calculateMSoS() {
        return 0.0;
    }
}
```

Sequence Diagram

Aim : To draw sequence diagram for rocket launch software

Description :

- Launch Engineer inputs data
- BFC receives input and computes ballastic flight
- TVC receives input and computes terminal velocity
- RAC receives input and computes rocket altitude
- MSSC receives input and computes Mach and Speed of sound
- Test Engineer request for log data
- Log record gives response back to Test

Description of roles :

- Launch Engineer : Responsible for activating software requires valid credentials to access the system input data for derived calculator. Receives response back from the system.
- Test Engineer : Responsible for identifying bugs and errors in the system.
- BFC : Receives input ,computes ballastic flight , sends response back to launch engineer.
- TVC : Receives input from user , compute terminal velocity and sends response back to user.
- RAC : Takes input from launch engineer, computes Rocket Altitude Calculator and sends response back.
- MSSC : Takes input,calculates mach and speed of sound.
- LR : Records every transaction in the system.

Sequence Diagram Code :

```
@startuml
actor "Launch Engineer" as foo
actor "Test engineer" as foo1
participant "BFC" as foo2
participant "TVC" as foo3
participant "RAC" as foo4
participant "MSSC" as foo5
database "LR" as foo6
activate foo6
foo1 →foo6 : request Log Report
foo6 →foo1 : response of Log Sheet
foo →foo2 :inputs data
alt successful case
activate foo2
foo2 →foo : response about Ballastic flight
else some failure
foo2 →foo : input proper values
foo2 →foo6 : save data into log
deactivate foo2
end
foo →foo3 : inputs data
