Active Use of a Repository in Production

Jürgen Kessler[*]

***Abstract**: There are many buzzwords used like dictionary, encyclopedia, repository in context with meta information systems. In this article we want to give a useful classification for those technical terms. In a second step some basic concepts for this work (e. g. what means active use ?) are discussed. Requirements and possible problems which will affect an active use of a repository in production is another main aspect of this work. We will show possible solutions of those points: We define the kind of data which should be stored in a repository. Regarding those special requirements of the production environment we analyze views of a selected repository metamodel. At the end the results are reviewed for their practical relevance.*

***Key Words:** Active Use, dictionary, encyclopedia, repository, meta information systems, production, repository metamodel, meta directory*

INTRODUCTION

Different software systems are used by the generic term "meta information systems" for development and documentation of business application software systems. After having released IBM's product "Repository Manager MVS" in the early nineties, the name "repository" is very popular to professional circles for meta information systems. Since this time different scientific approaches have pursued to clearly define the term repository for its special character. Generally two categories can be distinguished:

The historical approach and the way to define a repository by means of special properties.

THE HISTORICAL APPROACH

With this basic approach the evolving history from a simple data dictionary, which only stores definitions and descriptions for real business data, to a dictionary, in which the same data plus meta data for programs is kept, to a so-called encyclopedia, in which data is not only stored from the technical view, to current systems, called repositories, in which all kind of metadata, that means for data, programs, configuration (hardware and systemsoftware) and organisation (user-/ service organisation) and all stages of software development are covered [Habermann 1993].

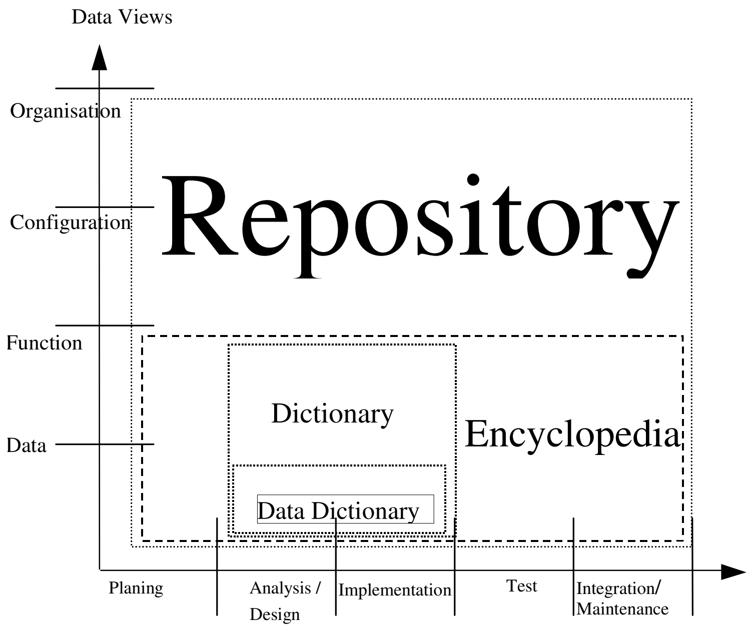

Figure 1: Stages of development towards a repository

REPOSITORY BY MEANS OF SPECIAL PROPERTIES

Another way to describe a repository is to define it by its special features, which are mainly vendor independence of development tools, extensibility of the meta model or versioning of stored objects [Trash 2001] [Tannenbaum 2001].

WORKING DEFINITION

In our opinion both approaches are useful and needed for specifying a repository. In this way we speak of a repository if those features are given and the system covers all those kinds of metadata and stages described above. Anyway, a repository as in its role of a meta information system must consist of a database system and user interface to interact with the user. In this way we use repository and repository system synonymously.

AIM AND STRUCTURE OF THIS WORK

The aim of this work is to show the possibilities for active use of a repository in the production environment. First of all it is necessary to clarify some technical terms, which are key to this area of research. Because of the importance of active use for this work we will further take a closer look on the development processes from the view of the user. Moreover, it is needed to define the kind of data (object vs. meta data), which should be defined by the repository metamodel.

On basis of the requirements for the production environment, it has to be analysed what views of the metamodel (including its underlying data) are qualified for use in production. Last but not least, we will verify this approach with real IT business examples.

TECHNICAL TERMS

PRODUCTION ENVIRONMENT

Applications running in the production environment are built for the support of business processes. This environment should be strictly separated from the test and development environment. Application objects should pass a special automatized process (release process) in order to be migrated from test to production environment. The production stage can be used synonymously to the maintenance stage, shown in figure 1.

REPOSITORY METAMODEL

In general, a repository metamodel is a data model for the repository database [Pottinger 2003]. Views and methods are defined on meta level for the application model level [Scheer 2000] (cf. figure 2).

