International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:12, 2009

# Software Reengineering Tool for Traffic Accident Data

Jagdeep Kaur, Parvinder S. Sandhu, Birinderjit Singh, Amit Verma, Sanyam Anand

***Abstract*—**In today's hip hop world where everyone is running short of time and works hap hazardly,the similar scene is common on the roads while in traffic.To do away with the fatal consequences of such speedy traffics on rushy lanes, a software to analyse and keep account of the traffic and subsequent conjestion is being used in the developed countries. This software has being implemented and used with the help of a suppprt tool called Critical Analysis Reporting Environment.There has been two existing versions of this tool.The current research paper involves examining the issues and probles while using these two practically. Further a hybrid architecture is proposed for the same that retains the quality and performance of both and is better in terms of coupling of components , maintainence and many other features.

***Keywords*—**Critical Analysis Reporting Environment, coupling, hybrid architecture etc.

## I. Introduction

EVERY software system has well defined structure including its components, their visible properties, and the connections between them. To reach on to an improved version of the existing software architectuer,re-engineering of the product is performed. Reengineering[3] is the process of examination, understanding, and alteration of a system with a core objective of implementing the system in a new and better form in some terms.Now this kind of reverse engineering require ome type of support tool. CARE [12] (Critical Analysis Reporting Environment) is a software tool developed over the past fifteen years which is being used to analyze large amounts of data quickly, and to present the results in a way that is easy for layman to understand. Initially CARE was developed as a single- user, stand-alone application for the Microsoft DOS/Windows platform (called version CARE 99), which is implemented entirely in [1],[2] Visual C++.It analyzes the traffic accident data that the department of transportation(DoT) collects. There are two distinct types of data that the DOT collects:

- The accident data includes the time and date of the accident, information about the drivers, causes, and vehicles, and other things
- The location data includes the nearest mile marker to the accident and similar information.

As these data correspond to traffic accidents, the data is discrete which is handled in the software by taking separate variables.In order to learn about the scope of improvement, first the basic working of the existing version has to be examined.This will serve as the scratch point for architecture re-engineering and improvement. The following figure shows a simplified view of the software in concern.

.

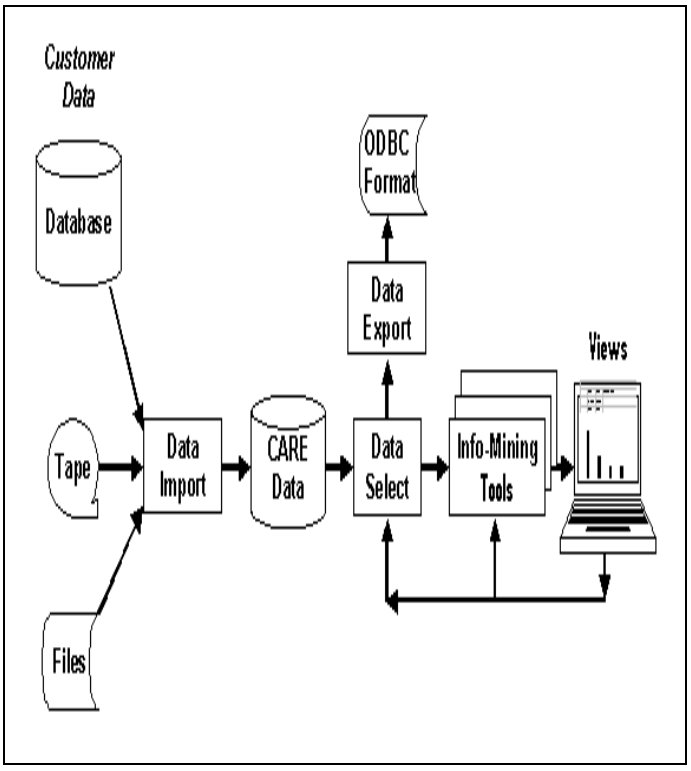


Fig. 1. Simplified Structure of Basic Version of Concerned Tool

The customers' data is imported into the specified proprietary format. The software then allows users to select data for various information-mining calculations, or for export to another tools, such as a database or spreadsheet. Some calculations produce results that can be re-used as inputs to

Sanyam Anand and Jagdeep Kaur are doing her Masters from Deptt. Of IT, Chandigarh Engineering College, Landran, Distt. Mohali, Punjab, INDIA.