alt successful case
activate foo3
foo3 →foo : response about Terminal velocity
else some failure
foo3 →foo : input proper values
foo3 →foo6 : save data into log
deactivate foo3
end
foo →foo4 : inputs data
alt successful case
activate foo4
foo4 →foo : response about Rocket Altitude
else some failure
foo4 →foo : input proper values
foo4 →foo6 : save data into log
deactivate foo4
end
foo →foo5 : inputs data
alt successful case
```

```
activate foo5
foo5 →foo : response about speed and mach
else some failure
foo5 →foo : input proper values
foo5 →foo6 : save data into log
deactivate foo5
deactivate foo6
end
@enduml
```

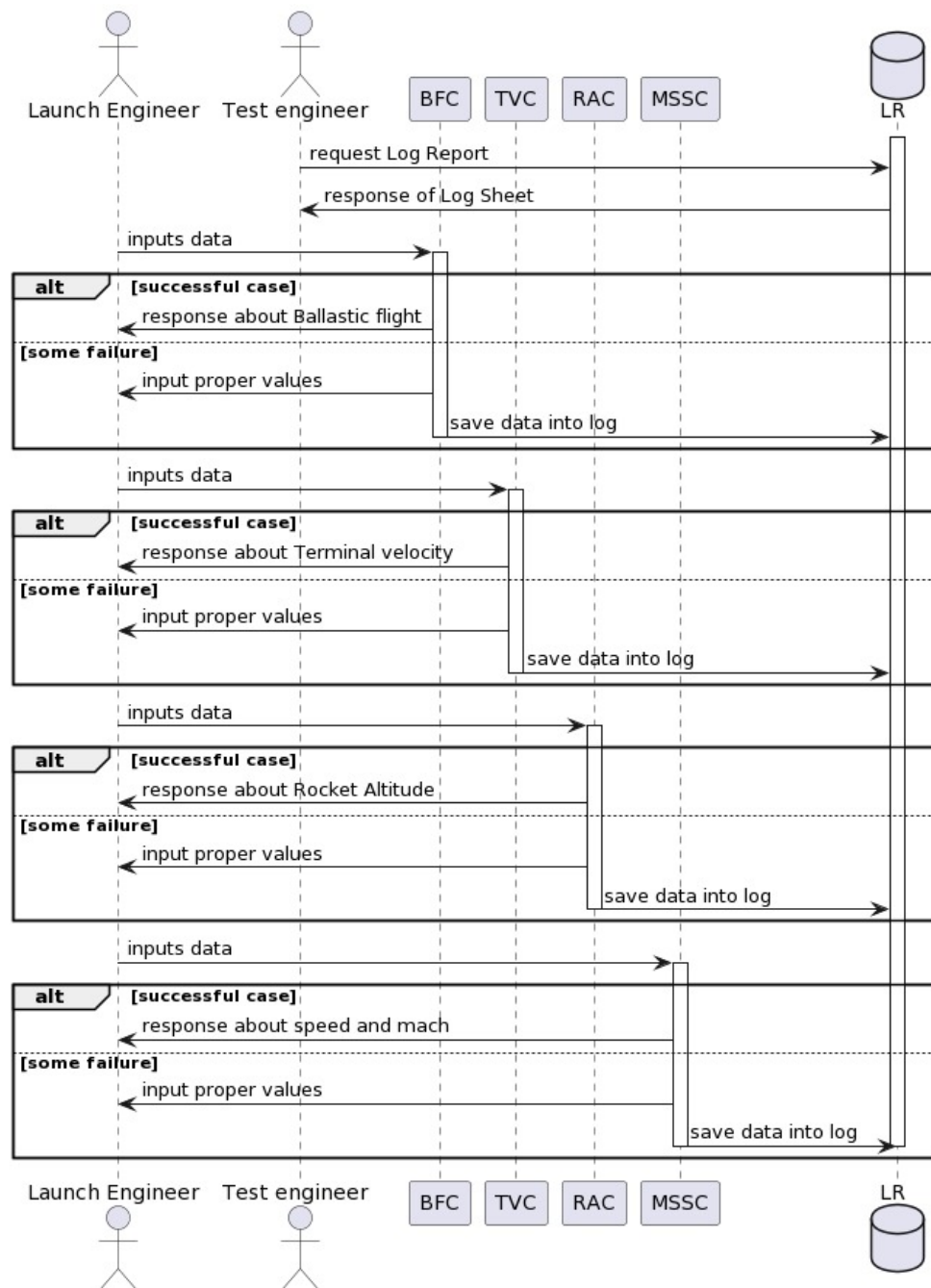


Figure 3: Sequence diagram

Ballistic Flight Calculator Code:

```
@startuml
actor "Launch Engineer" as foo
actor "Test engineer" as foo1
participant "BFC" as foo2
participant "TVC" as foo3
participant "RAC" as foo4
participant "MSSC" as foo5
database "LR" as foo6
activate foo6
foo1 ->foo6 : request Log Report
foo6 ->foo1 : response of Log Sheet
foo ->foo2 :inputs data
alt successful case
activate foo2
foo2 ->foo : response about Ballastic flight
else some failure
foo2 ->foo : input proper values
foo2 ->foo6 : save data into log
deactivate foo2
end
foo ->foo3 : inputs data
alt successful case
activate foo3
foo3 ->foo : response about Terminal velocity
else some failure
foo3 ->foo : input proper values
foo3 ->foo6 : save data into log
deactivate foo3
end
foo ->foo4 : inputs data
alt successful case
activate foo4
foo4 ->foo : response about Rocket Altitude
else some failure
foo4 ->foo : input proper values
foo4 ->foo6 : save data into log
deactivate foo4
end
foo ->foo5 : inputs data
