Combining Data from Existing Company Data Sources: Architecture and Experiences

Jari Vanhanen, Kai Risku, Pekka Kilponen
Helsinki University of Technology, P.O.Box 9555, FIN-02015 HUT, Finland
E-mail: firstname.lastname@hut.fi

Abstract

Combining and utilizing data from different sources is a common problem in many companies today. Useful data can be found in various applications, such as spreadsheets, project management software or version management systems, but it is often an arduous and manual task to combine the information and regularly utilize it in a larger context.

In this paper we describe an open architecture for collecting and utilizing data from different sources and our implementation based on this architecture. Finally, we present experiences gained from industrial environments, where the system is used for improving the controllability of product development.

The architecture consists of a Metrics Server (MESS), with external data transfer modules (EDAMs) and applications utilizing the collected data. In addition to the architecture, an EDAM for Microsoft Excel and a data visualization application are briefly presented.

1. Introduction

Combining and utilizing data from different applications is a common problem in many companies today. In the LUCOS research project we have focused on developing methods [1,2] and tools for improving the controllability of new product development (NPD). We have noticed that in the domain of NPD, the problem of dispersed data is no different. Useful data can be found in various applications, such as spreadsheets, project management software or version management systems, but it is often an arduous and manual task to combine the information and regularly utilize it outside each application in a larger context.

Data warehousing systems are typically intended for helping business decision making based on combining large amounts of historical data from dispersed operational databases [3]. We think that an information system that borrows the data warehousing ideas of combining data, presenting it to potential users, and utilizing both historical and real time data, is useful also in supporting NPD controllability. In general, the visibility of the NPD process is poor. It can be seen, e.g., in the poor controllability of many product development projects [4]. This kind of information system can improve the visibility of individual projects as well as dependencies between projects, and help in summarizing data from individual projects to be used on process or strategic level management. Archiving and analyzing project data allows a company to learn from it for its future projects.

In the domain of product development useful data is not necessarily explicitly collected into databases. It is not always even recognized that there may already be several data collection systems in a project whose data could be utilized, especially if the data were combined. These data collection systems typically include, e.g., time reporting systems, project management applications, version control systems, and bug reporting databases. Instead of simply querying the data from databases, it must often be extracted from application specific files such as time reporting spreadsheets or logs of a version control system. In contrast with a typical data warehouse the amount of data is moderate but the real time requirements are higher for making daily project management decisions in addition to observing longer trends.

We have designed an open architecture, whose purpose is to serve as an information system for product development organizations. It is intended, especially, but not exclusively, for collecting project data and visualizing it in a customized and understandable form for all stakeholders at all levels of an organization, e.g. developers, testing personnel, project managers, and corporate management. Our implementation based on this architecture is a system that is simple to deploy and for which it is easy to write new modules for transferring data to our system or for utilizing data from the system. Our system is available for any authorized user having an

Intra-/Internet connection to the server, and it is scalable for large amounts of data and users.

The next section of this paper presents the architecture and our implementation, including descriptions of all the constituent parts. In the last section we report experiences on using our system in an industrial environment.

2. Architecture

The LUCOS Tool Framework (Figure 1) consists of the *Metrics Server* (MESS), with external data gathering and replication modules (EDAMs) and applications utilizing the collected data, e.g., the *Visualization Client Applet* (ViCA). MESS stores collected data, performs user authentication and access control, and provides services for manipulating metadata. All client programs, i.e., EDAMs and client applications, communicate with MESS over an Intranet or the Internet using an HTTP based protocol.

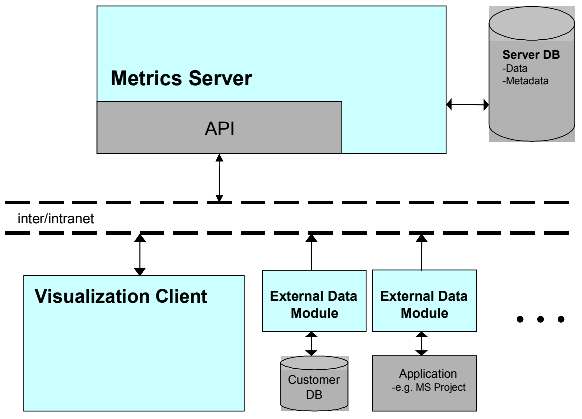


Figure 1: The LUCOS Tool Framework.