ACTIVE USE OF A REPOSITORY

First of all, active use of a repository needs a connection between the repository and other systems for using directly repository contents for development tasks. It is not necessary to look on further systems at first – in principle this prerequisite is already needed when looking at the internal communication between repository server and –client. Let's take a look on an concrete product (software configuration tool ClearCase) from the view of the user. [White 2000]. In this context, the concept of a view is of great relevance:

If you want to access files, stored in the repository, you have to create a view. This view is a representation of a directory, addressing specific versions of elements in the repository (cf. figure 3). The selection of these objects is done rule-based by means of a configuration specification (config spec), which acts as a filter. Furtheron, it provides a workspace where you can work on assignments in isolation from other developers. For example, you work on web-documents, no one can see the changes until you check in.

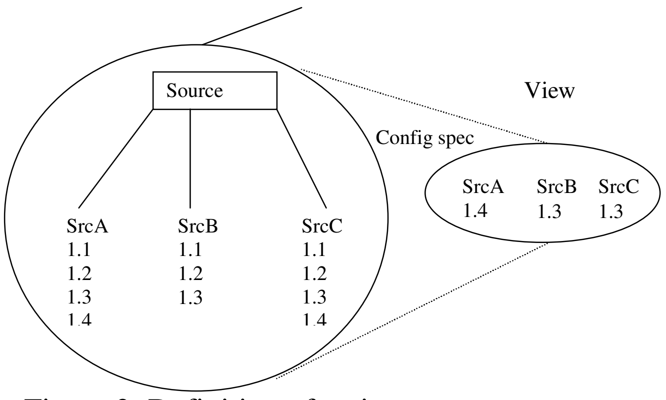

Figure 2: Definition of a view

ClearCase provides two different views:

Snapshot view: elements are copied to the local workspace. An update operation is needed to see the latest versions of elements in the view. In this case active use is not available due to non-permanent connection between the repository server and –client.

Dynamic view: provides immediate, transparent access to data stored in the repository. The latest versions of elements can be captured in a dynamic view.

Dynamic views correspond to above definition for active use.

We will demonstrate on two examples the differences between dynamic views and snapshot views. First the impact on checkin/ checkout operations in a concurrent development environment is shown (cf. figure 3).

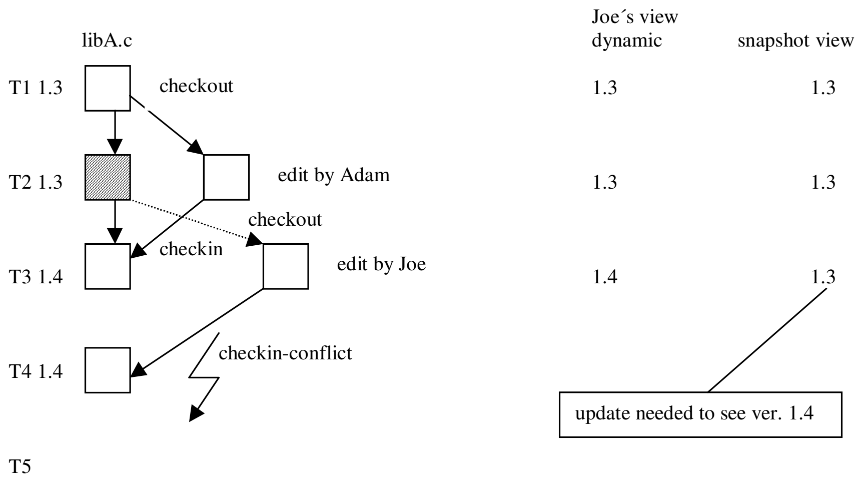

Figure 3: Checkout/ checkin operations

In this case the advantage of a dynamic view is that you always have an oversight of the current state of your elements in view. Developer Joe could avoid this conflict by for example waiving his active checkout of version 1.3 and start a new checkout of the new version 1.4.

The second example focuses on the config spec rules for views.

The following rules in the config spec are given:

element * checkedout

element * /main/latest

With these rules versions of elements are selected either if checkedout or having the latest version (cf. figure 4).

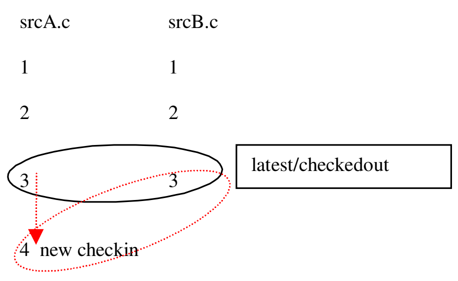

Figure 4: Dynamic/ snapshot views defined by one config spec

We presented on two examples the advantages of active use (dynamic views) from the user perspective in the development stage: There is always up-to-date information available to be able to meet the right decision. These examples provide another important aspect, which is not covered yet by our above definition: Real-time data updates of the repository-server by checkin/ checkout operations. For our later work, it is obligatory to make sure that data used from the repository in production environment is up-to-date by automatic triggered processes. We have to check if this requirement has to be met by real-time updates.

