Finding the Root Cause of defects

by B.C. Sekar
HCL Technologies Ltd.,
Chennai.

Organization

1. Abstract

It is not uncommon for software companies to find after a product release, that the customer feels that the product's quality does not meet his/her expectations. Companies may address this issue after receiving customer feedback by analyzing and fixing what went wrong. However, this issue may also be addressed pro-actively by improving product life cycle processes, thereby improving the quality of the product. Any attempt to improve product lifecycle processes will require answers to the following questions.

- What are the bottlenecks in the software lifecycle?
- Which process steps have to be improved?

There might be more specific questions that also need to be addressed.

- How effective is the development?

- How effective is the testing?
- What types of defects affect quality?

All these issues can be addressed by identifying the root cause of these problems. The objective of this presentation is to explore how to find the root cause of defects with the help of a tool, that we propose, which automates analysis.

By definition, root cause is the reason, which, if eliminated or corrected, would have prevented a problem from existing, or occurring. It is not a restatement of the most obvious symptom, but the result of a systematic analysis of the problem, leading to the principal cause. The knowledge gained due to such an analysis could be used for a variety of activities like project management, improving customer satisfaction, quality control, improving software life cycle processes, improve productivity, measure test effectiveness etc.

The tool uses the defect stream as a source of information, to measure and analyze the product and the process. The user has to fill in some additional fields for this purpose. These fields deal with the impact of the defect on the customer, the event that caused the defect, origin of the defect and type of defect. This information is used for finding the root cause of defects.

Some of the existing systems used to find the root cause of defects involves manual collection, processing of defects and the additional fields mentioned above. The disadvantage of this system is that the engineers who are involved in development and testing may have to be involved in this process, which could be tedious if the defects are voluminous.

Some other implementations may tie the finding of root cause to that of the defect-tracking tool. So, there could be rules in the defect entry screen, which expects values for fields like impact, origin etc., be compulsorily entered for every defect entry. Entering information in these fields will take approximately 2 minutes per defect. We may be entering the additional fields like impact, origin for defects for which we may not be interested in finding root cause.

The tool we propose automate processing to find the finding of root cause and is independent of the defect tracking system. It has the flexibility to allow entry of additional fields only, for the defects for which we need to find root cause. Hence it overcomes the limitations of the other approaches mentioned above.

2. Important additional fields.

The fields in a typical defect entry include description, submitter, severity etc., These information may not be sufficient for finding root cause of defects. One may have to drill deep into the descriptions of the bugs to make any meaning out of it. Hence for finding root cause of defects, 5 additional fields are used along with other fields in a defect entry. The five additional fields are impact, trigger, origin, type and reason.

2.1 Impact

Deals with how the customer is affected by a defect. The submitter of the defect enters the value for this field. Some of the possible values are maintainability, performance, scalability, security etc.,

2.2 Trigger

This is the specific activity that caused the defect to surface. It may be a test or an event. The submitter of the defect enters the value for this field. Some of the possible values are simple function, Workload/stress, side effects etc.,

2.3 Origin

The defect origin is the development activity or deliverable where the defect originated. The developer fixing the defect enters the value for this field. Some of the possible values are requirements, design, bad code fix etc.,

2.4 Category

Deals with the nature of the defect and what needs to be corrected. The developer fixing the defect enters the value for this field. Some of the possible values are feature, algorithm, initialization, error handling etc.,

2.5 Reason

This is a modifier of category. It spells out the sort of correction needed. Some of the possible values are missing, wrong, extra. When combined with category it gives more meaning. For example, wrong algorithm.

3. A tool for Finding Root Cause:

We propose a web-based tool for finding root cause of defects. It interfaces with the defect tracking system. The tool will allow input and modification of the fields required for finding root cause. It also helps in generation of root cause reports, which match a query.

The input and modification of the additional fields from this tool will update the defect tracking database for a defect. The generation of reports is automated. The automation is extremely useful, because the person who generates the reports to find root cause need not be a test engineer or a developer. Also the time to find out the root cause of defects is lesser when compared to manual processing of defects to find root cause. After the user submits a query for which the root cause has to be determined, the tool retrieves the defects from the defect-tracking tool and generates a presentable output. The user of this tool is relieved of any manual effort for finding root cause after entering information in the root cause fields. Also, the reports will present the findings in output formats like pie charts and correlation tables. The report will clearly depict what went wrong, what needs improvement etc.,

The tool being web based would scale up well even when those involved in a product life cycle are geographically distributed. For example., the testing team is in City X, the development team in City Y and the manager who needs the root cause reports in City Z.