2.1. Metrics Server

The Metrics Server (MESS) is the central component in the framework providing a variety of client programs with persistent storage facilities, user authorization, and data querying and manipulation. In addition it can operate as a simple www-server. MESS is the only program that directly accesses the server database.

MESS stores and manipulates both the data collected from the data sources and framework data such as *users, groups, schemas, views, charts and panels.* User and group information is required for authentication and for limiting access to any object in the database on a group level. Schemas describe the field names and data types of database tables in the server database. The collected data is stored in these tables. Views are user generated virtual table descriptions typically used for simplifying complex queries. Charts and panels are objects generated and used by the Visualization Client Applet. Charts describe for ViCA, e.g., SQL queries, data mappings and appearance of visualizations. Panels group related charts together. In addition to entity specific attributes, each framework data entity contains also common metadata: name, description, and read, write and admin group name.

MESS allows client programs to create, read, update, and delete entities and manipulate their metadata. Data can be dumped to user created tables and SQL queries executed to read the data. MESS can store and return files, thus serving as a simple www-server. Logging of user actions can be utilized both for evaluating the use and performance of the system, and for improving its usability.

All communication with MESS is done using the *LUCOS Tool Protocol*, which is based on the hypertext transfer protocol commonly known as HTTP/1.0 [5]. HTTP is a simple lightweight and stateless protocol, taking place over TCP/IP communications. MESS processes incoming requests and returns a proper response to the client. Each HTTP requests contains a method, a resource identifier and optional data (authorization credentials, metadata information, etc.). The method (GET, PUT, or DELETE) specifies the action to take on the identified resource, and the optional data can alter the behavior of the processing. The resource identifier specifies the type of resource by the top-level "directory" of the identifier. For example, the request `GET /user/fred HTTP/1.0` fetches information (metadata) on user 'fred'.

In fact, MESS can be thought of as a www-server that implements a virtual file system with different types of named objects residing in different directories. Special directories (`/dump, /sqlquery, /meta, /file`) and files inside the directories exist for executing specific functions.

Sometimes source databases may be so huge that data replication to MESS is unfeasible. Replication can be avoided by connecting MESS to a database management system that is capable of accessing remote, heterogeneous data servers, e.g., Enterprise Data Studio from Sybase Inc. [6]. From the viewpoint of MESS, the difference is noticed only in slower response times to SQL queries.

MESS allows proactive caching of commonly used SQL queries to minimize waiting times. Usage logs are used to find out usage patterns such as which are the most common SQL queries, how long they typically take to execute, and when they are executed. Based on this information MESS tries to maintain valid results for common queries in its cache by executing the queries using processor idle time. Because MESS relays SQL queries from clients to the server database, it can return the result from its cache, if available.

Advantages of including MESS to the architecture. We chose to include a separate server instead of directly accessing a central database from client programs using ODBC for several reasons. MESS provides a higher level interface on top of SQL and ODBC interfaces for manipulating entities such as users, groups, application specific entities, and metadata of all entities. Manipulating this information does require neither writing SQL queries nor knowledge of the database schema. The access control mechanism is implemented in MESS and provides *read*, *write* and *administration rights* to each entity and entity directory on a group basis.

Because HTTP is a very common protocol, several programming libraries support it. Therefore it is easy to write communication code for client programs and they can be developed in most programming languages on a variety of platforms.

The proactive caching and the option not to replicate all data make the system more scalable for large installations.

2.2. External Data Modules

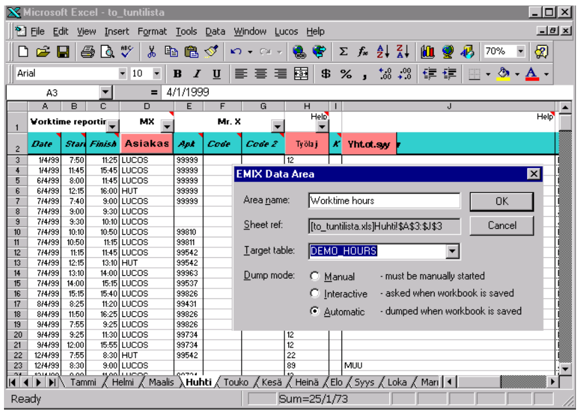
The External Data Modules (EDAMs) are typically company-specific programs that connect to some existing source of data (database, log files, etc.), perform some optional data transformation, e.g., cleaning and reformatting, and then copy the relevant data to the server. Since an EDAM must connect to some company data source, the choice of architecture and programming tool or language is limited by the methods of access to the data source. Different EDAMs can be developed on different architectures with different programming tools as needed.

