13th February 2016

Project Plan

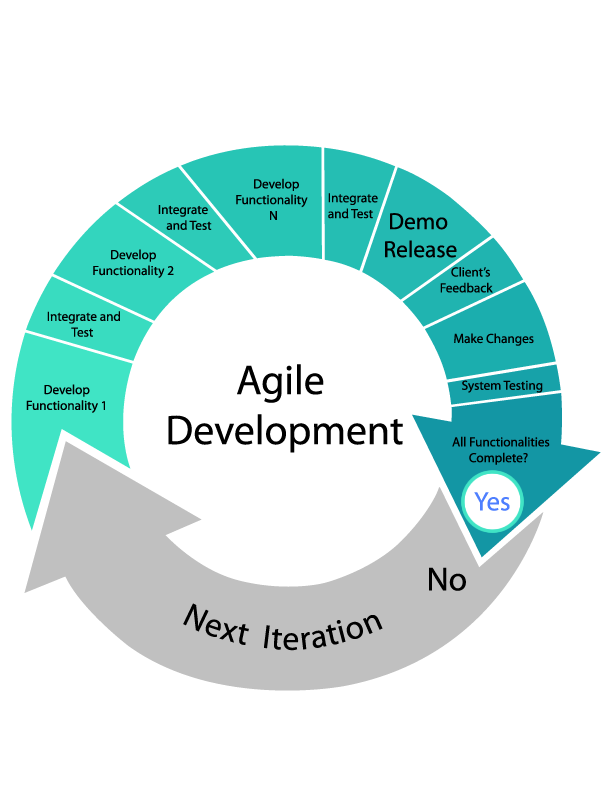
“IoT Enabled Integrated Intelligence System for Automatic Pothole Detection, Pollution Monitoring, Post Accident Medical Response and Breakdown Assistance”

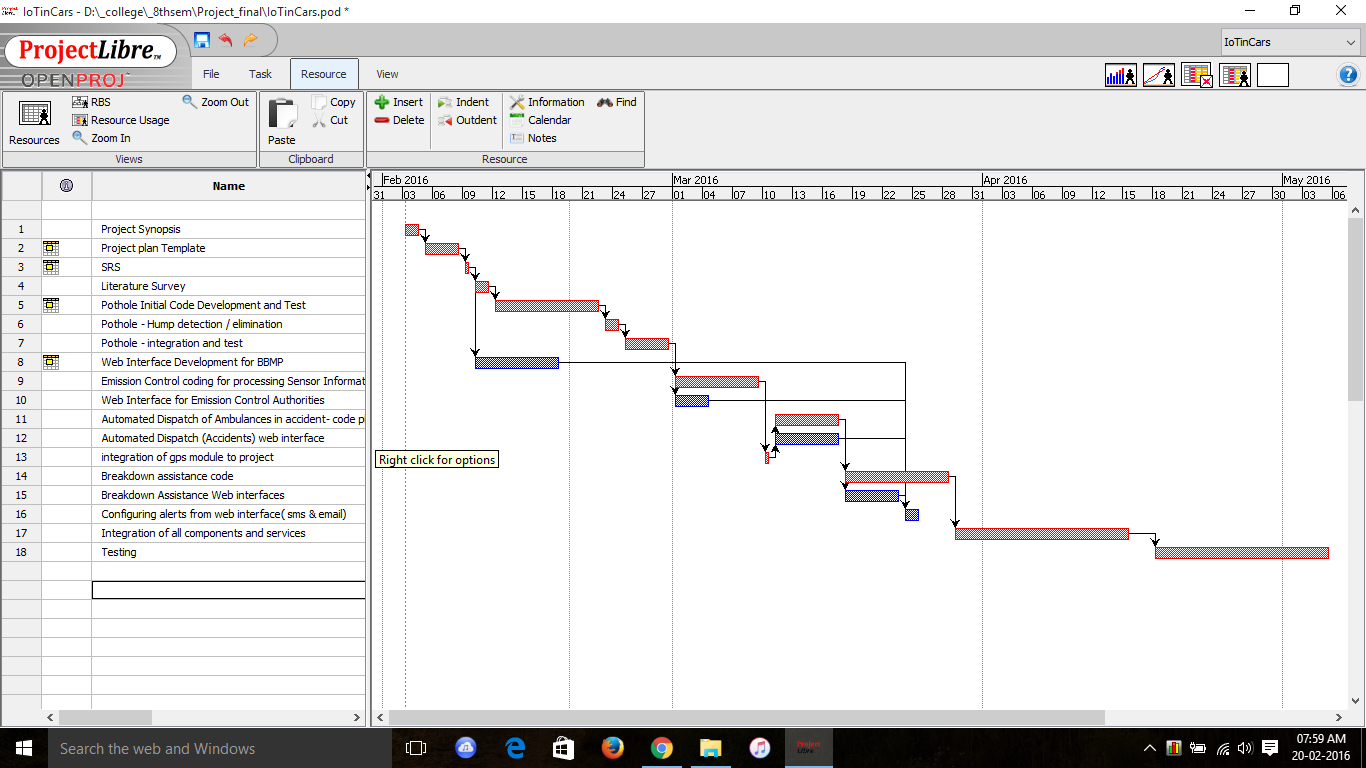
|  |
| --- |
| **Project Guide:**  Prof. Sanjeetha R |
|  |

