INFORMATICA, 1995, Vol. 6, No. 4, 387-396

A VISUALIZATION AND ANALYSIS OF EXPERIMENTALLY GATHERED RESULTS IN AN USER-FRIENDLY MODE

Baiba APINE

Riga Institute of Information Technology
Skanstes St. 13, Riga, Latvia, LV1013

Abstract. Software development consists of several phases, where each phase has its own results. Language, which allows to describe collected results, their transformation and displaying, is discussed in this paper. A software tool is offered as an interpreter for this language. The language and software tool form a complex for data analysis. The complex is open and could be adapted for usage in different software system development stages. Object-oriented methodology for system specification design is used to show structure of the language and software tool.

Key words: computer aided system engineering, CASE, data analysis, information presentation, data description languages, software systems.

Specialists involved in software system development process use different software tools to study various aspects of the system. Tools accumulate information and create results, which must be organised and passed to the developer in the user-friendly mode (reports, charts, tables, etc.). Problem has three solutions:

- The first – to create results analysis tool for every aspect of software system. Results analysis tool provides minimum of calculations and reports, but supports data export to more powerful data analysis tool. This approach is very labour – consuming and depends on specific data.
- The second – to use already existing multi-purpose results analysis tools, spreadsheets, statistical packages, for instance. These tools are powerful, they have their own languages for data description. User must profit by the experience of using them. Frequently user doesn't need even fifty per cents of all features provided by universal data analysis tool.
- The third – to create a simple data description language, which de-

scribes collected information structure, transformation and visualisation, and software tool as interpreter for this language. Software tool is independent of the process, which stores results.

The third solution is provided in this paper. Solution is a midway of creating the particular results analysis tool for every software development tool and usage of universal data analysis tools. Process, which stores data, hasn't take care about collected data visualisation, but user won't need special skills for working with collected data (Fig. 1).

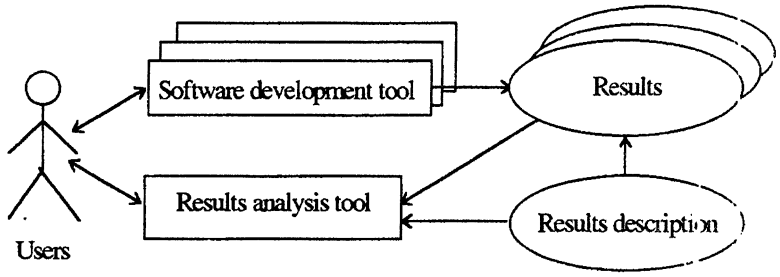

Fig. 1. Language for data description and tool as interpreter for this language.

Results analysis complex may be used by different processes. It is convenient to consider results analysis complex as closed system, which interacts with environment via separate objects, and choose the object–oriented methodology (Rumbaugh, 1991) for its specification design. Object model shows data description language structure as well as structure of the software tool (Fig. 2).

Object *Data description* and its subclasses show the data description language structure. Subclass *Data transformation* is the part of the language, which describes how get results from previously collected data by using of predefined functions (Fig. 3). For instance, a sequence of events is accumulated during some time interval. Each event in the sequence may be a seizing of a computer or it's releasing. Following attributes describe each event: time moment when event is registered, computer identification and event type (seizing or releasing). One of possible functions is calculate the total and average seances of computer usage during fixed time period.

Object *Data structure* shows, how data stores in data storage (Fig. 4).

