M S RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute affiliated to VTU, Belgaum)

MSR Nagar, Bangalore-560054

****

January to June 2016

Department of Computer Science & Engineering

**PROJECT PLAN**

**ON   
REDUCING ACCIDENTS USING TRAFFIC DATA ANALYSIS AND DETECTION**

Submitted by

**ABHINAV SHIKHAR 1MS12CS002  
NAVEEN JS 1MS13CS416**

**SANDEEP KUMAR 1MS12CS097**

**T HEMA 1MS12CS139**

Under the guidance of

**Sowmya BJ**

**Assistant Professor, MSRIT**

**PROCESS MODEL: V MODEL**

V- Model means Verification and Validation model. Just like the [waterfall model](http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/), the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins.  Testing of the product is planned in parallel with a corresponding phase of development.



The various phases of the V-model are as follows:

1. **Requirements** like BRS and SRS begin the life cycle model just like the waterfall model. But, in this model before development is started, a [system test](http://istqbexamcertification.com/what-is-system-testing/) plan is created.  The test plan focuses on meeting the functionality specified in the requirements gathering.
2. **The high-level design (HLD)** phase focuses on system architecture and design. It provides overview of solution, platform, system, product and service/process. An integration plan is created in this phase as well in order to test the pieces of the software systems ability to work together.
3. **The low-level design** **(LLD)** phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. Component tests are created in this phase as well.
4. **The implementation** phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.
5. **Coding:** This is at the bottom of the V-Shape model. Module design is converted into code by developers.

**Advantages of V-model:**

* Simple and easy to use.
* Testing activities like planning, [test designing](http://istqbexamcertification.com/what-is-test-design-or-how-to-specify-test-cases/) happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
* Proactive defect tracking – that is defects are found at early stage.
* Avoids the downward flow of the defects.
* Works well for small projects where requirements are easily understood.

**Disadvantages of V-model:**

* Very rigid and least flexible.
* Software is developed during the implementation phase, so no early prototypes of the software are produced.
* If any changes happen in midway, then the test documents along with requirement documents has to be updated.

**Why V-model:**

* The V-shaped model should be used for small to medium sized projects
* Requirements are clearly defined and fixed.
* The V-Shaped model should be chosen when ample technical resources are available.