alt successful case
```



```
activate foo5
foo5 →foo : response about speed and mach
else some failure
foo5 →foo : input proper values
foo5 →foo6 : save data into log
deactivate foo5
deactivate foo6
end
@enduml
```

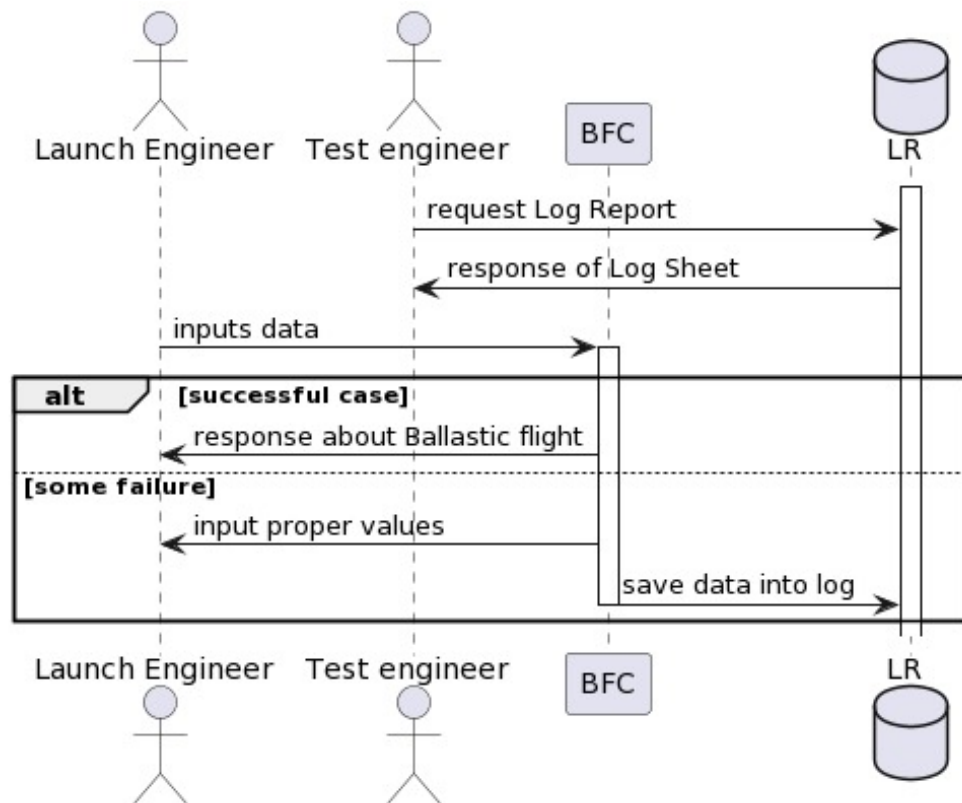


Figure 4: Sequence diagram 1

Ballastic Flight Calculator for Ariane 5

Terminal Velocity Calculator Code:

```
@startuml
actor "Launch Engineer" as foo
actor "Test engineer" as foo1
participant "TVC" as foo3
database "LR" as foo6
activate foo6
foo1 →foo6 : request Log Report
foo6 →foo1 : response of Log Sheet
foo →foo3 : inputs data
alt successful case
activate foo3
foo3 →foo : response about Terminal velocity
else some failure
foo3 →foo : input proper values
foo3 →foo6 : save data into log
deactivate foo3
end
@enduml
```

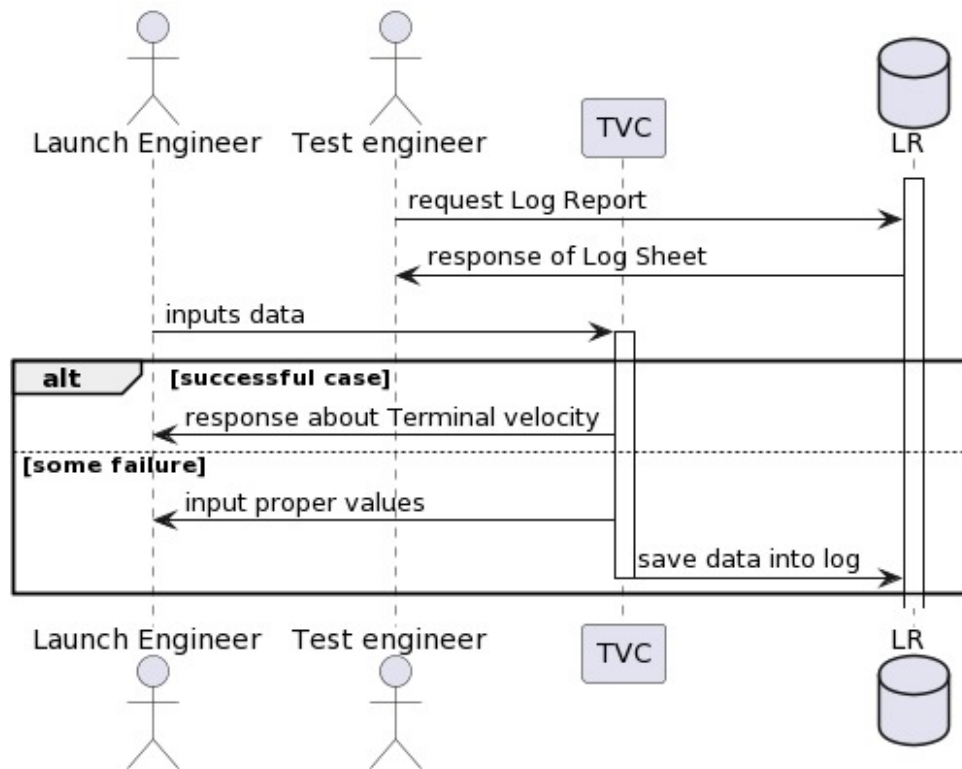


Figure 5: Sequential diagram 2

Terminal Velocity Calculator for Ariane 5

Rocket Altitude Calculator Code:

```
@startuml
actor "Launch Engineer" as foo
actor "Test engineer" as foo1
participant "RAC" as foo4
database "LR" as foo6
activate foo6
foo1 →foo6 : request Log Report
foo6 →foo1 : response of Log Sheet
foo →foo4 : inputs data
alt successful case
activate foo4
foo4 →foo : response about Rocket Altitude
else some failure
foo4 →foo : input proper values
foo4 →foo6 : save data into log
deactivate foo4
end
@enduml
```

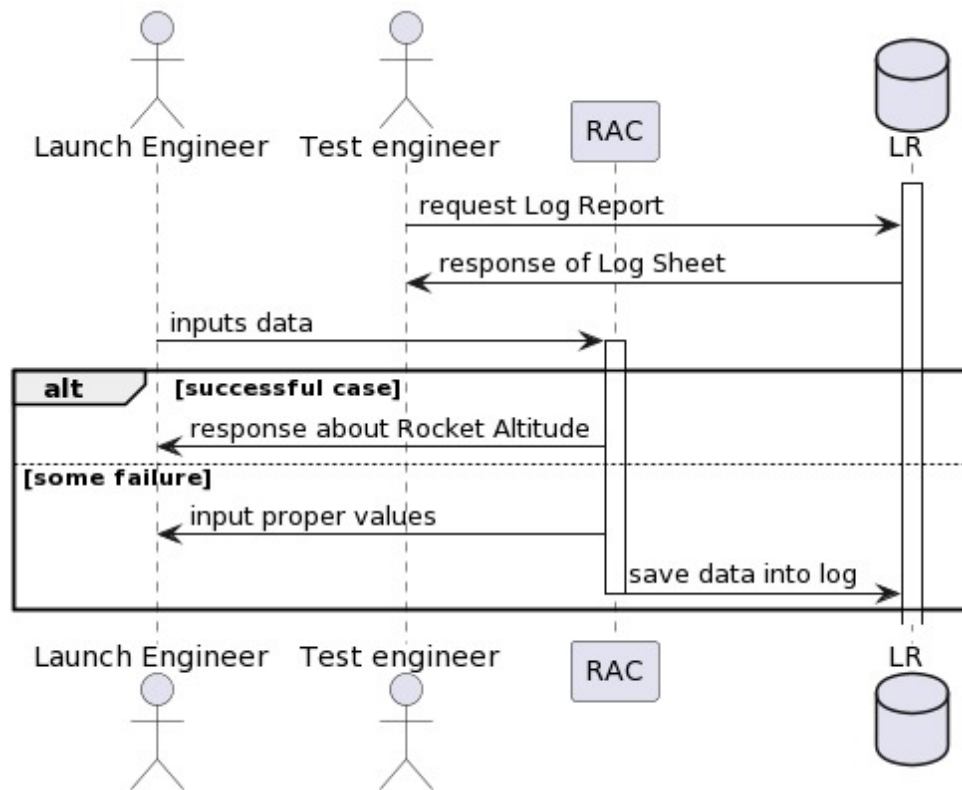


Figure 6: Sequence Diagram 3

Rocket Altitude Calculator for Ariane 5

Mach and Speed of Sound Calculator Code:

```
@startuml
actor "Launch Engineer" as foo
actor "Test engineer" as foo1
participant "MSSC" as foo5
database "LR" as foo6
activate foo6
foo1 →foo6 : request Log Report