At this point we stress the neighbourhood to the concept of active database systems/ active repository systems which goes much further. These systems react (e.g. start updates) rule-based to occurred events (e.g. data changes) automatically [Schrefl 2001].

After all, we can distinguish between three kinds of use: Active use of a repository in development, at compile/ generation time and execution time [Ortner 1993]. The first two ones are usually related to the development/ test stage, while the last one is associated with the production stage [SAP2004]. In our understanding all three kinds of active use are covered by this work. It is necessary to have active use at execution time for production purposes (permanent online connection).

PROBLEMS

Thinking about the role of a repository - usually it is not associated with storing business data like information about customers or articles but to have a storage for meta data. Meta data describe and define business data [Ortner 1993] [SAP 2004]. In professional circles this system was early assisted by some standards, like the ANSI IRDS four level architecture or the ISO/ IEC Standard, which is quite the same [ANSI IRDS 1989] [ISO/ IEC 1990] [ISO/ IEC 2004].

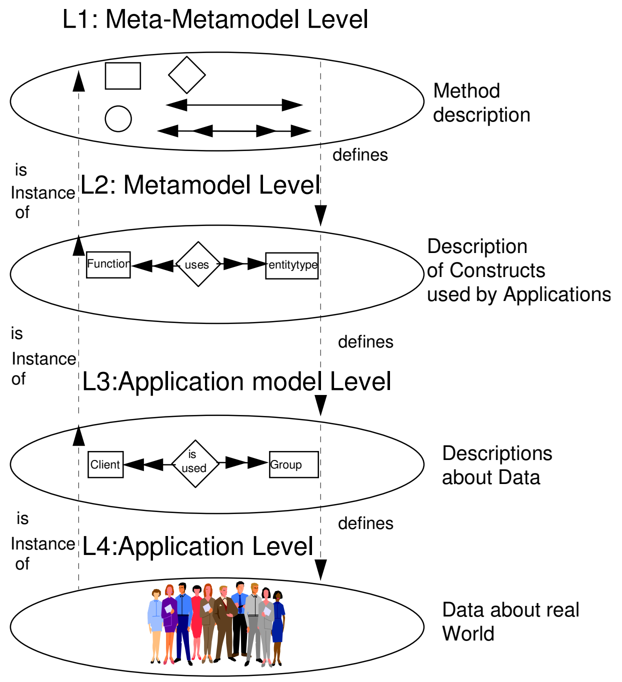

Figure 5: The 4-Level Architecture of ISO/ IEC

Detailed analysis leads to the insights that there is not only object data on L4. It is also possible on L3: You can have meta data ("customer") as an instance of "entity type" from L2 and you can have the object data "Juergen Kessler" as an instance of "user"(as a meta entity type of view organisation) from L2.

Special requirements are needed when repository data is used for production purposes:

Data is needed directly for execution time.

Connecting a repository to production environment causes dependences. That means systems which use data directly from the repository cannot work if the repository is "down".

You have to restrict the scope of data, which is probably relevant for this research. Dealing with a extended metamodel, you can maybe think about integrating all tool data (from databases, operating systems, tools for administrating programm sources,....) and their definitions (L2) into the repository to be use actively in production. How does this work ?

The further discussion will take place regarding this background.

SOLUTIONS

KIND OF DATA

The general statement that only meta data is stored by a repository cannot be confirmed because of the detailed analysis done for the four-level architecture. For our work we have to modify this statement. The kind of data, stored by a repository can only be charaterized by its focus. It is not the real business but the IT infrastructure (data, programmes, organisation and configuration) of a company.

REQUIREMENTS FOR PRODUCTION

Data needed directly for execution time is a must for production. In our opinion it is sufficient to provide production environment with updated data at the right time. In this way you can think of a nightly file transfer from the repository to candidate applications in production. According to our above definition it is active use and dependence from a repository is not present anymore.

For several reasons (performance, software release updates, complexity,....) we do not think that integrating every tool holding meta data into the repository by means of metamodel integration is a practical application. Dealing with a extensible and complex repository metamodel needs a limitation, a scope. On one hand this is our view towards the kind of data (IT infrastructure, see above) and on the other hand we refer to a internationally well-known meta model, which is called ARIS [SCHEER 1999].