Typically an EDAM performs the following steps when executed. First it establishes a connection to MESS and authenticates itself to gain access to target tables on the server database. It checks if the required database tables are present in MESS, and if needed, creates the missing tables and gives them reasonable access rights. After initialization, the EDAM connects to one or several data sources and fetches (or parses) the data and performs optional filtering, transformation or aggregation of the data. If possible, the EDAM only reads the new or changed data thus minimizing the amount of data to dump to MESS. Finally, the source data that has been read and processed is dumped to the corresponding tables in MESS.

Usually administrators of the system should write EDAMs to solve the organization's specific data transformation needs. However, we have implemented a programming library and a couple of EDAMs to alleviate the initial effort in deploying the system. *The LUCOS library* is a C-language library providing high-level functions to communicate with MESS. These functions include, e.g. initializing and closing connections, creating and deleting tables, and dumping data to a table. *CSVTool* is both a sample program for using the library and a useful generic EDAM that reads files containing data in CSV (comma separated value) format and dumps it to a specified table in MESS. CSV is a format that many applications, including MS Excel and MS Project, can generate.

Because most of our pilot users are using Microsoft Project and Microsoft Excel for project management purposes we have developed plug-ins for those products. *The Excel Metrics Information Exchange* (EMIX) plug-in and the respective MS Project counterpart *Project Metrics Information Exchange* (PMIX) automate data transfer from these applications to MESS. They can be configured for each document separately to dump selected data in the document to specific tables in MESS. Thereafter they can make the dump automatically whenever a user modifies and saves the document.

For example, when a new spreadsheet document template is created, the author marks in Excel all areas containing data to be transferred and specifies for each data area the table where the data is to be dumped (Figure 2). After this any user may make a copy of the document template for personal use and EMIX will transfer the data automatically when saving of the document.


Microsoft Excel - to_tuntilista
File Edit View Insert Format Tools Data Window Lucos Help
70%
Arial
10
A3
= 4/1/1999
Worktime reportir
MX
Mr. X
Help
Help
Tammi Helmi Maalis Huhti Touko Kesä Heinä Elo Syys Loka Marı
Ready
Sum=25/1/73

Date	Start	Finish	Asiakas	Apk	Code	Code 2	Työlaj	K	Yht.ot.syy
1/4/99	7:50	11:25	LUCOS	99999			12		
1/4/99	11:45	15:45	LUCOS	99999					
6/4/99	8:00	11:45	LUCOS	99999					
6/4/99	12:15	16:00	HUT	99999					
7/4/99	7:40	9:00	LUCOS	99999					
7/4/99	9:00	9:30	LUCOS						
7/4/99	9:30	10:10	LUCOS						
7/4/99	10:10	10:50	LUCOS	99810					
7/4/99	10:50	11:15	LUCOS	99811					
7/4/99	11:15	11:45	LUCOS	99542					
7/4/99	12:15	13:10	HUT	99542					
7/4/99	13:10	14:00	LUCOS	99963					
7/4/99	14:00	15:15	LUCOS	99537					
7/4/99	15:15	15:40	LUCOS	99826					
8/4/99	8:25	11:20	LUCOS	99431					
8/4/99	11:50	16:25	LUCOS	99826					
9/4/99	7:55	9:25	LUCOS	99826					
9/4/99	9:25	11:30	LUCOS	99734			12		
9/4/99	12:00	15:55	LUCOS	99734			12		
12/4/99	7:55	8:30	HUT	99542			22		
12/4/99	8:30	9:00	LUCOS				89		MUU

EMIX Data Area

Area name: Worktime hours

Sheet ref: [to_tuntilista.xls]Huhti!$A$3:$J$3

Target table: DEMO_HOURS

Dump mode: ( ) Manual - must be manually started; ( ) Interactive - asked when workbook is saved; [✓ Automatic] - dumped when workbook is saved

OK Cancel

Figure 2: EMIX.