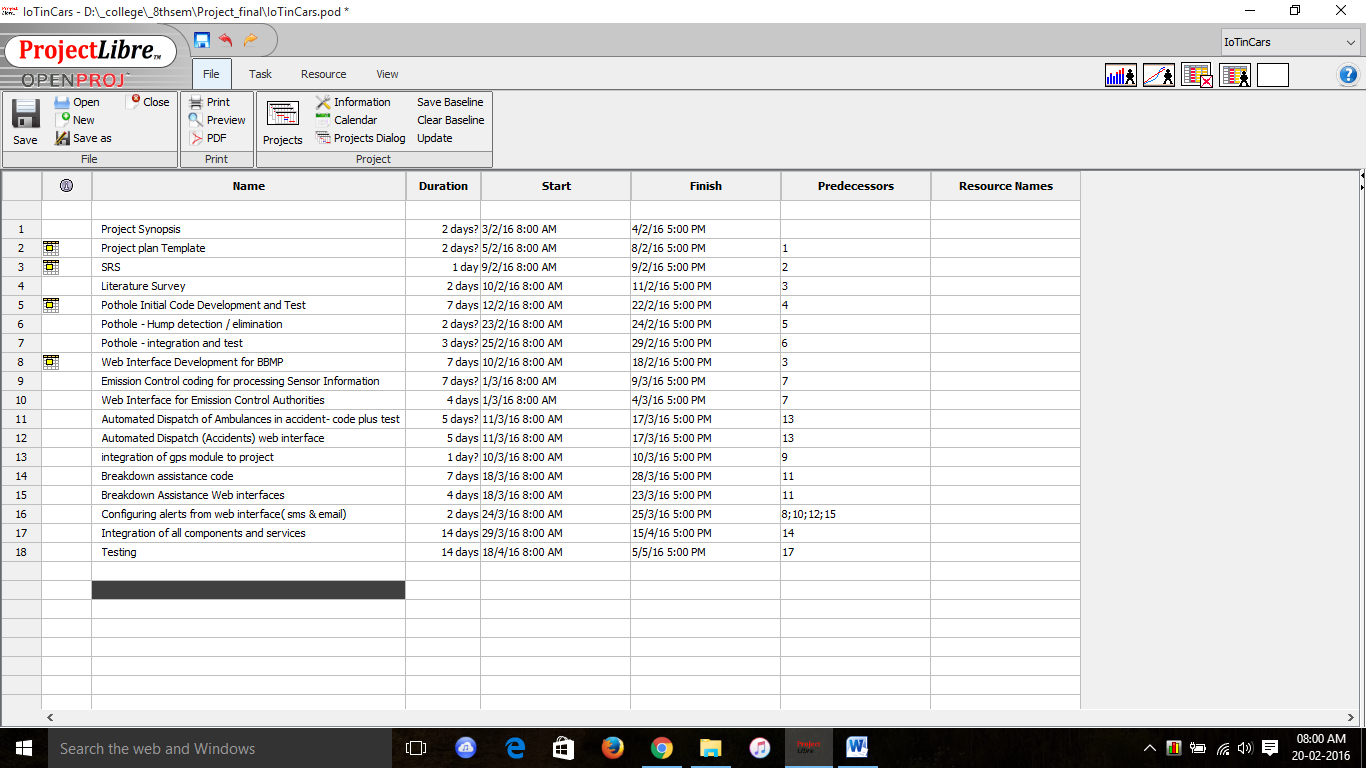
**Team members:**

1. Nikhil Bhat 1MS12CS138
2. Prithvi Alva 1MS12CS077
3. BM Tanvi Raj 1MS12CS137
4. Krunal Bhatt 1MS12CS048

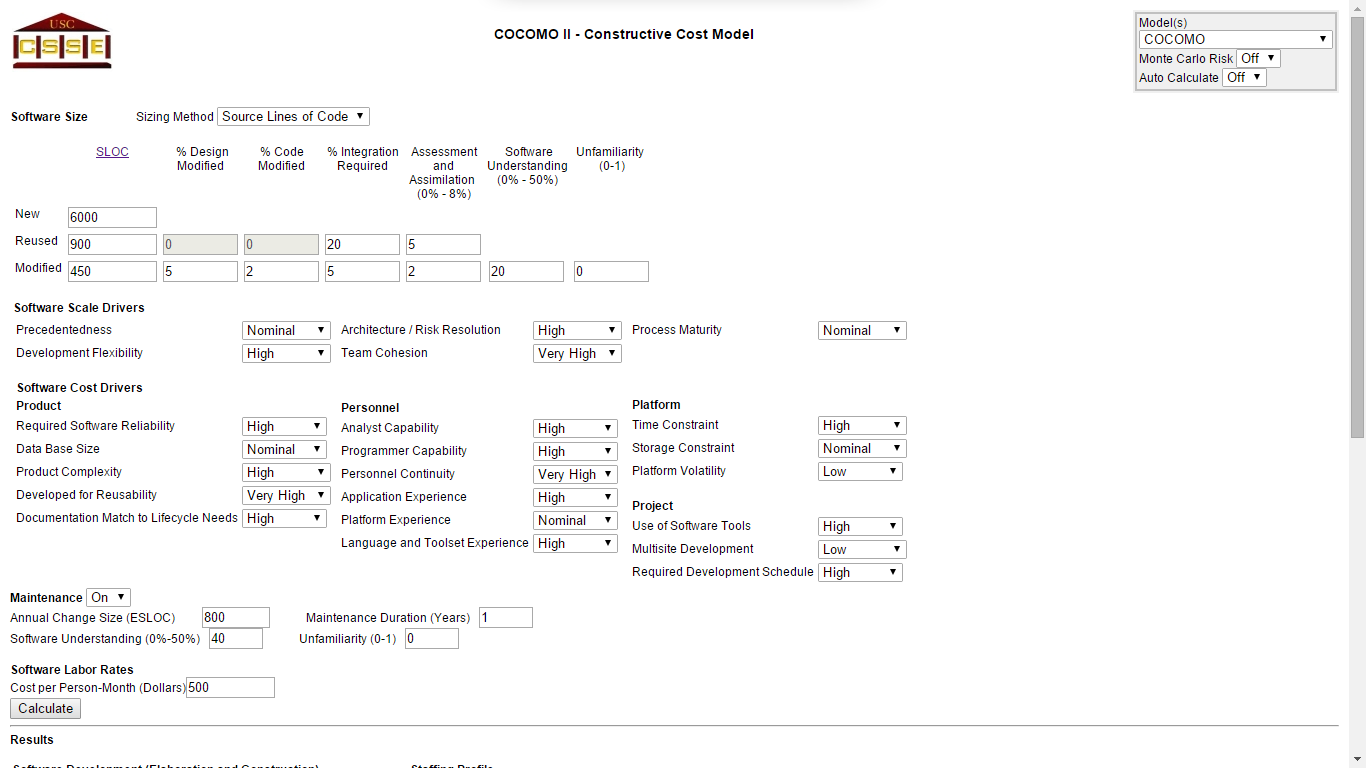
**Process Model**

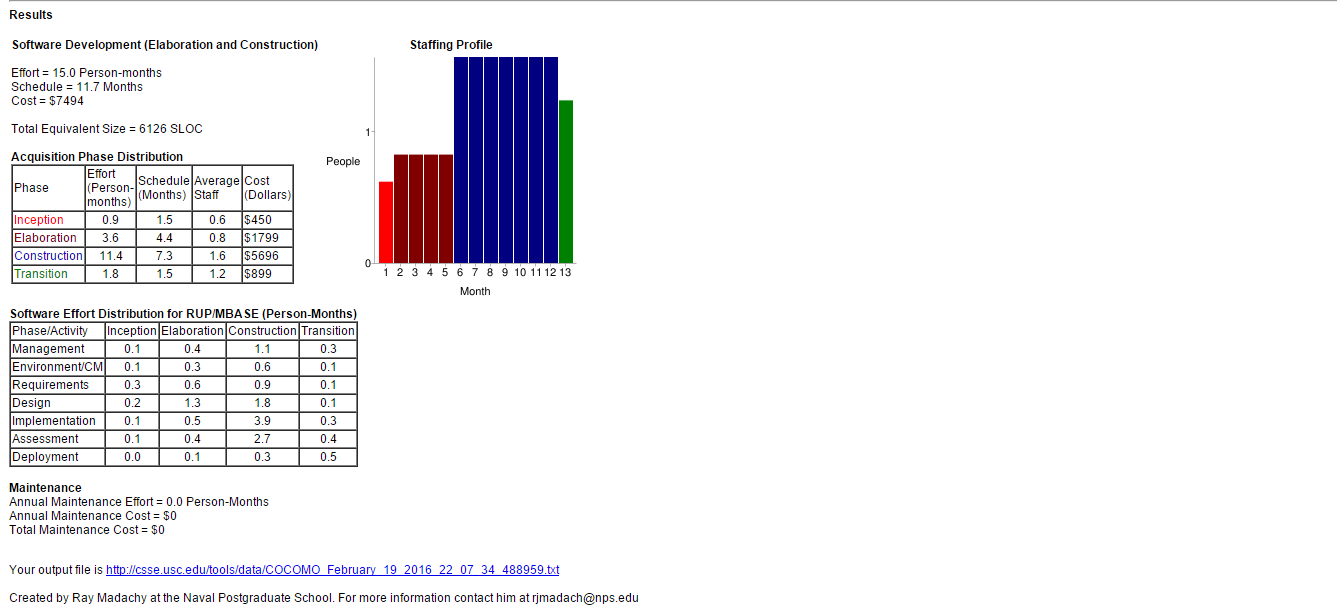
The process model being followed is the Agile development model. 

**Project Schedule - GANTT Chart**



**Effort Estimation – COCOMO Model**

****

****

**Results.txt:**

startCOCOMO, 1

Models, COCOMO

MonteCarlo, MonteCarlo\_Off

AutoCalculate, Off

size\_type, SLOC

new\_size, 6000

reused\_size, 900

IM\_reused, 20

AA\_reused, 5

modified\_size, 450

DM\_modified, 5

CM\_modified, 2

IM\_modified, 5

AA\_modified, 2

SU\_modified, 20

UNFM\_modified, 0

prec, Nominal

flex, High

rely, High

data, Nominal

cplx, High

ruse, Very\_High

docu, High

resl, High

team, Very\_High

acap, High

pcap, High

pcon, Very\_High

apex, High

pexp, Nominal

ltex, High

pmat, Nominal

time, High

stor, Nominal

pvol, Low

tool, High

site, Low

sced, High

software\_maintenance, On

software\_change\_size, 800

software\_maintenance\_duration, 1

maintenance\_software\_understanding, 40

maintenance\_software\_unfamiliarity, 0

software\_labor\_cost\_per\_PM, 500

submit2, Calculate

software\_EAF, 0.75512923785355

software\_effort, 14.988974010821

software\_schedule, 11.657734111047

**Risk Identification & Mitigation Steps**

1. **Unrealistic schedules, overwhelmed by other activities**:  
     
   This risk arises due to the limited time available to work on the project as we are required to study other courses as well, and work on projects in those courses. Schedules may change due to internals or externals in the college. Handling both, the project and the courses may result in a lack of time to finish the project deliverables within the deadline.