4. Architecture of the Tool

The Tool consists of the following modules.

*. Input and Modification of Field values

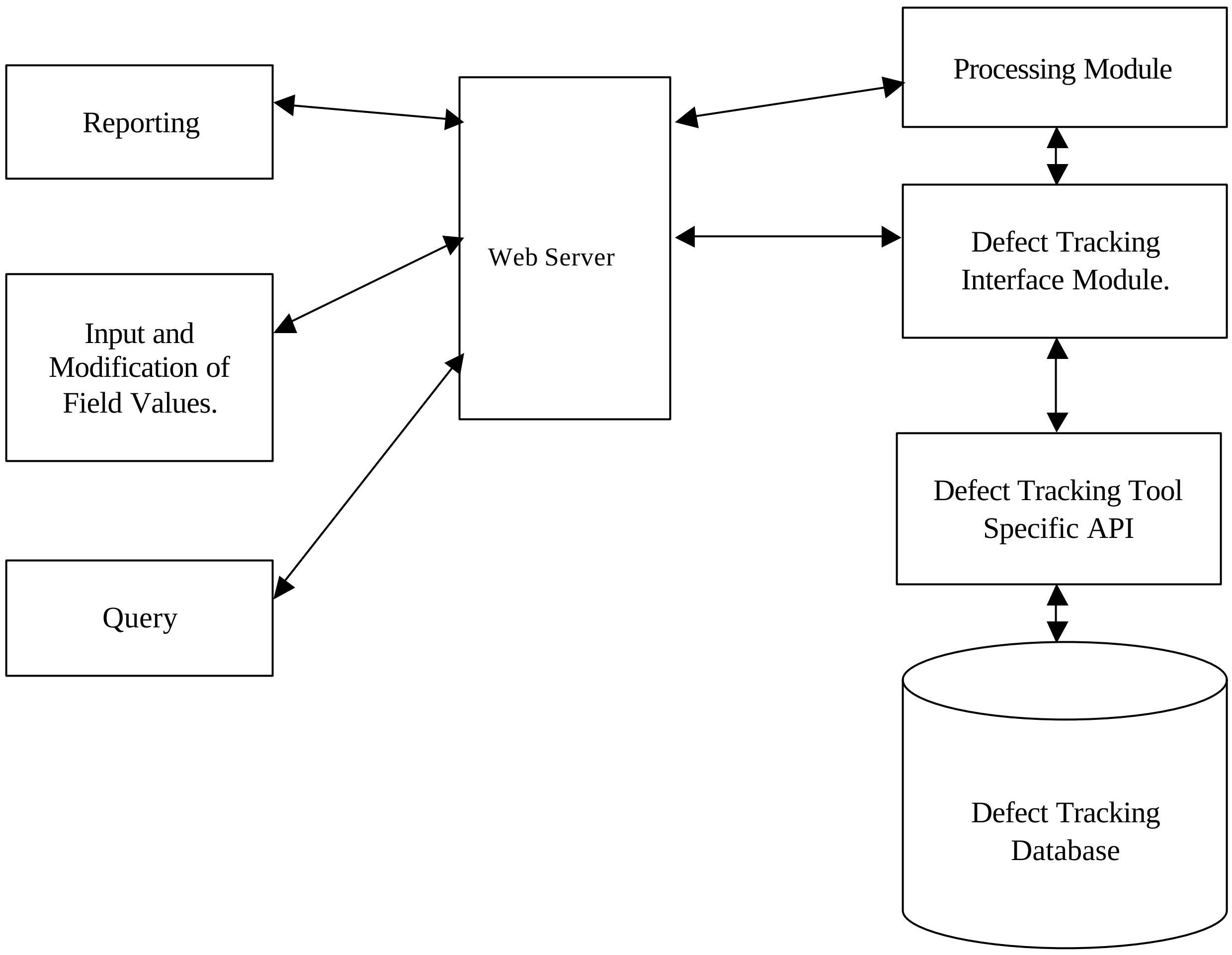
*. Query

*. Interface to Defect Tracking System

*. Processing

*. Reporting

The defect tracking database is part of the defect-tracking tool. Moreover the database is accessed by using the defect tracking tool specific APIs. A web server enables usersto access the tool from browsers.