foo6 →foo1 : response of Log Sheet
foo →foo5 : inputs data
alt successful case
activate foo5
foo5 →foo : response about speed and mach
else some failure
foo5 →foo : input proper values
foo5 →foo6 : save data into log
deactivate foo5
deactivate foo6
end
@enduml
```

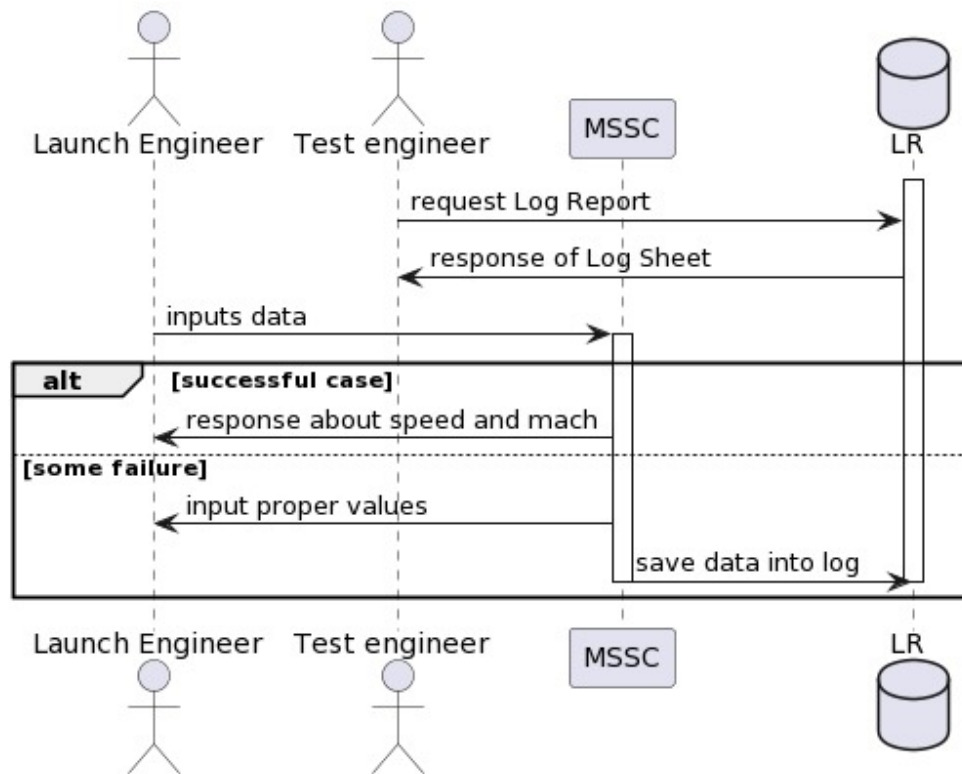


Figure 7: Sequence diagram 4

Mach and Speed of Sound Calculator

Log:

@startuml

actor "Test engineer" as foo1

database "LR" as foo6

activate foo6

foo1 →foo6 : request Log Report

foo6 →foo1 : response of Log Sheet

@enduml

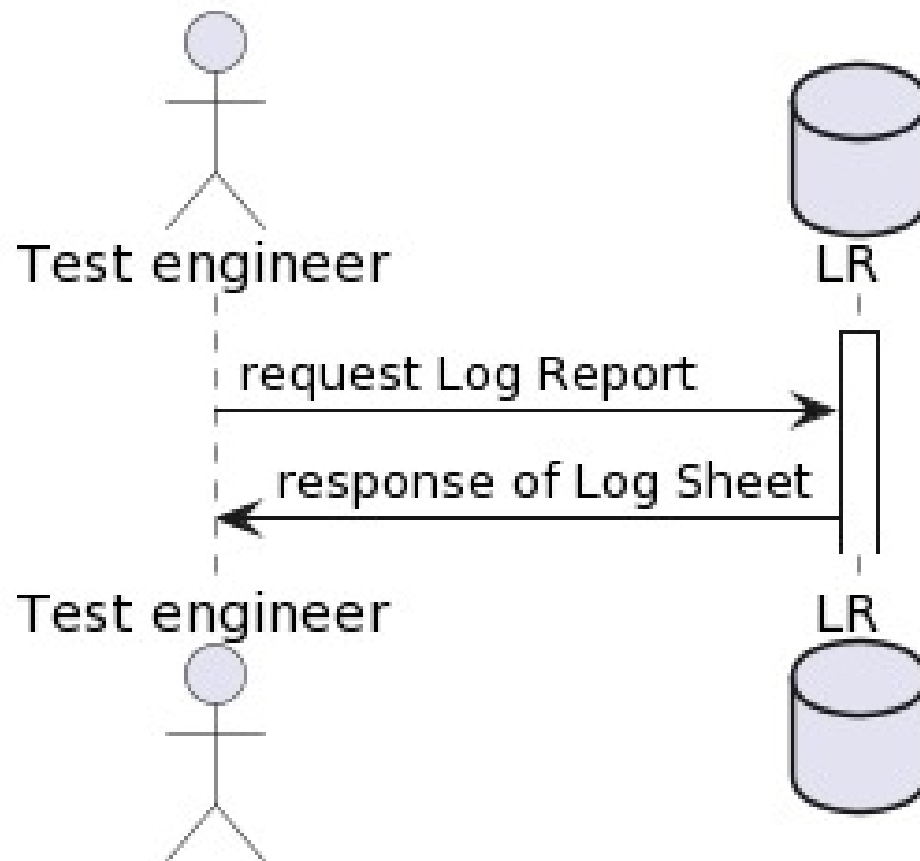


Figure 8: Sequential diagram 5

Log for Ariane 5

Collaboration diagram

Aim:-To draw a collaboration diagram for rocket for rocket launch software

Description

- The collaboration diagram used to show the relationships between the object and systems
- Instead of showing how the message ,It despects the architecture of recording in the system as it based on object oriented programming
- The collaboration diagram which is also known as communication diagram is used portray objects architecture in the system
- the diagrams use the sequence diagram and collaboration diagram to define and clarify the roles of objects that performing particular parts of events of use case

Collaboration Diagram Code

```
@startuml
actor "Launch Engineer" as foo
actor "Test engineer" as foo1
participant "BFC" as foo2
participant "TVC" as foo3
participant "RAC" as foo4
participant "MSSC" as foo5
autonumber
database "LR" as foo6
activate foo6
foo1 →foo6 : request Log Report
foo6 →foo1 : response of Log Sheet
foo →foo2 :inputs data
alt successful case
activate foo2
foo2 →foo : response about Ballastic flight