**PROJECT SCHEDULING: PROJECT LIBRE TOOL**

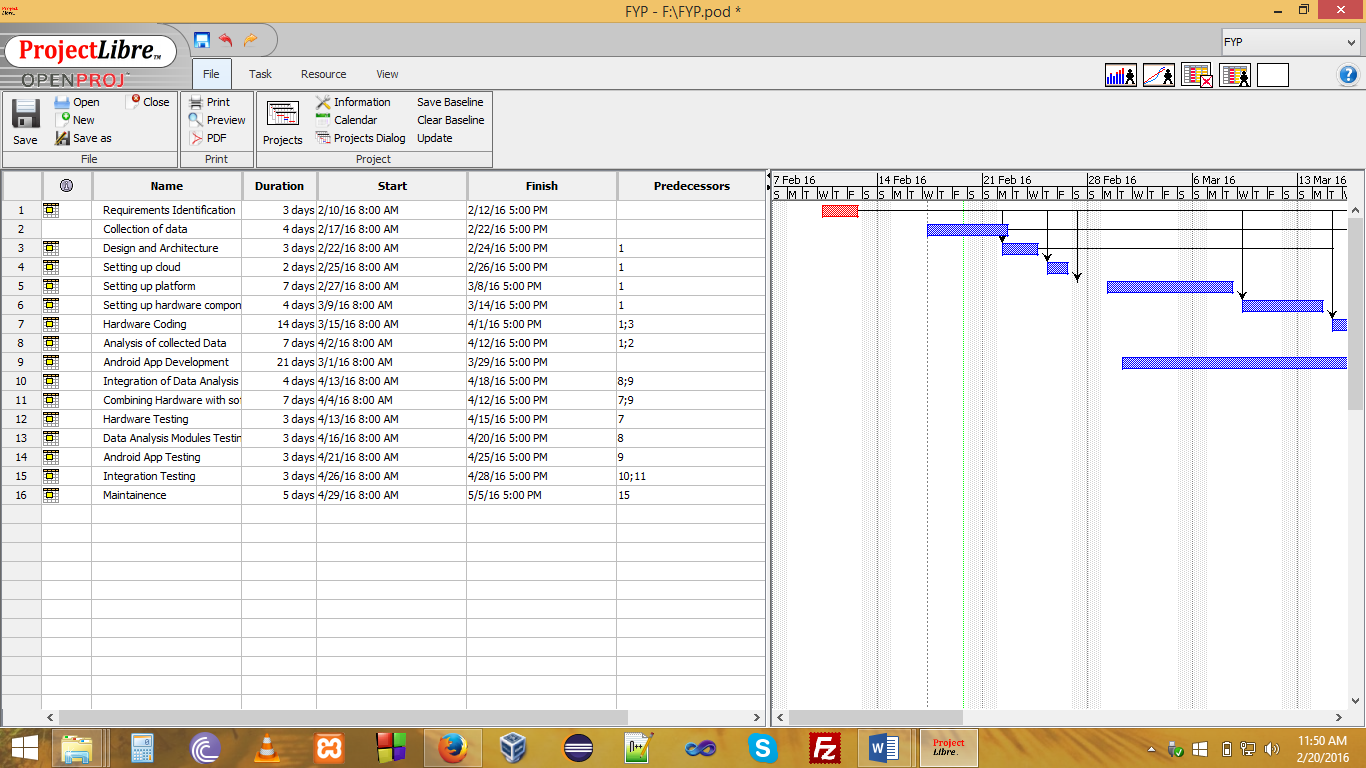
The project schedule is the tool that communicates *what work* needs to be performed, *which resources* of the organization will perform the work and the *timefram*es in which that work needs to be performed. The project schedule should reflect all of the work associated with delivering the project on time.