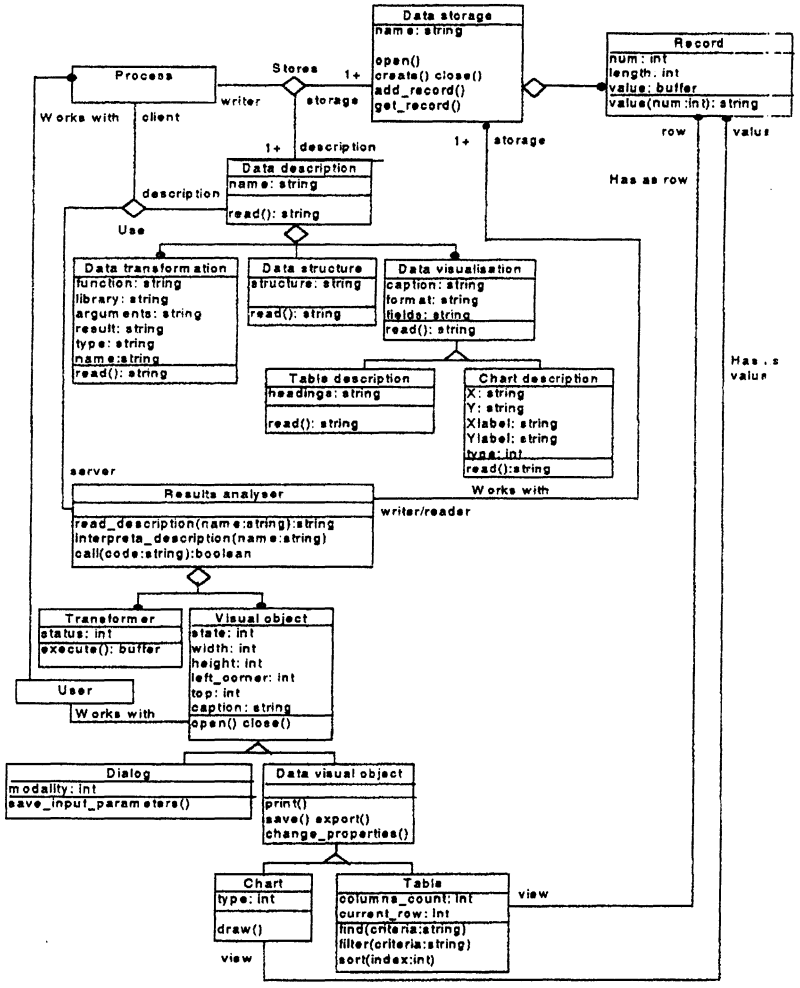

Fig. 2. Data description language and interpreter makes a whole unit for gathered results analyze.

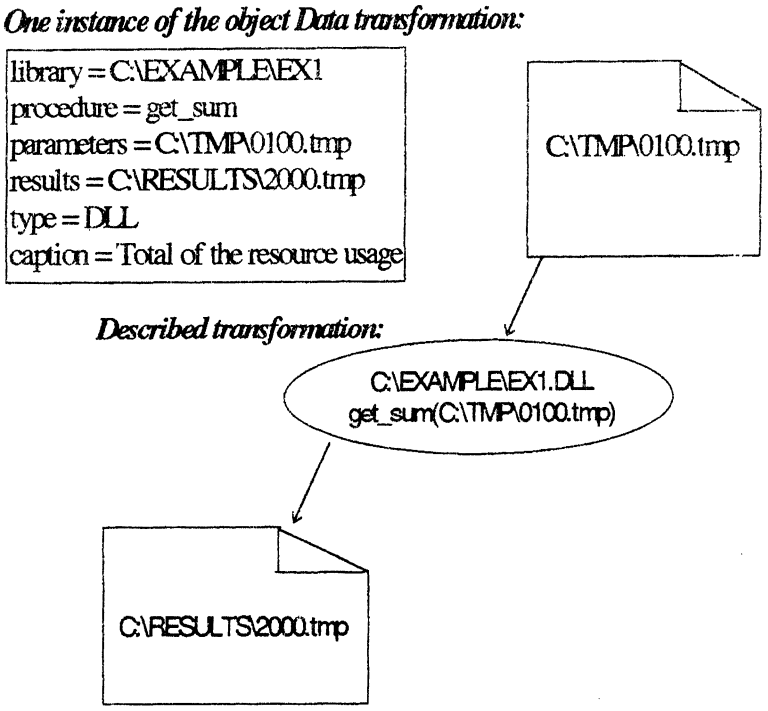

Fig. 3. Sample of data transformation description.

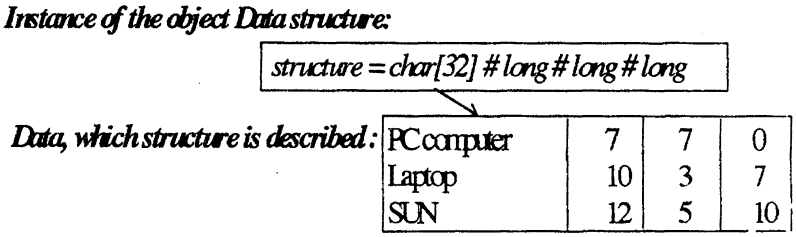

Fig. 4. Sample of data structure description.