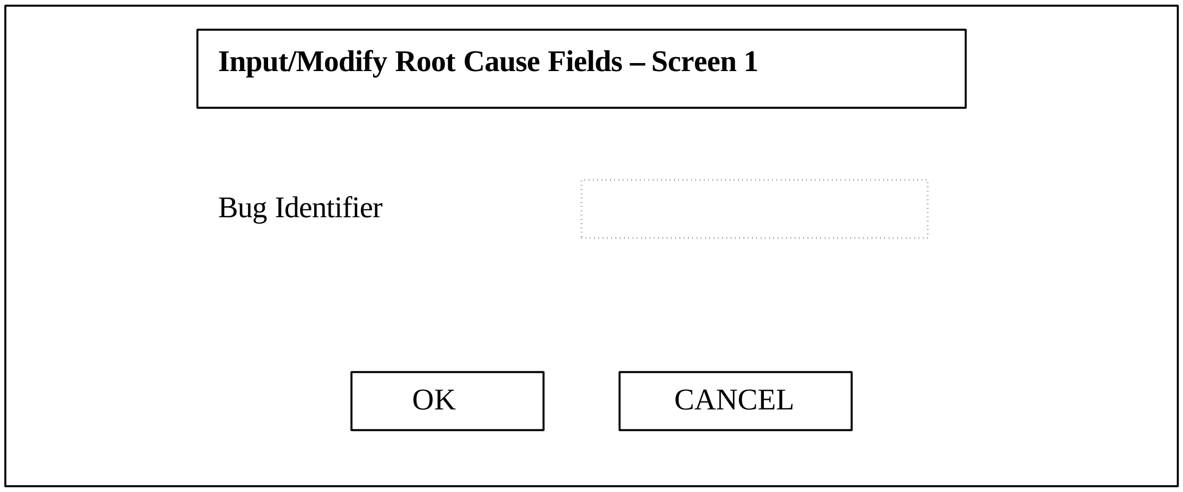

Reporting
Input and Modification of Field Values.
Query
Web Server
Processing Module
Defect Tracking Interface Module.
Defect Tracking Tool Specific API
Defect Tracking Database

5. Input and Modification of Field Values

The additional fields like impact, event, origin and type of defect could have information entered in them from the screen for bug submission and modification that the defect tracking tools provide. This is possible by the extensible features of the bug tracking tools.

Since finding root cause is not a regular activity like dealing with bugs, another approach is to design a separate screen, using which the user could get input/modify the extra fields. Our tool uses this approach. This screen would be part of our web-based tool.