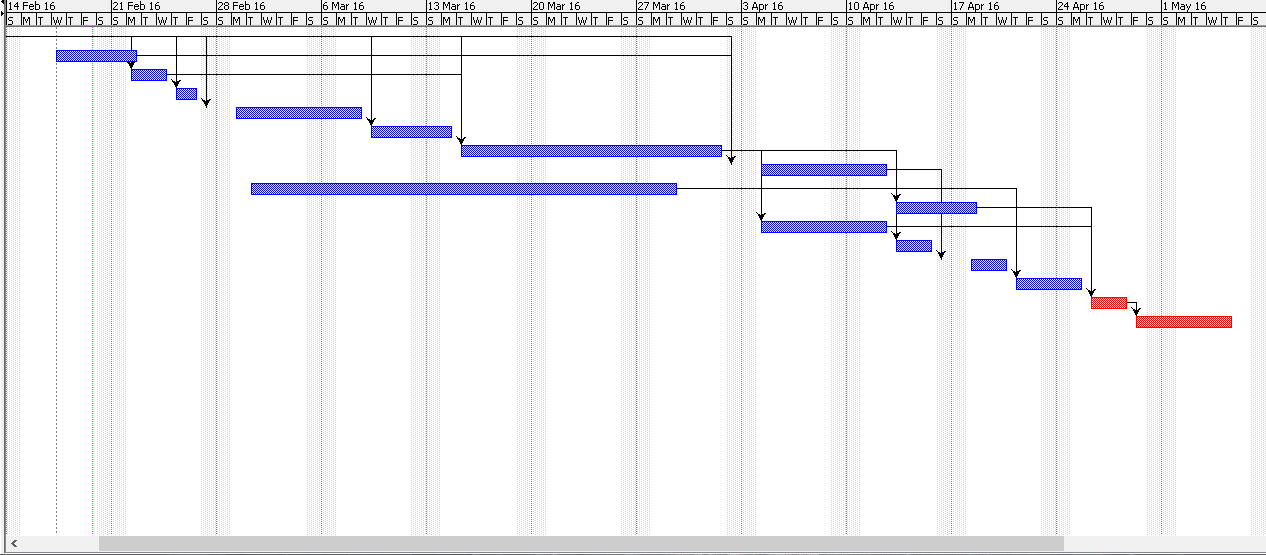
**Project Libre:**

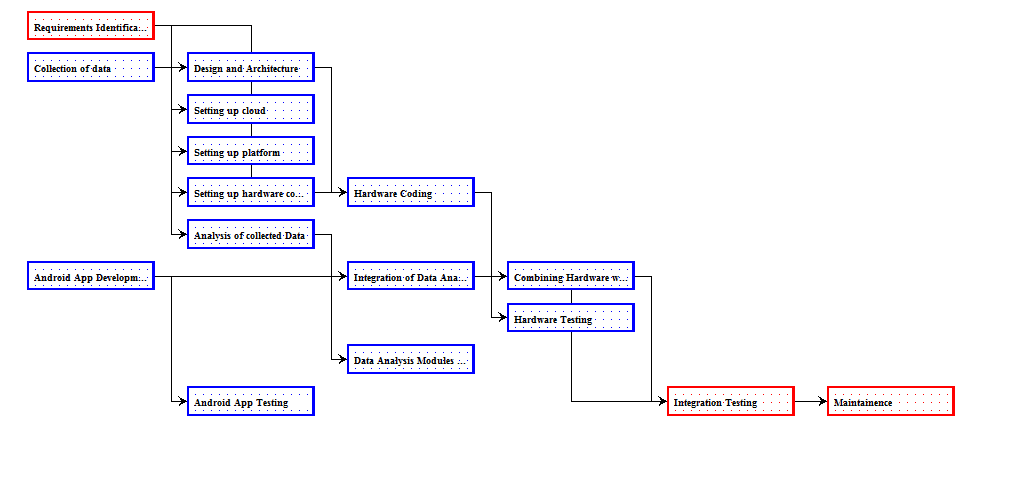
ProjectLibre is a [project management software](https://en.wikipedia.org/wiki/Project_management_software) system. ProjectLibre runs on the [Java Platform](https://en.wikipedia.org/wiki/Java_Platform), allowing it to run on the [Linux](https://en.wikipedia.org/wiki/Linux), [Mac OS](https://en.wikipedia.org/wiki/Mac_OS) or [MS Windows](https://en.wikipedia.org/wiki/MS_Windows) [operating systems](https://en.wikipedia.org/wiki/Operating_system). It is released under the [Common Public Attribution License](https://en.wikipedia.org/wiki/Common_Public_Attribution_License) (CPAL) and thus qualifies as [Free Software](https://en.wikipedia.org/wiki/Free_Software) according to the [Free Software Foundation](https://en.wikipedia.org/wiki/Free_Software_Foundation).

**Gantt Chart:**

Gantt charts allow us to visualize a project’s start and end date along with each element or task that needs to be completed. Because many project tasks are dependent on previous tasks, these charts also allow us to see these dependencies and schedule around them.



****

****

**Advantages:**

1. Allows for efficient organization
2. Helps establish timeframes.
3. Highly Visual

**Disadvantages:**

1. Potentially overly complex – If you’ve ever worked on a complex project, and looked at the Gantt chart, you know that these charts can be large and hard to read.
2. Need to be updated – Gantt charts are developed early in the planning stages of a project, there is a good chance that the project will change, thus the chart will need to be updated.
3. Don’t show the whole picture.

**EFFORT ESTIMATION: COCOMO Model**

[Software development](https://en.wikipedia.org/wiki/Software_development) *effort* [*estimation*](https://en.wikipedia.org/wiki/Estimation) is the process of predicting the most realistic amount of effort (expressed in terms of person-hours or money) required to develop or maintain [software](https://en.wikipedia.org/wiki/Software) based on incomplete, uncertain and noisy input. Effort estimates may be used as input to project plans, iteration plans, budgets and investment analyses, pricing processes and bidding rounds.