The mapping between the columns of a data area and the target table fields is automatic if the header line of each area contains the respective table field names. Otherwise, the user can manually configure the mapping for each data area.

A data area may be appended with constant value columns. A constant is defined either as a reference to a worksheet cell or as a literal value. The constants can, e.g., be used to qualify time reporting hours by a person without the need to have the username duplicated on each row in the worksheet.

2.3. The Visualization Client Applet

In addition to EDAMs, there are more advanced clients that provide the information access level to our system. These client applications make data queries to MESS and present the data to the users in an informative way. Client applications may also be used for administrative tasks such as building views to the data, or managing user accounts and access rights. Because the communication with the server takes place using HTTP, clients can be developed in most programming languages on a variety of platforms.

The Visualization Client Applet (ViCA) is based on Java technology and works via a WWW browser thus allowing access to authorized users from a company intranet or from anywhere via the Internet. Use is not limited to any specific hardware or operating system platform because the only requirement is the availability of either the Internet Explorer or the Netscape Communicator browser. This choice of technology has been made to allow access to the system regardless of the physical location of potential users, and without need for any software installation, except the browser.

ViCA provides users with tailored panels, enabling them to see only the information that is relevant to them. Panels are either presented in separate windows or embedded on WWW pages. Each panel may contain any number of charts, navigation buttons and images.

The charts are visualizations of data fetched from MESS. The contents of a chart can be scrolled allowing putting large amounts of data in a chart. Charts can be created by the users themselves, or selected from a predefined set created by a system administrator. Collecting predefined charts to new panels and changing the visual attributes of charts, e.g. size, color, and labels, is simple. Users can themselves assemble new panels containing all the information they need.

For an end user, using ViCA is as easy as web browsing. However, implementing new charts from scratch requires good knowledge of ViCA and of the contents of the MESS database. Data for the charts is fetched from MESS using standard SQL queries. The mapping of a query result to a chart is flexible and allows visualizing several data sets, e.g., information on several projects, in a single chart. Fourteen chart types such as line, bar, area, pie and Gantt charts are available for creating illustrative visualizations.

The difficulty in creating charts is alleviated by visualization libraries. They contain common domain specific visualizations and data requirement definitions. These libraries are added to ViCA as external components depending on the needs of the users. A domain specific library allows a user organization to know the minimum set of data required for practical visualizations and get the basic visualizations in use immediately. All the chart definitions in a library are built on an SQL view interface. The installation of a library requires this interface to be mapped to the company database. Thereafter a simple Chart Wizard can be used to create new charts based on the templates in a library.

Charts are typically grouped into proper panels providing information on, e.g., a process or a project, or some aspect of them such as schedule follow-up. Panels may be linked together into hierarchies using navigation. Navigation means that by clicking a button or a certain part of a chart, e.g., a task in a Gantt chart, a new panel opens providing more detailed information on the clicked item.

The need to create several similar panels for the same purpose is minimized by parameterization of panels, i.e. allowing SQL queries to contain variables, which get their values during navigation. Thus, for example, a general parameterized project panel may be constructed that shows information on a project that was selected for navigation in a project portfolio panel.

Two sample panels are presented in Figure 3. In the first panel, the Gantt chart shows all projects in a company. Clicking a bar in the Gantt chart opens a detailed panel showing detailed data of the selected project. If another project would have been chosen, a visually identical panel would have opened, the only difference being that the data would have been from the other project.

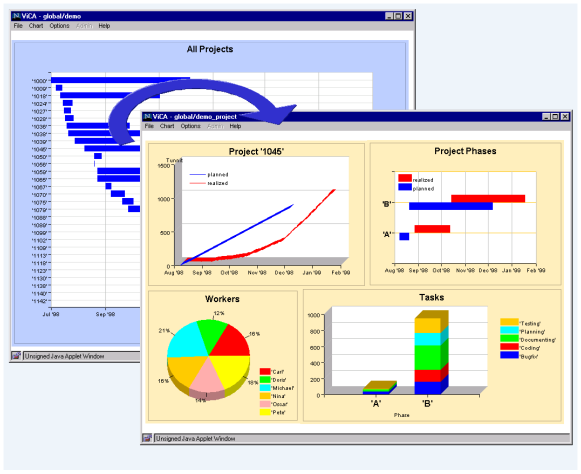


Figure 3: Two ViCA panels and navigation from a chart to another panel.