Instance of the object Table description:

caption = Total About Resource Usage
headings = Resource # Total time used # Average seance length # Maximum seance length
fields = 0 # 1 # 2 # 3
format = # DURATION # DURATION # DURATION
sortDefault = 0

Data:

PC computer	7	7	0
Laptop	10	3	7
SUN	12	5	10

Table generated by tool according to this description:

Resource	Total time used	Average seance length	Maximum seance length
PC computer	7s	7s	0s
Laptop	10s	3s	7s
SUN	12s	5s	10s

Fig. 5. Sample of table description.

Object *Data visualisation* shows how collected and transformed data have been shown to user. Its subclasses *Table description* and *Chart description* show data representation in table or in chart. Every instance of these objects describes one table (Fig. 5) or chart (Fig. 6).

Software tool is an interpreter of the language. It takes collected data, transforms them, if necessary, and displays according to description. Object model highlights, which objects of the software tool are essential for interacting with an environment. For instance, user will interact with the software tool via object *Visual object*. This part of the software tool is interactive process, therefore it would be better to implement it in an event-driven environment. Part, which transforms (object *Transformer*) data may be considered as continuos process and implemented by using of a procedural environment.

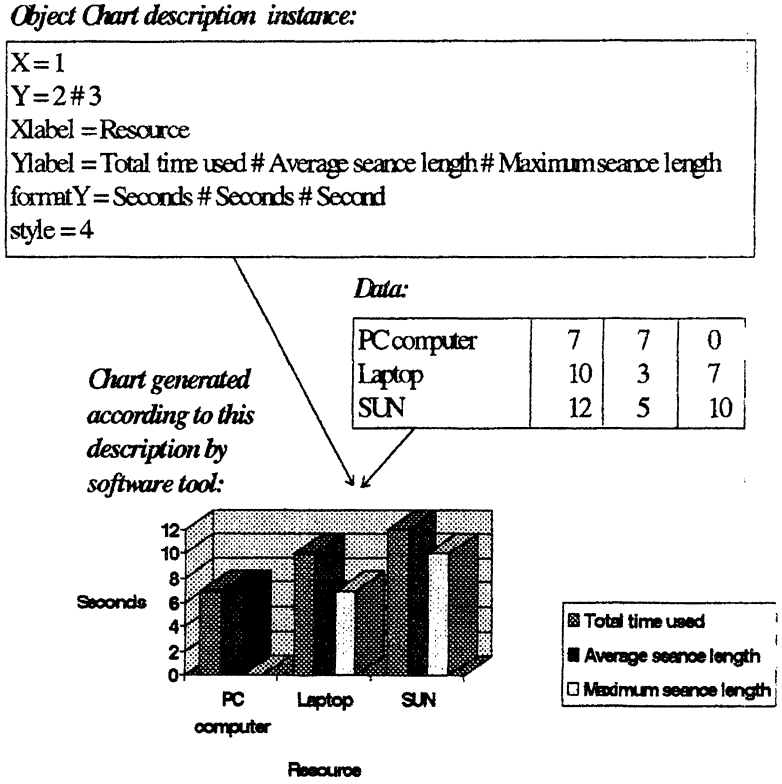

Fig. 6. Sample of chart description.

Software tool is implemented and used by some modelling tools. Software tool is the component of the set of GRADE (1993) system development tools. Its name is **Trace Browser**. **Trace Browser** is used by business modelling component for displaying of collected results (GRAPES–BM, 1995). Results may be described as follows:

- Task name – task name, at which events queue is considered. Task is a function, for which all necessary resources are fixed. For instance, task is to register an order and necessary resource for its carrying out is a computer.

- Event name – event name at the task identifies events queue. Event is an initiator of the beginning of the task execution. For instance, entering of an order invokes order's registration task.
- Maximum (average, minimum) length of queue-maximum (average, minimum) of events, located at the task during fixed time period.

Business modelling tool stores data. If business model designer wants to work with collected data, business modelling tool generates data description according to the data description language, calls **Trace Browser** and passes collected data and generated data description.