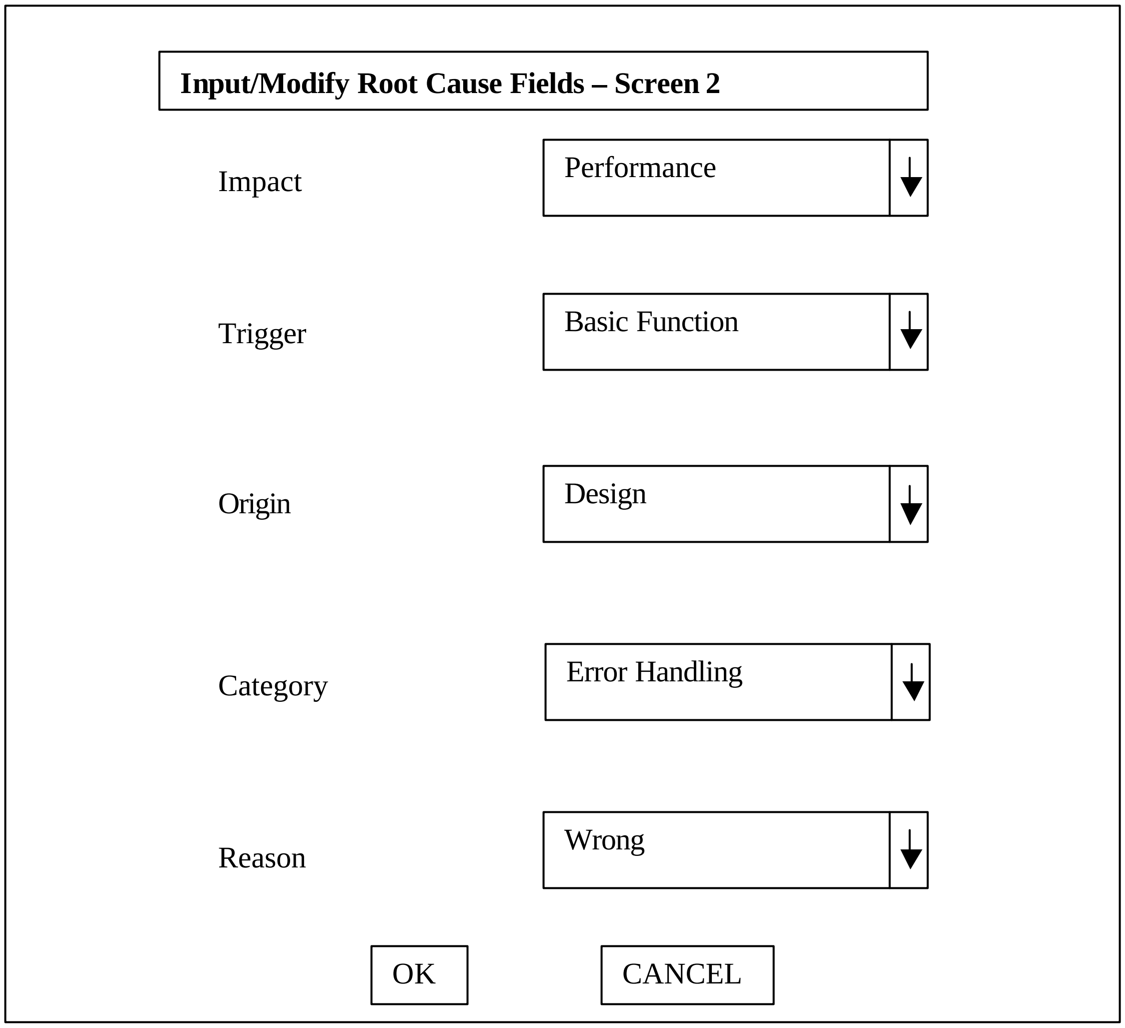
One advantage of having the input screen also as part of our tool is that it will provide flexibility to the user, so that only for the bugs for which root cause has to be found, we can enter the values for the additional fields.


Input/Modify Root Cause Fields – Screen 1

After the user types in the bug identifier, another screen is displayed which allows input and modification of the values of the fields required for finding root cause of defects.

The values for impact, trigger, origin, category and reason are listed in the corresponding combo boxes. Already selected values for the fields are displayed to the user.

After the user selects the values for the fields, the values for the fields are updated in the defect database.



Input/Modify Root Cause Fields – Screen 2

Impact: Performance

Trigger: Basic Function

Origin: Design

Category: Error Handling

Reason: Wrong

6. Query

The additional fields can be queried with other regular defect tracking system fields.

For example., you can specify a query which queries root cause fields of the bugs submitted from <start date> to <end date>, or the state of bugs in New state which have impact as "Maintainability". A screen could be provided for this. This screen is not dealt in detail in this document, as it does not really contribute much for finding the Root cause of defects, but is used to view the additional fields. But it is mentioned, as it is a component of the tool.