else some failure
foo2 →foo : input proper values
foo2 →foo6 : save data into log
deactivate foo2
end
foo →foo3 : inputs data
alt successful case
activate foo3
foo3 →foo : response about Terminal velocity
else some failure
foo3 →foo : input proper values
foo3 →foo6 : save data into log
deactivate foo3
end
foo →foo4 : inputs data
alt successful case
activate foo4
foo4 →foo : response about Rocket Altitude
else some failure
foo4 →foo : input proper values
foo4 →foo6 : save data into log
deactivate foo4
end
```

```
foo → foo5 : inputs data  
alt successful case  
  activate foo5  
foo5 → foo : response about speed and mach  
else some failure  
foo5 → foo : input proper values  
foo5 → foo6 : save data into log  
  deactivate foo5  
  deactivate foo6  
end  
@enduml
```

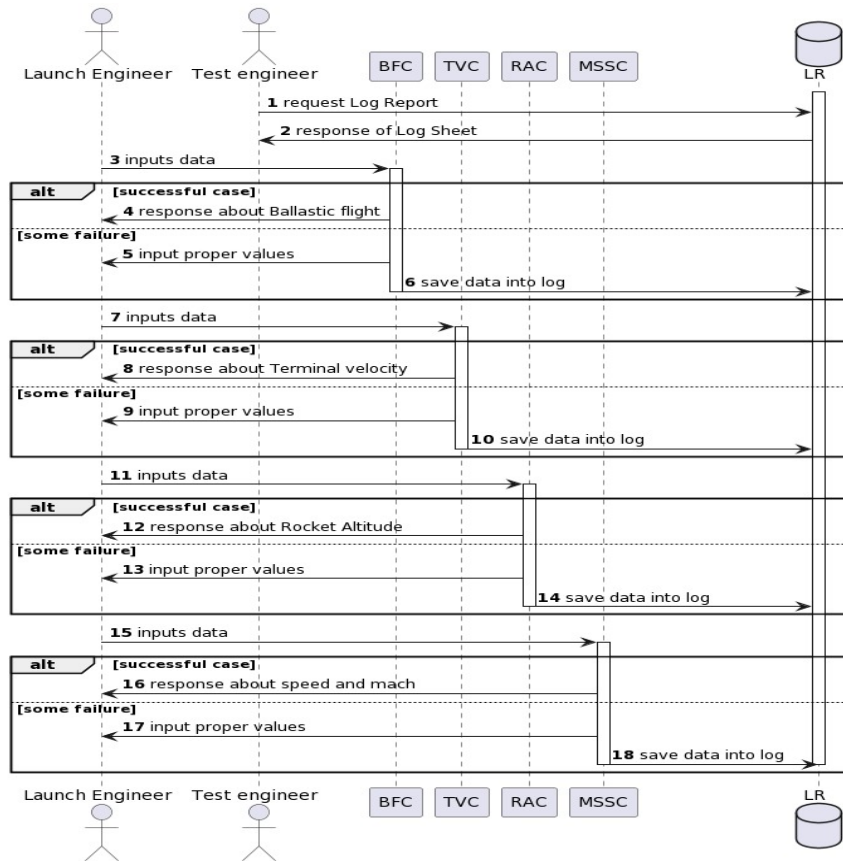


Figure 9: Collaboration diagram for Ariane 5

Collaboration diagram

COCOMO MODEL

AIM:- To estimate the cost of the personal insulin pump in a cocomo model

DESCRIPTION:-

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e number of Lines of Code. It is a procedural cost estimate model for software projects and is often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality. It was proposed by Barry Boehm in 1981 and is based on the study of 63 projects, which makes it one of the best-documented models.

The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort Schedule:

Effort:

- Amount of labor that will be required to complete a task. It is measured in person-months units.

Schedule:

- Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, months.
- Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below.