For instance, data are collected in data storage, which structure is described as follows: structure = char[33]#char[33]#double#double#double (Fig. 7).

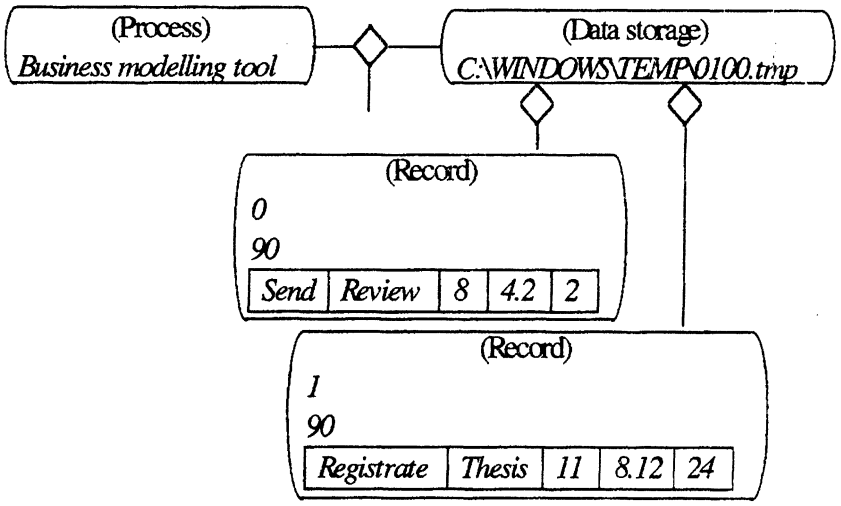

Fig. 7. Part of instance diagram, which shows data storage contents.

Business modelling tool assigns semantics to collected data, when generates data visualisation description. The following fragment is necessary to show collected data in a table (Fig. 8):

caption = Events Queues Length
headings = Task Name # Event Name # Maximum Queue Length # Average Queue Length # Minimum Queue Length
fields = 0#1#2#3#4
format = ##FLOAT#FLOAT#FLOAT

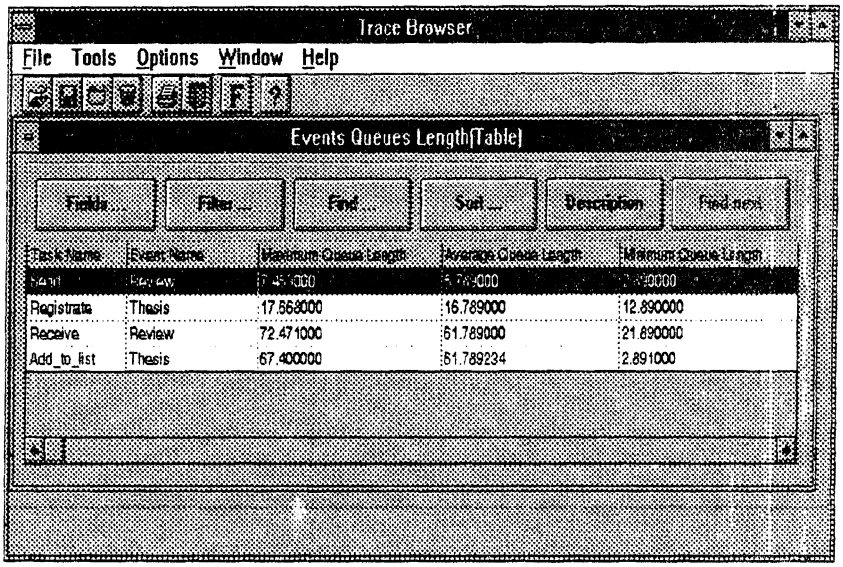

Fig. 8. Table generated by Trace Browser according to table description.

Table caption is the value of the attribute *caption*. Values of the attributes *headings*, *fields* and *format* are lists separated by "#". Their values are interpreted simultaneously: value from data storage zeroes field is collected in the first column of the table in the same format as it stores in the data storage. The first column caption is Task *name*. And so on with every item of the lists.