**Mitigation Steps:**

* Create a detailed project management plan with the various activities planned, their deadlines and ownership. Ensure there is some slack time, so that any deadline missed due to external factors mentioned above, will not hamper the schedule. Update the project management plan frequently as the activities are completed.
* Define detailed milestones that are to be completed in various stages of the project. Ensure that these milestones are clearly defined and that they are unambiguous. Create a schedule estimation and ensure clarity in the same.
* Develop in incremental steps so that the basic functionality is created, tested and then the additional features follow the same cycle and are integrated one by one, rather than trying to develop all the features once, and failing to meet basic requirements, in the case that schedule is overrun.
* Whenever possible try to reuse code, to avoid duplication of effort, and spend that time in useful work.

1. **Continuing stream of requirements changes:**  
     
   This risk arises as newer requirements could be found as the project progresses. As more modules are produced and added, they may create certain limitations on the components like sensors, thereby causing changes in requirements often.   
     
   **Mitigation steps:**

* Ensure there is a detailed requirements analysis done for the project at the start and that all user expectations are clearly noted down. These requirements must be quantifiable, easily definable and unambiguous.
* Follow incremental development so that if in the case that there are changes, these changes can be deferred to later increments.
* Analyze possible dependencies before starting the work on a requirement. Initially time may have to be spent in speculating these dependencies, but it avoids unpredictable delays when requirement have to be changed.

1. **Personnel shortfall - unavailability of team members:**

This risk is caused by the unavailability of the team members due to possible placement activities, internship commitments, or personal reasons. This highly impacts the project, as there is a limited time frame for completion and there are only 4 people to complete it. Each team member specializes in a skill, so unavailability means development could come to a halt.

**Mitigation Steps:**

* Use knowledge management strategies such as pair programming to understand each other’s work. This ensures that in the case that a team member is not available, there is another team member who can continue that work. This redundancy in knowledge, can avoid any delays due to unavailability of team members.
* Assure files are uploaded and integrated consistently so that even in the case that a team member is unavailable, the source files and resources of the member are still available to the team to continue working.
* In the beginning of the project, determine course schedules, examinations, reviews and plan for unpredictable activities like placements or other unavoidable commitments. Based on these considerations, come up with a common time to work, and plan for when team members aren’t available.

1. **Integration problems:**

This risk arises when different modules are integrated together for a certain feature. Since each team member contributes to feature, insufficient discussion on dependencies could cause problems when different modules are integrated together, which will then mean spending useful time on fixing the code for dependencies related problems.

**Mitigation Steps:**

* Ensure there is proper communication amongst the team members, as each member is responsible for their part of work, and only they can describe the dependencies in their work. Set up a group Web page, group e-mail accounts, use instant messaging, meet regularly
* Integrate more frequently, even if they are smaller modules, as it is easier to fix errors. This will save useful time when compared to doing only a single integration, as debugging a problem in such a situation is like finding a needle in a haystack. This could cause unprecedented delays, that in a limited time like this, will be difficult to solve.
* Ensure that there is a copy of all previous builds and source codes before the next integration is done. In the event that the integration causes an irreversible change, the team should be able to come up with another build from previous files.
* Plan to ensure that possible dependencies problems are solved before integration as this ensures clarity.

1. **Overriding other people’s work, not having the latest versions of code:**

This risk could arise when a team member uses code from another team member, which is an older version, or the team member modifies another member’s code. The source code control is vital, as overriding working code, could mean original code being lost, thereby spending potentially useful time in debugging and duplication of coding.

**Mitigation Steps:**

* Use version control/source code control tools which track and provide control over changes to source code so that team members always have access to the latest code from the owner. Tools include OpenCVS, Git.
* Ensure that at all times there are backups of the previous builds, done frequently, so that in the event of something going wrong with the current version, one can revert back to the previous working version, rather than having to delete the newer code that was written.
* Communicate effectively, set up a group Web page, group e-mail accounts, use instant messaging.