Boehm's definition of organic, semidetached, and embedded systems:

Organic –

- A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.

Semi-detached –

- A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, knowledge of the various programming environment lie in between that of organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered of Semi-Detached type.

Embedded –

- A software project requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.
All the above system types utilize different values of the constants used in Effort Calculations.

Types of Models:

COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. Any of the three forms can be adopted according to our requirements. These are types of COCOMO model:

- Basic COCOMO Model
- Intermediate COCOMO Model
- Detailed COCOMO Model

The first level, Basic COCOMO can be used for quick and slightly rough calculations of Software Costs. Its accuracy is somewhat restricted due to the absence of sufficient factor considerations.

Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project

phases, i.e in case of Detailed it accounts for both these cost drivers and also calculations are performed phase-wise henceforth producing a more accurate result. These two models are further discussed below.

Estimation of Effort: Calculations –

Basic Model –

$$E = a(KLOC)^b$$

$$\text{time} = c(\text{Effort})^d \quad \text{Person required} = \text{Effort} / \text{time}$$

The above formula is used for the cost estimation of for the basic COCOMO model,

and also is used in the subsequent models. The constant values a,b,c and d for the Basic Model for the different categories of system:

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code.

The development time is measured in months.

These formulas are used as such in the Basic Model calculations, as not much consideration of different factors such as reliability, expertise is taken into account, henceforth the estimate is rough.

Intermediate Model –

The basic Cocomo model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software systems. However, in reality, no system's effort and schedule can be solely calculated on the basis of Lines of Code. For that, various other factors such as reliability, experience, Capability. These factors are known as Cost Drivers and the Intermediate Model utilizes 15

such drivers for cost estimation.

Classification of Cost Drivers and their attributes:

(i) Product attributes –

Required software reliability extent

Size of the application database

The complexity of the product

(ii) Hardware attributes –

Run-time performance constraints

Memory constraints

The volatility of the virtual machine environment

Required turnabout time

(iii) Personnel attributes –

Analyst capability

Software engineering capability

Applications experience

Virtual machine experience

Programming language experience

(iv) Project attributes –

Use of software tools

Application of software engineering methods

Required development schedule

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Detailed Model –

Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the

software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are:

Planning and requirements

System design

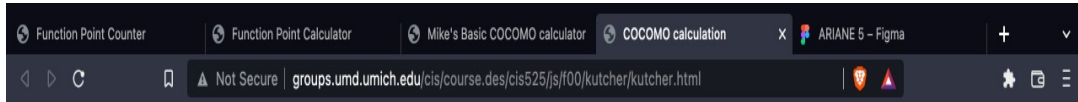
Detailed design

Module code and test

Integration and test


Cost Constructive model

The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software lifecycle.



BASIC COCOMO calculation

☐ Organic Mode:
☐ Semi-detached Mode:
☒ Embedded Mode:



KLOC estimate:

This application derives the **COCOMO** software engineering metric as found in Robert Pressman's *"Software Engineering, A Practitioner's Approach"*, (McGraw-Hill,97). The specific version utilized here is the "basic" model.

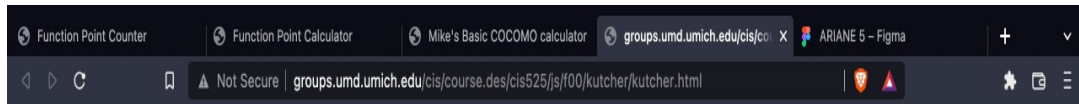
The available modes, selected by the radio buttons, are explained by Pressman as follows, (page 122):

ORGANIC: Relatively small, simple software projects in which small teams with good application experience work to a set of less than rigid requirements.

SEMI-DETACHED: An intermediate, (in size and complexity), software project in which teams with mixed experience levels must meet a mix of rigid and less than rigid requirements.

EMBEDDED: A software project that must be developed within a set of tight hardware, software and operation constraints.

INSTRUCTIONS: Choose a mode for your project based on the criteria above by selecting a corresponding radio button. Then enter the estimated lines of code - in the THOUSANDS - in the KLOC input text box. To see your result, simply hit the "calculate COCOMO" button. If you enter anything other than a number, you will be shown a prompt informing you so. You will then be taken to the results page, so that you can, if desired, see the coefficients for the mode that you selected. The results, however, will be read "NaN" and will be bogus. Hit the "back" button on your browser, select the "reset" button, re-select the mode and enter a number in the KLOC text box and finally hit the "calculate COCOMO" button.



YOUR BASIC COCOMO RESULTS!!								
MODE	"A" variable	"B" variable	"C" variable	"D" variable	KLOC	EFFORT, (in person/months)	DURATION, (in months)	STAFFING, (recommended)
embedded	3.6	1.2	2.5	0.32	300	3379.465416094449	33.6644381591599	100.38680580727042

Explanation: The coefficients are set according to the project mode selected on the previous page, (as per Boehm,81). The final estimates are determined in the following manner:

effort = $a * KLOC^b$, in person/months, with KLOC = lines of code, (in the thousands), and:

duration = $c * effort^d$, finally:

staffing = effort/duration

For further reading, see Boehm, "Software Engineering Economics", (81)

WARNING: If you see "NaN" in any field above, you have entered an **INVALID** value for KLOC!! Hit the "BACK" button on your browser, hit the "RESET" button, and enter a **DECIMAL NUMBER** in the KLOC input text box!