**COCOMO Model:**

The Constructive Cost Model (COCOMO) is an algorithmic [software cost estimation model](https://en.wikipedia.org/wiki/Estimation_in_software_engineering) developed by [Barry W. Boehm](https://en.wikipedia.org/wiki/Barry_Boehm). It makes use of various parameters and a defined formula to estimate effort. It is a *Parametric Model* which accepts as input quantative and qualitative weighted characteristics and produces effort estimation.

COnstructive COst MOdel II (COCOMO II) is the latest major extension to the original COCOMO (COCOMO 81) model published in 1981.

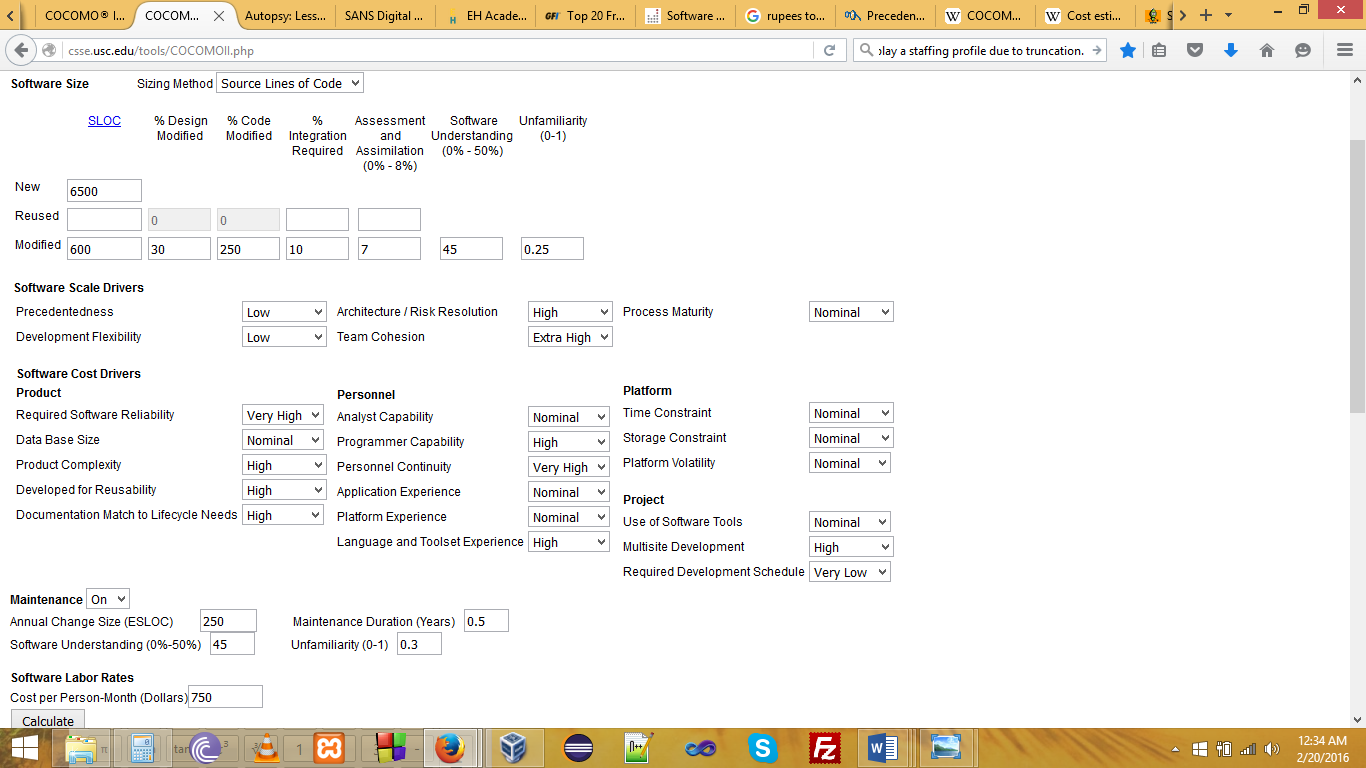


Figure: Input to COCOMO Effort Estimation Tool

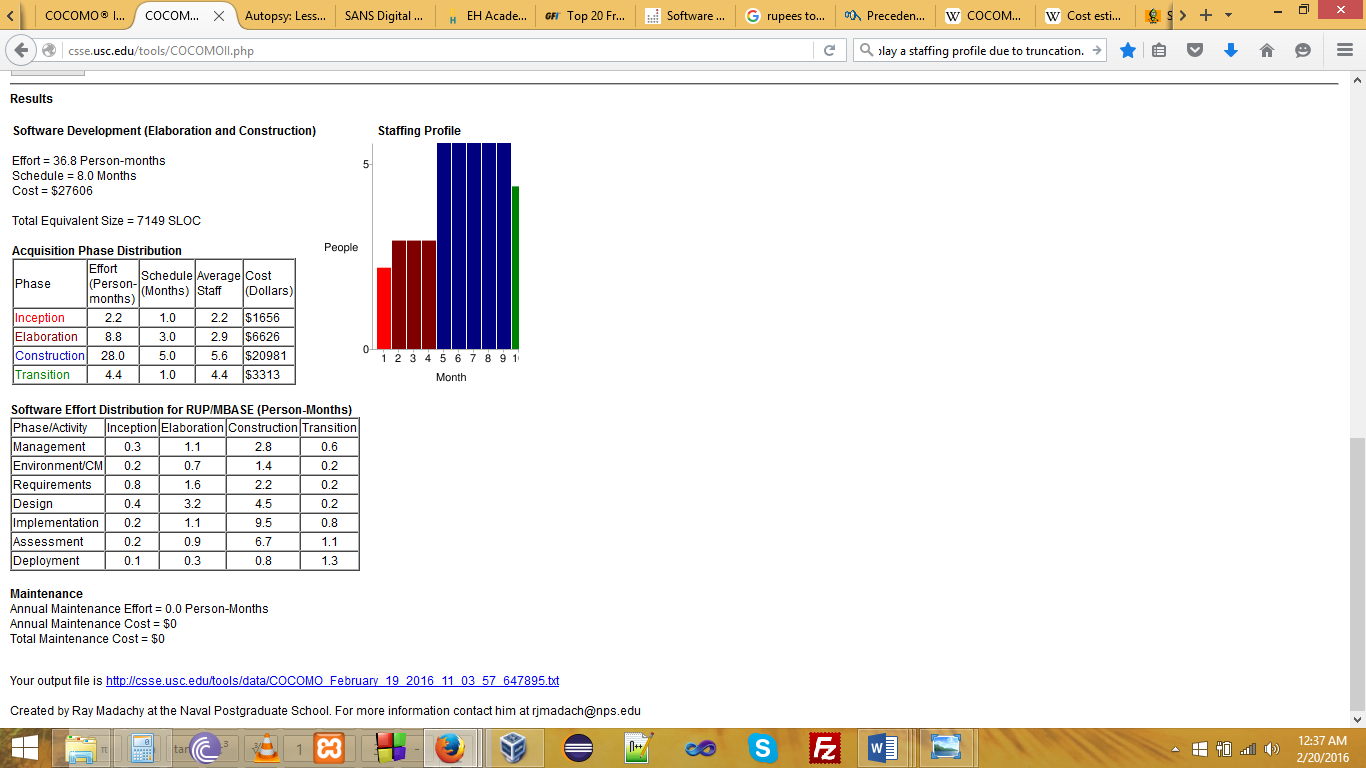


Figure: Output of COCOMO Model

**Advantages:**

1. COCOMO is factual and easy to interpret. One can clearly understand how it works.
2. Accounts for various factors that affect cost of the project.
3. Works on historical data and hence is more predictable and accurate.
4. The drivers are very helpful to understand the impact on the different factors that affect the project costs.

**Drawbacks:**

1. COCOMO model ignores requirements and all documentation.
2. It oversimplifies the impact of safety/security aspects.
3. It ignores hardware issues
4. It is dependent on the amount of time spent in each phase.

**RISK ASSESSMENT: USING RISK MATRIX 220**

**What Is Risk?**

Risks are events or occurrences that prevent a program from achieving its cost, schedule, or performance objectives.