ViCA is also used for some administrative tasks, such as creating new user accounts and groups, and assigning access rights to charts and panels.

3. Experiences

The prototype version of the LUCOS Tool Framework is in pilot use in product development departments of five high-technology companies. In two of the companies the deployment has advanced to real use of the system in a couple of product development projects. So far the use has mostly been visualization of project progress with ViCA without aggregating data on higher levels.

3.1. A deployment scenario

Before deploying the LUCOS Tool Framework, the company should define the metrics, whose visualizations will be provided for different users in the company. The LUCOS method [2] can be used to derive metrics that are connected to the strategic goals of the company. Usually the existing reporting systems in a company do not provide all the required data for the desired metrics. Therefore the reporting systems must be enhanced or new systems must be deployed.

The deployment of the LUCOS Tool Framework can begin when the required reporting systems are in use. First, the database schema for the data to be stored in MESS must be designed, and an EDAM for each data source must be configured or implemented. If some of the provided EDAMs can be used, the configuration takes only a few hours for each data source. Otherwise, the EDAMs must be programmed, requiring a couple of days of work for each EDAM from a skilled programmer. Finally, the charts, panels and navigation structures must be built in ViCA, and the correctness of the data in the charts must be verified. This is the responsibility of a dedicated ViCA system administrator, and other users will only watch the visualizations. The effort required in creating a chart depends on the complexity of the chart, the SQL knowledge of the user, and whether some provided visualization library contains the desired chart or not. Therefore creating a chart may take anything from ten minutes to several hours. When the charts are ready, any user can construct a personal panel in a couple of hours or immediately watch a predefined panel.

Naturally, there is a continuos cycle in the development of metrics and visualizations. Whenever new goals and metrics are defined, new visualizations should be created for them in ViCA.

3.2. A deployment case

In one of the pilot companies our system has been in extensive use in observing project schedule keeping in a software development project. The project manager generates a project plan using MS Project. The project is scheduled with very low granularity, each task having the duration of two or three days at maximum. Working time data is collected using pre-formatted templates in MS Excel. Each row in Excel contains a task id that either links the task to the project plan or indicates unplanned work. These data are combined in MESS, and the original project plan with the current deviation from the original schedule is visualized in a Gantt chart (Figure 4) using ViCA.

