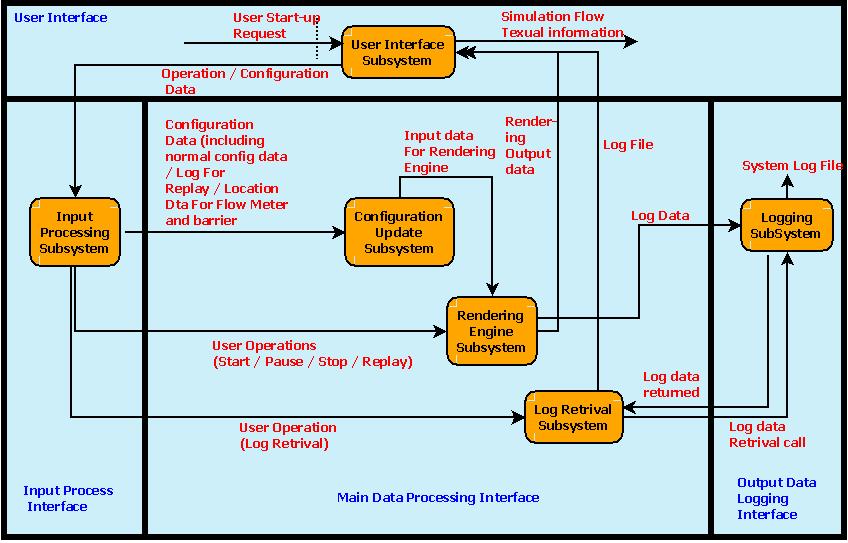
**Answer – 1**

****

**Answer - 2.1**

We are planning to use Iterative Model for our project development.

Compared to Waterfall model Iterative model gives following advantages:

* Results are obtained early and periodically.
* Parallel development can be planned.
* Testing and debugging during smaller iteration is comparatively easy.
* With every increment an operational product is delivered.
* Risk analysis and management is better.

Compared to Spiral model Iterative model gives following advantages:

* Management is less complex.
* End of the project is not the case with spiral model.
* It is applicable to small and moderate complex project.

Few reasons and factors which inclined our decision towards the use of this model over other methodologies in SDLC are:

* Initial product is delivered faster with most important functionality.
* As the development team is a new team with not much project development experience it is better to spread across smaller increments instead of concentrating in one large development model like Waterfall model.
* By use of iterative model important functionality can be provided in initial iterations and cosmetic or enhancements can be provided based on project progress with respect to timeline of the project which is not the case in Waterfall model. Lesson learnt from CS 524 projects from previous year is that we should try to have a working model at all time and iterative model provides that
* As the project timeline is constrained by the Mid-term and Final presentation deadline it is better to follow iterative model to have the working model for both mid-term and final presentation. This cannot be achieved by following Waterfall model where the deliverable would not be available at mid-term.
* Spiral model is not applicable for small project as the process is for more complex systems and can result in a process overhead for small project like this.
* Customer can share their view of the project at every build iteration which is not the case in Waterfall model.

**Answer – 2.2.a**

Attached file.

**Answer – 2.2.b**

Project Effort Distribution:

From the Gantt chart prepared for the project we have derived the following effort distribution for our project:

|  |  |
| --- | --- |
| **Effort** | **Distribution** |
| Requirement | 37.50% |
| Design | 10% |
| Development | 18.75% |
| Testing and Debugging | 17.50% |
| Release preparation time and slippage buffer | 16.25% |

Here Requirement phase distribution includes project related research and understanding, requirement gathering, SRS documentation with review and signing. In total it takes 30 days of the project life cycle which is 37.5% of the total effort. The next major distribution entity is Design phase which takes 8 days and includes architecture design and preparation and updating of Software Architecture document. The design phase takes 10% of the total distribution. The third major distribution is development phase which includes development and coding for different iteration cycles. This distribution has been allocated 15 days out of total 80 days and result in 18.75% of total effort. Testing and debugging phase which is the next major distribution has been allocated 14 days and result in 17.5% of total project effort. Finally 13 days has been distributed across project life cycle for preparing deliverables for the release milestones and also includes milestone slippage buffer. This distribution accounts for 16.25% of total project distribution effort.

**Answer – 3**

COCOMO model is a model used to estimate the effort for the project. For calculation of the effort it takes into account not only the project but also development team and its development environment. The projects are categorized as organic, semidetached or embedded. This project can be considered of type organic because we are developing a well understood application program, the size of the development team is reasonably small, our team members have experience to develop desktop based application and this desktop based application is a data processing program.

Here, we have used Basic COCOMO Model to get approximate estimate for the project parameters.

As per Basic COCOMO Model expression for Effort can be given as:

Effort = a1 х (KLOC)^a2 PM

Where:

a1 is a constant with value 2.4 for projects of type organic

a2 is another constant with value 1.05 for projects of type organic

KLOC here represents number of lines of code in unit of thousand

PM represents Person Month as a measuring unit for effort

By listing estimated lines of codes for each subsystem listed in project architecture diagram we get:

User Interface subsystem: 300 LOC

Input processing subsystem: 200 LOC

Configuration update subsystem: 200 LOC

Rendering Engine subsystem: 800 LOC

Log Retrieval subsystem: 100 LOC

Logging subsystem: 400 LOC

User Interface subsystem handles user interactions with the system. Input processing system validates the input and decides which subsystem the request should be passed to. Configuration update subsystem handles user or replay related configuration setting which is used by rendering engine subsystem to perform the calculations. Log retrieval subsystem works with logging subsystem to provide logging capability.

In total 2000 lines of codes is an estimated code size for the system and hence KLOC can be taken as 2. (KLOC=2)

By replacing KLOC value in formula for calculating effort we get:

Effort = = 2.4 х (2)^1.05 = 4.97 PM