**Baseline Risk Assessment Process**

A good risk assessment and management process is essential to the success of any program. The process summarized here consists of

* Planning for Risk Assessment
* Identifying Program Objectives or Requirements
* Defining Program Risks
* Ranking Program Risks
* Managing Program Risks
* Managing Action Plans
* Continuously Assessing Program Risks.

**Planning for Risk Assessment:**

The first stage of a risk assessment is to plan the activity. We begin this process by selecting the risk assessment team, setting forth ground rules, and determining the supporting risk management tools, such as the Risk Matrix application. The risk assessment team should include representatives from all areas of the program, not just technical experts. In addition, a facilitator and recorder should be selected to assist the team. The Risk Matrix application can be used at any time.

So in our project we are using the Risk Matrix application to assess the risks involved and to identify their type, probability of occurrence, impact and mitigation plan.

**Identifying Program Objectives or Requirements:**

Once we have identified the risk assessment team and tools, the team

Identifies the key program objectives or requirements. The program objectives and

Requirements should assist the team in identifying risks.

**Defining Program Risks:**

A team facilitator leads the team through a structured brainstorming process to identify the Program risks. For example, each team member individually writes down risk ideas. Next, the facilitator asks each person to present one idea in sequence or pass in a rotation until all candidate risks are offered. Once all ideas are heard, an affinity diagram is created to Group, merge, and eliminate duplicate risks, and to identify dependent risks.

Following the identification of the risks, we assign various attributes to each risk. At a minimum, relevant time frame, impact, and probability of occurrence are assigned. Time frame is the beginning and end dates of when a risk may occur.

Then the team sets impact definitions. The Risk Matrix impact definitions are:

**C (Critical):** If the risk event occurs, the program will fail. Minimum acceptable requirements will not be met.

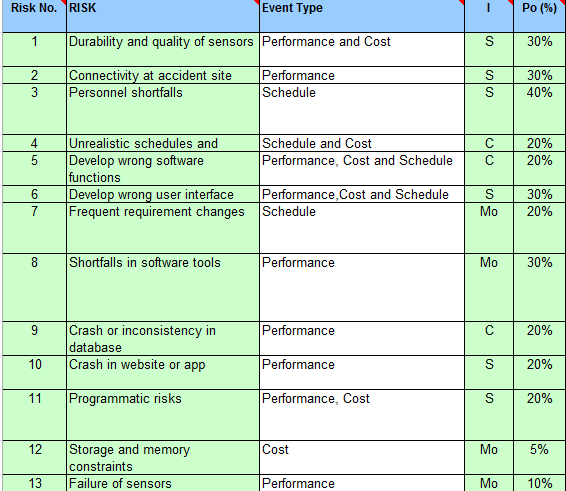
* **S (Serious):** If the risk event occurs, the program will encounter major cost/schedule increases. Minimum acceptable requirements will be met. Secondary requirements may not be met.
* **Mo (Moderate):** If the risk event occurs, the program will encounter moderate cost/schedule increases. Minimum acceptable requirements will be met. Some secondary requirements may not be met.
* **Mi (Minor):** If the risk event occurs, the program will encounter small cost/schedule increases. Minimum acceptable requirements will be met. Most secondary requirements will be met.
* **N (Negligible):** If the risk event occurs, it will have no effect on the program. All requirements will be met.

Probability of occurrence is the team’s assessment of the likelihood of a risk happening.

For this purpose, we have estimated probabilities using a relative scale:

* **0-10%:** very unlikely the risk will occur
* **11-40%:** unlikely the risk will occur
* **41-60%**: even likelihood the risk will occur
* **61-90%**: likely the risk will occur
* **91-100%**: very likely the risk will occur.