Dr. Parvinder S. Sandhu is associated with Computer Science & Engineering Department, Rayat & Bahra Institute of Engineering & Bio-Technology, Sahauran, Distt. Mohali (Punjab)-140104 INDIA .

Amit Verma and Birinderjit Singh are associated with Electronics & Communication Engineering Department, Rayat & Bahra Institute of Engineering & Bio-Technology, Sahauran, Distt. Mohali (Punjab)-140104 INDIA

.

more calculations.

## II. Analysis and Problems in Existing Architecture

The following are the major points to be understood:

i.) First, the import is defined in C++ code, and the code must be changed for each customer. The importing dat requires involvement of programmer.[11]

ii.) The current code is monolithic, and does not allow new functionality to be added in or changed easily.So adaptation of tool to new domainsis a tidious job.

iii.) It requires altering a large part of the single enormous piece of code to add new calculations, filtering techniques, or views.

iv.) Also, the current code is not well organized, not well commented and very difficult to read.

The enhanced version of the same tool appeared , making it more modular and hence more extensible.In that, the user interface is a physically separate component. But the control of CARE 2000 is still embedded inside the user interface calculations, whereas filter management[8], file access, etc. are separate. Here all components are put together into a single compiled library. So , it all must run on one processor even though there are two independently compiled components.In other words there arises a need for a plug-in architecture which has been proposed as below.

## III. Proposed Solution Architecture

The figure 2 that follows shows the proposed solution to the above problems.The basic aim is to have an architecture that allows for all of the functionality of previous version as well as solves the issues of modularity and adaptabilityin the functionality.This structute is made up of a kernel, along with pluggable tools, views, and data. Each tool is an independent component that performs some operation on a set of inputs, and produces an output. Each tool has a declaration of its input and output parameter types, and is associated with a view. Further more the view is actually a separate component from the tool and parameter types. So each tool can support multiple views. Here, each of these components is a separately compiled unit with a well- defined interface. So it offers multi-dimentional deployment options.[7] Now each unit may run on the same processor, as in the previous version.The other option is to make each component run on separate networked processors. The available deployment choices straight a way affect the performance and security, along with other issues.It is obvious that performance would be best if the data and tools were on the same processor. Also, the data can be kept on different processors for security reasons.

The major advantages of the proposed architecture over the previous one are listed as follows:

i.) Each of the components can be compiled independently,leading to better deployment options.

ii.) A new tool can be developed separately from the rest of the CARE code, and the new compiled code can be added to the CARE system [5] without changing any of the old codes.

iii.) Views are more separate and all code in a view is unique.

iv.) If something has to be changed, the changes will be localized to just one or a few small components, and will not impact many components.This promotes easy maintenance.[10]

v.) Improved performance and security[12]

vi.) Well-defined and consistent architecture.