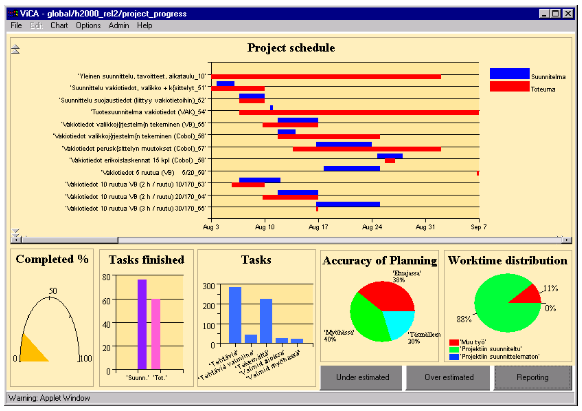

ViCA - global/h2000_rel2/project_progress
File Edit Chart Options Admin Help
Project schedule
'Yleinen suunnittelu, tavoitteet, aikataulu_10'
'Suunnittelu vakiotiedot, valikko + k{sittelyt_51'
'Suunnittelu suojaustiedot (liittyy vakiotietoihin)_52'
'Tuotesuunnitelma vakiotiedot (VAK)_54'
'Vakiotiedot valikkoj{rjestelm{n tekeminen (VB)_55'
'Vakiotiedot valikkoj{rjestelm{n tekeminen (Cobol)_56'
'Vakiotiedot perusk{sittelyn muutokset (Cobol)_57'
'Vakiotiedot erikoislaskennat 15 kpl (Cobol) _58'
'Vakiotiedot 5 ruutua (VB) 5/20_59'
'Vakiotiedot 10 ruutua VB (2 h / ruutu) 10/170_63'
'Vakiotiedot 10 ruutua VB (2 h / ruutu) 20/170_64'
'Vakiotiedot 10 ruutua VB (3 h / ruutu) 30/170_65'
Aug 3
Aug 10
Aug 17
Aug 24
Aug 31
Sep 7
Suunnitelma
Toteuma
Completed %
0
50
100
Tasks finished
80
60
40
20
0
'Suunn.'
'Tot.'
Tasks
300
200
100
0
'Tehtäviä'
'Tehtäviä valmiina'
'Tekemättä'
'Valmiit ajoissa'
'Valmiit myöhässä'
Accuracy of Planning
'Etuajassa' 38%
'Myöhässä' 40%
'Täsmälleen' 20%
Worktime distribution
11%
0%
88%
'Muu työ'
'Projektiin suunniteltu'
'Projektiin suunnittelematon'
Under estimated
Over estimated
Reporting
Warning: Applet Window

Figure 4: A project control panel in ViCA.

By clicking a bar in the Gantt chart, a new panel opens showing the daily effort put on the task by each employee and their daily estimation on effort required to finish the task. The first panel also shows statistical graphs of the project. It tells the number and percentage of tasks finished in time, early or late, and the degree of completion calculated based on finished tasks. The same company also utilizes data from version control system log files. The stability of design documents is observed by visualizing change rates to the documents.

There have been both technical and non-technical issues slowing down the deployment of the system. On the technical side there have been difficulties in finding resources for writing all the required EDAMs in the companies. Therefore we started the development of the most commonly used ones, i.e. CSVTool, EMIX and PMIX. Non-technical problems pertain mostly to the difficulty in defining the metrics that should be visualized in ViCA. Overhauling our method for defining metrics based on the strategic goals of the company requires commitment and effort. So far most metrics have been invented in a rather informal way. We have also seen that verifying the correctness of the visualizations is important, because writing erroneous database queries may happen even for an advanced user.

4. Conclusions

In this paper we have presented an architecture for combining and utilizing data from different sources, and a tool set based on this architecture. The architecture allows adding new data transfer modules (EDAMs) and client applications, and writing them on practically any platform using any programming tool. However, providing users with plug-and-play configurable EDAMs is essential to decrease the effort required in deploying this kind of a system. Other advantages of the architecture are that it hides the database interface except from SQL queries from clients, and provides flexible access control. Our implementation contains a very capable, web-based visualization tool (ViCA) and some EDAMs for collecting data from common office applications.

The future development work will mostly focus on ViCA. We need to develop further the idea of visualization libraries and provide more of them. Visualizing target values and alarm limits for each metric in a chart is also under development. Concerning EDAMs we must investigate integrating third-party data collection solutions into our system to enhance the set of plug-and-play EDAMs.

The advantages of the use of ViCA are further investigated, in addition to our pilot companies, in an educational environment. We will give our system to half of the twenty groups in a software project course in the Helsinki University of Technology. They can use our system for project management purposes during a one-year software development project. The effect of using ViCA is evaluated by interviews, usage logs, and comparisons with the groups not allowed to use the system.

5. References

[1] Lassenius, Casper, Kristian Rautiainen, Maarit Nissinen, and Reijo Sulonen. 1998. The Interactive Goal Panel: A Methodology for Aligning R&D Activities with Corporate Strategy. In Proceedings of the 1998 IEEE Conference on Engineering Management.

[2] Lassenius, Casper, Kristian Rautiainen. An Incremental Approach for Improving the Controllability of Product Development. In Proceeding of the 6th International Product Development Management Conference. 1999.

[3] Devlin, Barry. 1997. Data Warehouse: From Architecture to Implementation. Addison-Wesley.

[4] Cooper, R. G. 1993. Winning at New Products. 2nd ed. Reading, MA, USA: Addison-Wesley.

[5] Berners-Lee T., R. Fielding and H. Frystyk. May 1996. "Hypertext Transfer Protocol – HTTP/1.0", RFC 1945, MIT/LCS, UC Irvine.

[6] Sybase Inc. October 1998. Sybase Enterprise Data Studio Feature Guide.