Thank you, and happy software engineering!



Function Point Analysis

Aim: Calculate effort using FP oriented estimation model

Description:

FPA provides standardized method to functionally are the software work product. This work product is the output of software new developmeist and improvement projects for subsequent releases is the software which is relocated to the production application at project implementation. It measures functionality from the users' point of view on the bass of what the user requests and receives in return.

Function Point Analysis (FPA) is a method or set of rules of Functional Size Measurement R assesses the functionality delivered to as unen, based on the user's external view of the functional requirements. It measures the logical view of an application not the physically implemented view or the internal technical view. The Function Point Analysis technique is used to analyses the functionality delivered by software and Unadjusted Function Point (UFP) is the unit of measurement.

Objectives of FPA:

1. The objective of FPA is to measure functionality that the user requests and receives.
2. It should be simple enough to minimize the overhead of the measurement process. It should be a consistent measure among various projects and organizations

Types of FPA:

1. Transactional Functional Type
 - External Input (EI): EI processes data or control information that comes from outside the application's boundary.
 - External Output (EO): EO is an elementary process that generates data or control information sent outside the application's boundary.
 - External inquiries (EQ): EQ is an elementary process made up of an input-output combination that results in data retrieval.

2. Data Functional Type

- Internal Logical File ((LF): A user identifiable group of logically related data or control information maintained within the boundary of the application.
- External Interface Fae (EIF): A group of user recognizable logically related data allusion to the software but maintained within the boundary of another software.

Calculation of Function Point Analysis for Ariane 5 Rocket Launch Software

1. External Input (EI)

- Initial Velocity
- Mass
- Cross section area
- Drag coefficient
- Altitude
- Angle A
- Angle B
- Angle C
- Angle D
- Reference length
- Speed

2. External Output (EO):

- Terminal velocity
- Max height
- Time to Max height
- Terminal velocity
- Altitude
- Speed
- Speed of sound
- Mach

3. External Inquiries (EQ)

- Home page
- Ballastic flight calculator
- Terminal velocity calculator
- Rocket altitude calculator
- Speed of Mach of sound calculator

4. Internal Logical File (ILF):

- Control Access
- File into
- File type
- Backup maintainence
- Input/Output log record

5. External Interface File (IFS)

- User
- User management
- System admin
- Testing admin
- Launch admin

Counting Function Point (FP):

Step-1: $F = 14 \times \text{scale}$

Scale varies from 0 to 5 according to character of Complexity Adjustment Factor (CAF) Below table shows scale.

1. No Influence
2. Incidental
3. Moderate
4. Average
5. Significant
6. Essential

On the Average Case we Estimate Scale as Average Scale=3

$$F = 14 \times 3$$

$$F=42$$

Step 2: Calculate Complexity Adjustment Factor (CAF)

$$\text{CAF} = 0.65 \times (0.01)$$

$$\text{CAF} = 0.65 + (0.01 \times 42)$$

$$\text{CAF} = 0.65 + 0.42$$

$$\text{CAF} = 1.07$$

Step-3: Calculate Unadjusted Function Point (UFP). Multiply each individual function point to corresponding values in TABLE

Function Units	Low	Avg	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Figure 10: FPC

As weighting factors are also average (even in question) hence we will multiply each individual function point to corresponding values in TABLE

User Input=32

User Output=19

User Inquiries=5

User Files=3

External Files=1

$UFP = (32*6) + (19*7) + (5*6) + (3*15) + (1*10)$ $UFP = 192 + 133 + 30 + 45 + 10$

$UFP = 410$

Step 4: Calculate Function Point.

$FP = UFP * CAF$

$FP = 410 * 1.19$

$FP = 487.9$

Function Point Counter

For help, please refer to the right side of the window.

Measurement Parameter	Count		Weighting factor		
number of user inputs	32	X	complex ▾	=	192
number of user outputs	19	X	complex ▾	=	133
number of user inquiries	5	X	complex ▾	=	30
number of files	3	X	complex ▾	=	45
number of external interfaces	1	X	complex ▾	=	10
Total Count					410

Complexity Adjustment Values		
1	Does the system require reliable backup and recovery?	Essential ▾
2	Are data communications required?	Significant ▾
3	Are there distributed processing functions?	Essential ▾
4	Is Performance critical?	Essential ▾
5	Will the system run in an existing heavily utilized operational environment?	Essential ▾
6	Does the system require online data entry?	Average ▾
7	Does the online data entry require the input transaction to be built over multiple screens or operations?	Average ▾
8	Are the master files updated online?	Average ▾
9	Are the inputs, outputs, files, or inquiries complex?	Essential ▾
10	Is the internal processing complex?	Essential ▾
11	Is the code designed to be reusable?	Average ▾
12	Are conversion and installation included in the design?	Significant ▾
13	Is the System designed for multiple installations in different organizations?	Incidental ▾
14	Is the application designed to facilitate change and ease of use by the user?	Average ▾

Total Function Point Count: 487.9 Compute Function Points

Figure 11: Function point

Function point Counter

GANTT CHART

AIM:- To develop a Timeline chart and project table using PERT or CPM project scheduling methods.