The following fragment is necessary to display collected data in bar chart (Fig. 9):

caption = Events Queues Length
Xlabel = Task Name # Event Name
Ylabel = Maximum Queue Length # Average Queue Length # Minimum Queue Length
X = 0#1
Y = 2#3#4

The heading of the chart is the value of the attribute *caption*. Labels on the horizontal axis are values from zeroes and first fields from data storage (attribute X value). Three values are shown for every record in the data storage. Values from second, third and fourth fields (attribute Y value).

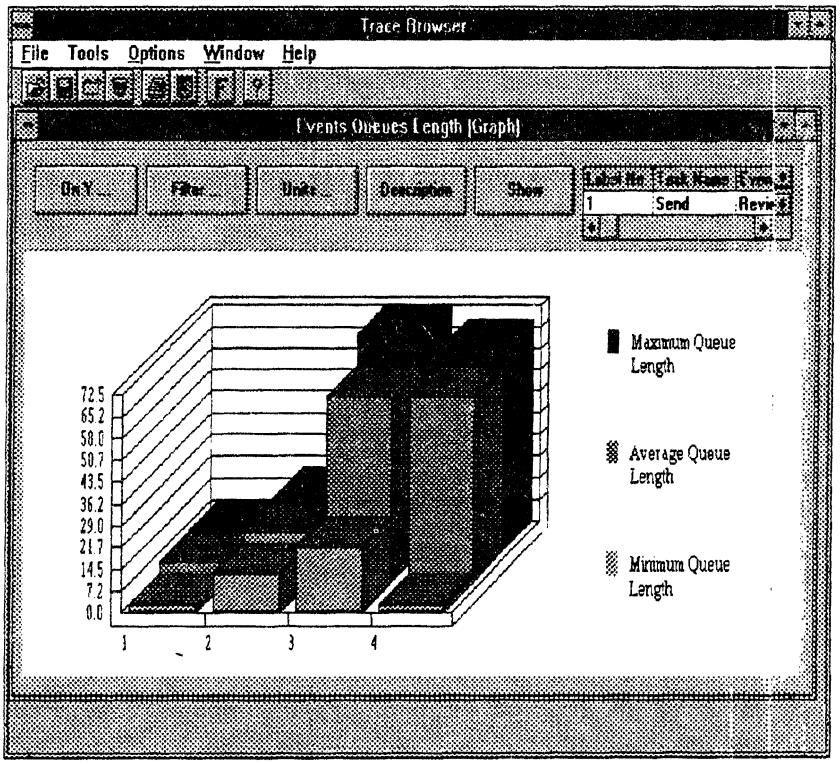

Fig. 9. Chart generated by Trace Browser according to table description.

Described approach has some advantages:

- Results analysis tool (Trace Browser) is independent from process, which stores data, and can be used by various processes.
- Results analysis tool may be updated with new functions for data transformation.
- It is better, if the whole set of software development tools use one and the same results analysis tool.

REFERENCES

GRADE V1.0 (Windows): Modeling and Development Environment for GRAPES–86 and GRAPES/4GL. User Guide. (1993). Siemens Nixdorf Informationssysteme AG. 282pp.

GRAPES–BM: Version 2.1: New Features of Language and Tool Support, Simulation Part. (1995). August 3. 23pp.

Rumbaugh, J. (1991). *Object–Oriented modelling and design.* Prentice–Hall, Inc. 542pp.

Received November 1995

B. Apine is a programmer at the Riga Institute of Information Technology (Latvia), Master of computer sciences. His current research interests include computer aided system engineering (CASE) and software systems, data analysis and information presentation methods.

APIE EKSPERIMENTO DUOMENŲ PATEIKIMĄ IR ANALIZĘ VARTOTOJUI PATOGIU BŪDU

Baiba APINE

Programų sistemos kuriamos etapais. Kiekviename etape gaunami rezultatai. Šiame darbe aptariama kalba, skirta gautiems rezultatams bei naudojamiems jų transformavimo ir vizualizavimo būdams aprašyti. Aprašytas tos kalbos interpretatorius. Kalba ir interpretatorius sudaro duomenų analizės priemonių kompleksą. Kompleksas gali būti pritaikytas bet kurio programų sistemos kūrimo etapo duomenims apdoroti. Kalbai ir interpretatoriui aprašyti panaudota objektinė metodika.