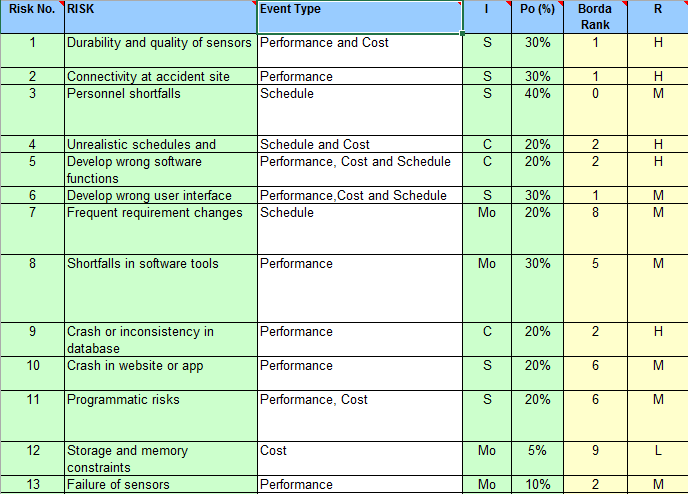
We have identified the various risks, their impact and probability of occurrence and depicted in the risk matrix.



**Ranking Program Risks**

After the above processes, we have all the information needed to rank the risks. Since we are using the Risk Matrix tool, this process becomes simple and automated. Risk rankings are calculated using the Borda voting method. The Borda method *ranks* risks from most-to-least critical on the basis of multiple evaluation criteria.

We have identified the various risks, their rankings and ratings depicted in the risk matrix.



**Managing Program Risks:**

All risks need some form of management, whether it involves a plan for handling risks (action plan) or merely keeping watch.

After the risks are ranked, we identify the risks that are high priority, need to be managed, and require resources (we identify the top *N* risks).

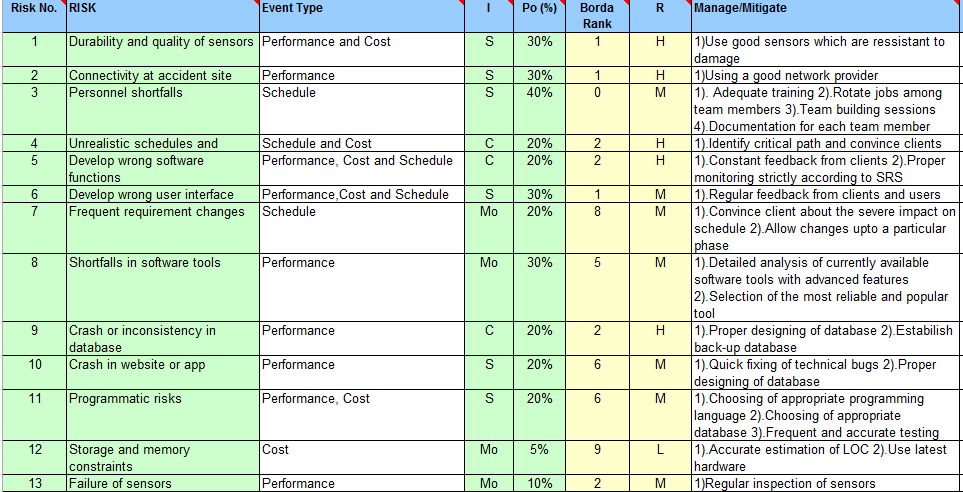
Decisions for handling the top *N* risks will vary. Some risks will be eliminated because the requirements changed; others will be transferred to other organizations (like a contractor) for action because the program team does not have the proper resources to handle the risks or because it's more appropriate for the other organization to handle them; and others will require mitigation strategies. The remaining risks (the ones not in the top *N*) should be watched.

**Managing Action Plans:**

We develop detailed action plans and enter an initial status, assign the Office of Primary Responsibility (OPR), and determine exit criteria for the top *N* risks. The status of each action plan should be reviewed and assessed periodically (approximately each week or month if possible), and the risks rankings adjusted accordingly.

The Risk Matrix tool provides an Action Plan worksheet in the Advanced Mode for tracking risk action plans and adjusting the risk rankings based on the action plan status.

We have identified the various risks manage and mitigation plans in risk matrix.



**Continuously Assessing Risks**

We have to continuously assess risks is essential to develop a good program risk management. As a program progresses, risks should be reassessed periodically (perhaps monthly or quarterly) to determine whether their level of importance has changed or whether new risks have developed that should be identified, assessed, ranked, and managed.