7. Interface to Defect Tracking System

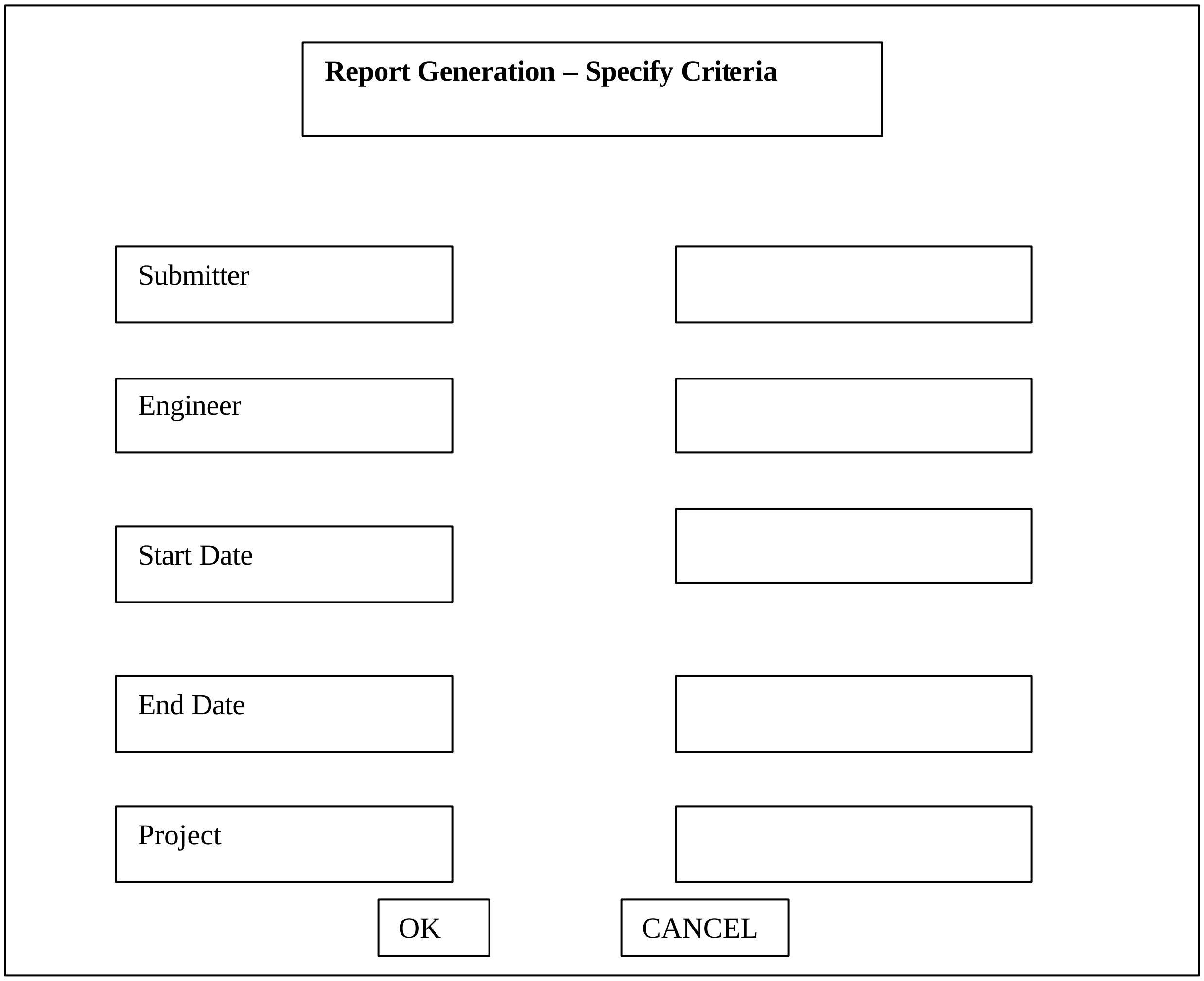
The defect tracking system needs to be contacted for creating extra fields, storing the field values and retrieving them. The Interface uses the defect tracking tool specific APIs to achieve this.

8. Processing

The input for the processing module is a set of defect identifiers generated through a query in the report generation phase. It then retrieves the defect entries for the set of defect identifiers. The defect entries include the regular defect fields like submitter, product, date of submission, etc., along with the additional fields required for finding root cause like impact, origin etc.,
The defect table entries are populated into internal data structures in the processing module. The processing module then has logic which would generate counts of various defects field wise. For example, number of bugs with Impact field as maintainability. After the counts are generated, the correlation between various fields are generated. This would then hand over the results to the reporting module.

9. Reporting

The reporting module shows a GUI to the user to specify the bugs for which the root cause has to be found out. It contacts the processing module and then after processing, gets the results and displays it to the user. It takes care of the presentation of output into pie charts, bar charts and tables.



Report Generation – Specify Criteria

Submitter	
Engineer	
Start Date	
End Date	
Project	

OK CANCEL

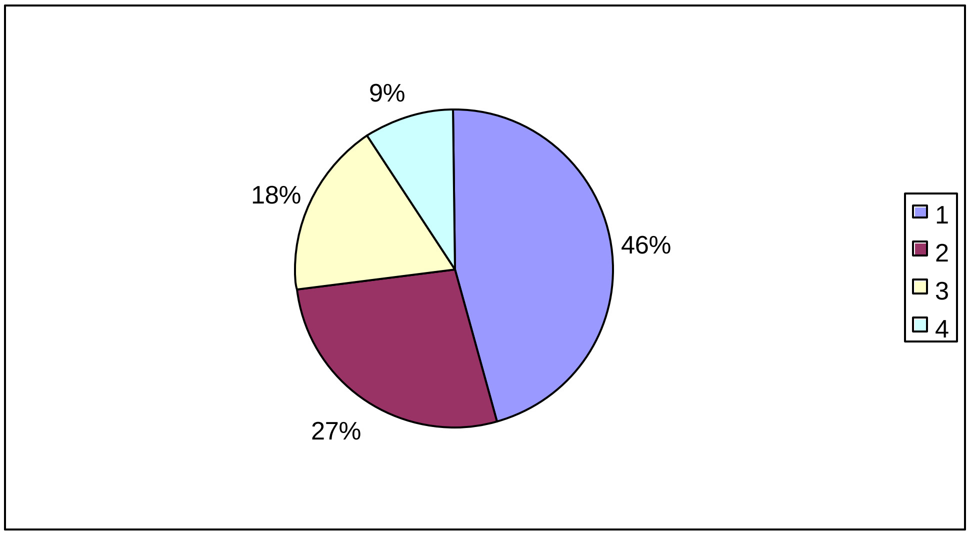
10. Query Output Format:

The query output has the bug Identifier and the other fields.

Bug Id	Impact	Trigger	Origin	Category	Reason

11. Report Output Formats

(1) Pie charts, which indicate the percentage with respect to other fields. E.g., a pie chart for the impact for a particular set of bugs.



Where 1 is performance, 2 is Maintainability, 3 is Installation problems, 4 is security.

Here the percentage of bugs which has performance problem is 46%, etc., So, we could conclude that the key bottleneck is performance.

Likewise we could have pie charts for trigger, origin, category and reason.

(2) Correlation tables

Correlation is powerful and allows attributing one factor to another. For example, the Usability problem came because of the Bad code fix.

Origin

Impact

	New Code	Bad Code Fix	Design
Usability			
Capability			
Security			
Maintainability			

Likewise the report would have co-relations between Impact and Category, Category and Origin, Trigger and Origin etc., In addition the standard fields in a defect tool like, Engineer, Submitter could be correlated to the additional fields like Impact, Category, Origin and Trigger.

12. Using the root cause reports to find root cause of defects.

The root cause reports gives answers to a variety of questions. It also shows potential problem areas. Let us see how the report aids in finding the root cause. For example, an impact analysis done on defects found by the customer found bugs. In this case, from the impact analysis pie chart it is possible to determine the top 3 impacts to the customer. I.e., what are the 3 problems in the product, which affect the customer most.

Let these top 3 problems identified from the pie chart output be security, maintainability and scalability. Next using correlation table output, which maps triggers to impact we could find out the top 3 triggers that caused an impact. Let the top 3 triggers that affect the impact security as identified from the correlation table be workload/stress, basic function and side effects. This indicates that to avoid or reduce the security impact the test plan could have more coverage dealing with workload/stress, basic function and side effects.

But, which stage of the process needs a fix to minimize the top 3 impacts? The correlation table between impact and origin has the answer. What were the stages that needed fix and what is the fix. The correlation table between origin and category will give the answer. Likewise it is possible to use the tables and charts based on the problems we are trying to address. The tables and charts also indicate the problem areas. For example, Top 5 activities in the software development life cycle that need improvement could be found out by the pie charts which shows the origin analysis.

13.Benefits of using the tool.

The following are the benefits of the tool:

- The tool being web based would allow teams working across the globe to enter data and find root cause for defects For example., Testing team in city X could enter information in the impact and trigger fields, development team in city Y could enter information in the fields origin, category and reason and a team in city Z could generate the root cause reports.
- Using the tool the user could do root cause analysis for a subset of bugs that is a result of a query.   For example, All bugs filed between 1st September 2001 to 1st October 2001 could be one query. So, the fields like impact, origin needs to be entered only for the defects which match the query, thus avoiding entry of the additional fields in all the defects.
- The tool is defect tracking system independent. It is easy to migrate from one defect tracking system to another, by just modifying the defect tracking interface module.
- The main advantage is that the finding the root cause can be done without involvement of any development group personnel after entry of the additional fields.

14.Conclusion

In summary, the tool provides the unique ability to automate finding of root cause of defects. The reports generated are in a presentable format and would help in easily determining the root cause of defects. In future, the tool could be extended to do qualitative comparisons between two similar projects. This would compare the trends between projects of the same type. This could indicate the skew in the quality between projects of similar type within the same organization.

15. References

Much of the concepts about RCA is derived from Orthogonal Defect Classification (ODC).

Links related to ODC - http://www.chillarege.com/odc/home.html
http://hockeytown.eas.asu.edu:8080/ODC/summary