DESCRIPTION:-

A Gantt chart is a graphical depiction of a project schedule. It's is a type of bar chart that shows the start and finish dates of several elements of a project that include resources, milestones, tasks, and dependencies. Henry Gantt, an American mechanical engineer, designed the Gantt chart.

How Gantt Charts Work

The Gantt chart is the most widely used chart in project management. These charts are useful in planning a project and defining the sequence of tasks that require completion. In most instances, the chart is displayed as a horizontal bar chart.

Horizontal bars of different lengths represent the project timeline, which can include task sequences, duration, and the start and end dates for each task. The horizontal bar also shows how much of a task requires completion.

[Important: The length of the bar is proportional to the time necessary for a task's completion. The project tasks are represented on the vertical axis.]

A Gantt chart helps in scheduling, managing, and monitoring specific tasks and resources in a project. The chart shows the project timeline, which includes scheduled and completed work over a period. The Gantt chart project manager in communicating project status or plans and also helps ensure the project remains on track.

Benefits of a Gantt Chart

The chart identifies tasks that may be executed in parallel and those that cannot be started or finished until other tasks are complete. The Gantt chart can help detect potentialbottle necks and identify tasks that may have been excluded from the time line.

The chart depicts task slack time or additional time for completion of a task that should not delay the project, noncritical activities that may be

delayed and critical activities that must be executed on time.

Gantt charts can be used in managing projects of all sizes and types. These charts are utilized in several industries and for a range of projects, such as building dams, bridges and highways, software development, and development of other goods and services. Project management tools, such as Microsoft Visio, Project, SharePoint, and Excel or specialized software, such as GanttPro or Matchware, can help in designing Gantt charts.

Historical development

Although now regarded as a common charting technique, Gantt charts were considered revolutionary when first introduced. The first known tool of this type was developed in 1896 by Karol Adamiecki, who called it a harmonogram. Adamiecki did not publish his chart until 1931, however, and only in Polish, which limited both its adoption and recognition of his authorship. construction project. It appears that Schürch's charts were not notable but rather routine in Germany at the time they were published. The prior development leading to Schürch's work is unknown. Unlike later Gantt charts, Schürch's charts did not display interdependencies, leaving them to be inferred by the reader. These were also static representations of a planned schedule.

The chart is named after Henry Gantt (1861-1919), who designed his chart around the years 1910-1915.

One of the first major applications of Gantt charts was by the United States during World War 1, at the instigation. In 1912, Hermann Schürch [de] published what would be considered Gantt charts while discussing a of General William Crozier.

The earliest Gantt charts were drawn on paper and therefore had to be redrawn entirely in order to adjust to schedule changes. For many years, project managers used pieces of paper or blocks for Gantt chart bars so they could be adjusted as needed. Gantt's collaborator Walter Polakov introduced Gantt charts to the Soviet Union in 1929 when he was working for the Supreme Soviet of the National Economy. They were used in developing the First Five Year Plan, supplying Russian translations to explain their use.

In the 1980s, personal computers allowed widespread creation of complex

and elaborate Gantt charts. The first desktop applications were intended mainly for project managers and project schedulers. With the advent of the Internet and increased collaboration over networks at the end of the 1990s, Gantt charts became a common feature of web-based applications, including collaborative groupware. By 2012, almost all Gantt charts were made by software which can easily adjust to schedule changes.

In 1999, Gantt charts were identified as "one of the most widely used management tools for project scheduling and control".

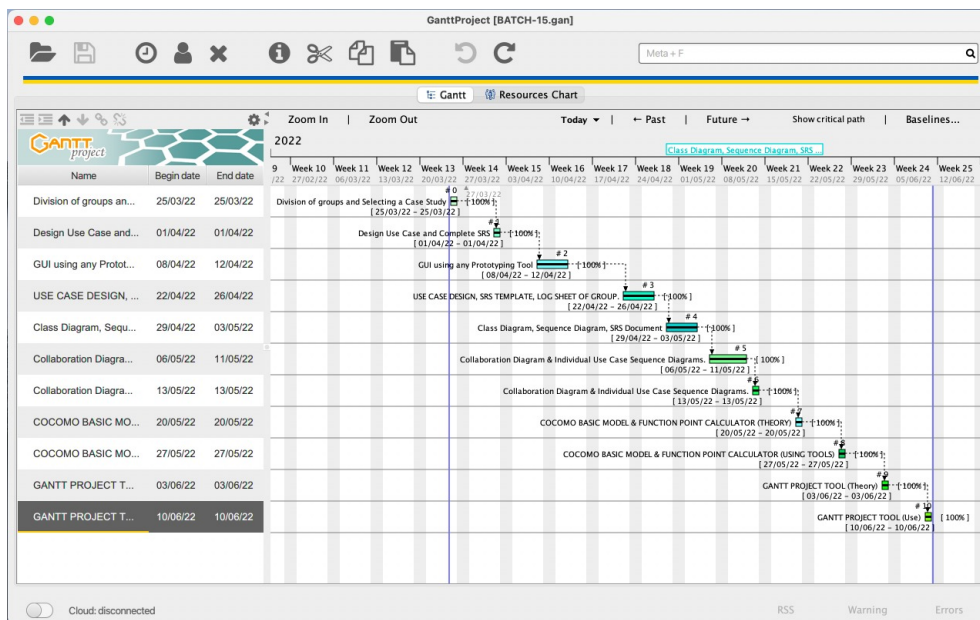


Figure 12: Gantt Chart