ANALYSIS-RESULTS

The analysis was done following the structure of ARIS that means views (data, programmes, organisation and control view) and their development stages (analysis, design and implementation) [Scheer 2000]. As a main result the control view is in focus. This view can be used for checking access rights. This function can be used either on base of design or implementation objects: it is dependent on the degree of realisation (1:1 or not).

PRACTICAL RELEVANCE

Data defined by the control view (especially: relation organisation with modules/ data on design-level) can be used for checking access rights. This function or better said this service has to be individually programmed or can be supplied by commercial products already available.

When you think about Windows2000 networks, the appropiate service is identified by the buzzword „active directory“ (AD) [Zubair 2000]. Talking about mainframe computers, you propably heard about RACF (resource access control facility) or newly called security server in case of IBM as vendor [IBM 2003].

Main common features of these products are:

- Access rights for users are usually organised by groups (in case of AD it's mandatory). This means you have to give rights to groups and connect users to these groups. It makes sense that a group is related to common tasks of the connected users, for example group „programmers" who all need the same files, programmes, databases and so on.
- The access checking service has its own integrated database, which means, the database is encrypted, capsulated and optimized for use of the service. In case of AD, it's a distributed database.
- Access checking on resources for users including user authorization (password handling) is not the only service of those systems. There is also a logging facility included which provides for retracking of all security relevant activities of every on-logged user in present or past.

For big companies which use those networks and mainframe computers, it's common to use these integrated security systems. In both examples, the access checking services are "on board". In case of AD, it is stricty advised to use it.

On the other hand, you've already got data for checking access rights, defined by the repository's meta schema and kept by the repository. So what to do ?

The only way to get those things together is to transfer data from the repository's database to the access checking services of your network systems and host computers.

This is normally possible through data exchange between these systems, e. g. via xml-file transfer [Bray 2004]. In this context you've got to consider that data exchange is not only from the repository to the access server but also vice versa. This is because of the daily business of the user help desk, e. g. new user, group not needed further, modified programme name and so on.

CONCLUSION

Compared to our definition for kinds of active use - it is active use of a repository in production at its lowest level. The requirement "updated data" mentioned there is fulfilled, although the repository is not directly used at execution time. In this respect, the repository acts as a meta directory for all access/directory services. More details for data matching mechanisms and data exchange service should be researched in a separate industrial paper.

REFERENCES

[1], ANSI IRDS 1989, American National Standard X3.138-1988: Information Resource Dictionary System (IRDS), American National Standard Institute, New York, 1989

[2], Bray 2004 Bray, T. Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergan, F., Extended Markup Language 1.0, (Third Edition), w3.org/TR/Rec-xml, 2004

[3), IBM 2003, IBM Publication, Resource Access Control Facility, www-1.ibm.com/servers/eserver/zseries/zos/racf, 2003

[4], ISO/ IEC 1990, ISO/ IEC Publication: ISO/ IEC 10027 Information Technology - Information Resource Dictionary System (IRDS) framework. Geneva: ISO/ IEC, 1990.

[5]. ISO/ IEC 2004, Meta Object Facility (MOF) Specification, ISO/ IEC 19502::2004, 2004

[6], Habermann 1993, Habermann, H. J., Leymann, F.: Repository. Introduction. Munich, Vienna: Oldenburg 1993

[7], Ortner 1993, Ortner Metainformationssystems, Course Material WS 93/94, Report 37-93-Constance 1993

[8], Pottinger, Pottinger, R. A., Bernstein, P. A., Merging Models Based On Given Correspondences, VLDB Conference, 2003, 826-873

[9], SAP2004, SAP, FAQ-ABAP/4 Dictionary, www.sappoint.com/faq/

[10], Scheer 1999, Scheer, A.-W., Business Process Frameworks, 3 rd Ed., 1999

[11], Scheer 2000,. Scheer, A.-W., ARIS Business Process Modeling, 3rd ed., 2000
[12], Schrefl 2001, Schrefl M., Bernauer M., Active XML Schemas, ER Conference, 2001
[13], Tannenbaum 2001, Tannenbaum, A., Metadata Solutions: using metamodels, repositories, XML and enterprise portals to generate information on demand, 2001
[14], Trash 2001, Trash, R., France, R. B., RIGR-A model based approach to Repository Management, www.cs.colostate.edu/~france/rigr.pdf, 2001
[15], White 2000, White, B., Software Configuration Management Strategies and Rational ClearCase, Addison-Wesley, 2000
[16], Zubair 2000] Zubair, A., Active Directory Overview, www.winnetmag.com/ articles/index.cfm?articleID=8178, 2000

ABOUT THE AUTHOR

Juergen Kessler, Department of Computer Science and Engineering, Univerzitni 8, CZ-306 14 PLZEN, Phone: +491719131053 E-mail: Juergen_kess@hotmail.com .