vii.) Enhanced flexibility

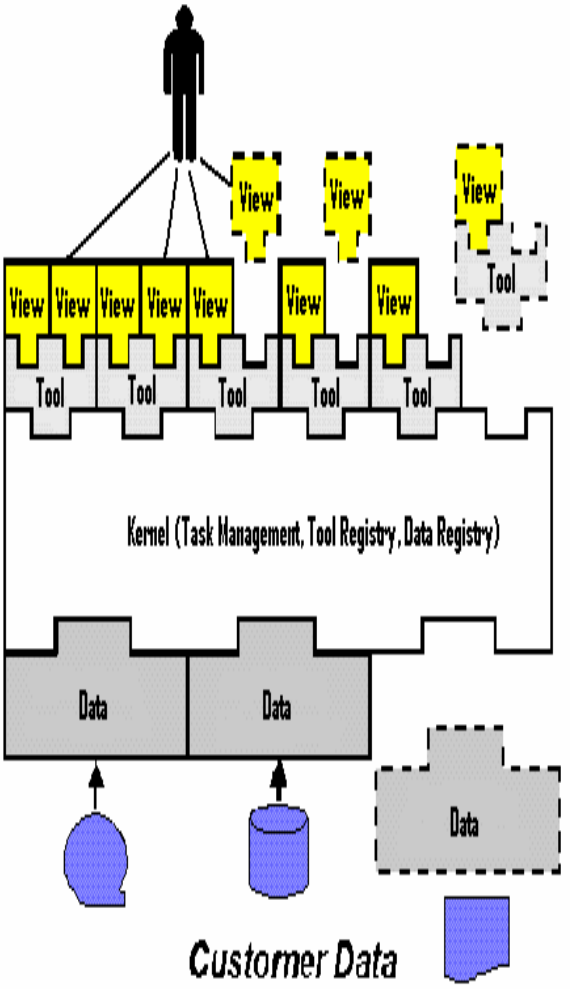


Fig. 2. Proposed "Plug-In" Architecture

## IV. Conclusion And Disscussion

By first discovering, documenting, and analyzing, and then re-engineering theseexisting architectures with specific architecture quality attributes in mind, this new architecture is proposed.These architectures[9] have now been analyzed and compared, only intuitively.

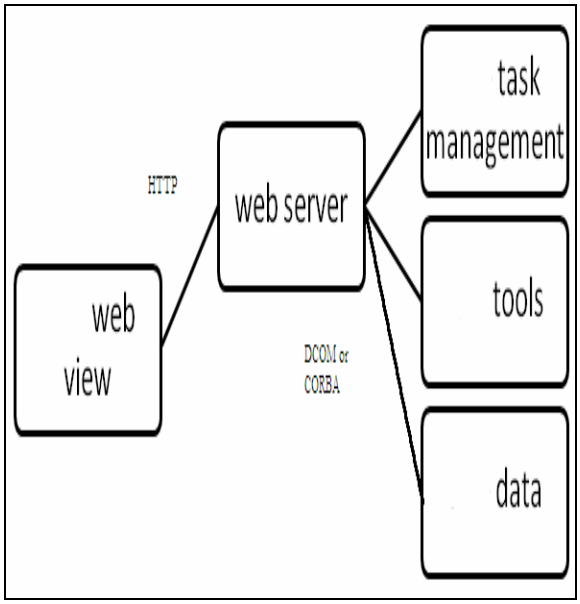


Fig. 3. Architecture Adaptability on Web

Further, the proposed architecture can be proved to be the best architecture out of the previous ones with respect to modifiability and distributability.This can be done using any of the available software analysis tools.To sum up, there can be given for two main reasons for at par performance of the proposition. First, the tool possesses the apty suitable variable-oriented storage and access techniques. Second, the data can still be passed by reference much like in the previous versions.

## REFERENCES

[1] Archer, Tom and Richard C. Leinecker. 1998. Visual C++ Bible. Foster City, CA: IDG
[2] Bass, Len, Paul Clements, and Rick Kazman. 1999. Software Architecture in Practice.
[3] Reading, MA: Addison Wesley.
[4] Box, Don. 1998. Essential COM. Reading, MA: Addison Wesley.
[5] Brown, David B. 2000. Critical Analysis Reporting Environment (CARE): A Paradigm for Information Mining. 2000 [cited 13 March 2001]. (http://care.cs.ua.edu/docs/marketing.ppt).
[6] Folwer, Martin. 1997. Dealing with Properties. 1997 (http://www.martinfowler.com/ apsupp/properties.pdf). Kazman, Rick, Len Bass, Gregory Abowd, and Mike Webb. 1994.
[7] SAAM: A Method for Analyzing the Properties of Software Architectures. Proceedings of the International Conference on Software Engineering. ICSE'16: 81-90.
[8] Kazman, Rick, Gregory Abowd, Len Bass, and Paul Clements. 2001. Scenario-Based Analysis of Software Architecture. Pittsburgh: Carnegie-Mellon-University,1999 (http://www.sei.cmu.edu/ architecture/ scenario_paper /).
[9] Kruchten, Philippe. 1995. "Architectural Blueprints – The '4+1' View of Software Architecture". Vancouver, B.C., Canada: Rational Software Corp., 1995 (http://www.rational.com/media/whitepapers/Pbk4p1.pdf).61
[10] The Object Management Group. 1999. UML 1.3 Specification, (http://www.rational.com/media/uml/post.pdf).
[11] Quatrani, Terry. Visual Modeling With Rational Rose 2000 and UML. Reading, MA: Addison Wesley.
[12] The University of Alabama. Engineering Research Lab. CARE Team. 2000. CARE Traffic Safety Data Analysis Page, (http://care.cs.ua.edu)