Improvement of a Configuration Management System

Frank Titze
CAD-UL AG
Lämmerweg 32
D-89079 Ulm
+49 7305 959 0
ftitze@cadul.com

ABSTRACT
The company CAD-UL AG develops software tools for embedded systems. Single tools as compilers, linkers and debuggers are offered as well as complete development tool chains for the software development process. In contrast to application software for personal computers, embedded systems require very specialized software of highly optimized and exhaustively tested code.

Since the previously existing configuration management was not efficient in comparison to the state-of-the-art in software engineering, an improvement was implemented by the introduction of a modern Configuration Management (*CM*) system [1].

In this presented paper, CAD-UL intends to show the results and the experiences of the European Systems & Software Initiative Process Improvement Experiment (ESSI-PIE) ICMS with the new configuration management system.

Keywords
Configuration Management, CVS, Client-Server, Customer, Call and Defect Database, Customer Web-Access to Defect-DB, Multiple Hosts Operating Systems, Multiple Target Systems, ODBC, Perl, Perl/TK, RCS

1 INTRODUCTION

The company CAD-UL AG with its headquarters at Ulm, Germany, offers comprehensive, customer-based solutions for hard- and software development. CAD-UL is organized in three profit centers with special product and service supplies:

- Electronic Services
  CAE/CAD/CAM-Service (Computer Aided Engineering / Design / Manufacturing) as pure service in the domain of the production of printed circuit boards
- Ariadne EDA System
  (Electronic Design Automation). This CAD program has been distributed since 1987.
- **Embedded Tools -** as the location of the ESSI-PIE
  Integrated development environments tool chains (*Fig. 1: Tool Chain*), consisting of C and C++ compilers, assemblers, linkers and high-level programming debuggers for *Motorola 68K*, *Intel 80x86 real* and *protected mode embedded systems* (*Fig. 2: Host-Target Matrix*).

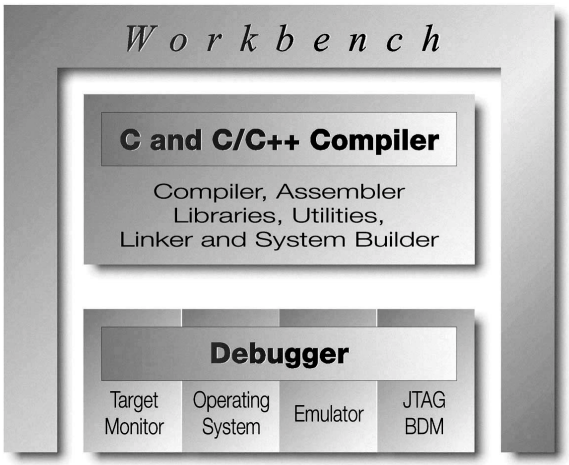

Fig. 1: Tool Chain

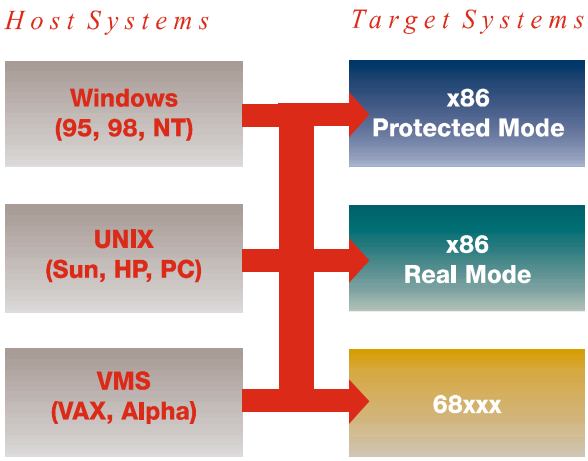

Fig. 2: Host-Target Matrix

ICSE 2000, Limerick, Ireland

The tool chains are available on different host platforms[1] and are each specialized for one of numerous target systems. CAD-UL tools are used for multimedia, telecommunications, computers and peripherals, automation, avionics, defence, medical and many other applications. The company has established branches in Scottsdale, USA, Birmingham, Great Britain, and Stockholm, Sweden.

2 Initial CM Practices

The initial CM practices consisted of two principles in handling of software sources and a distribution tree for the generated distributions.

Additionally there is a world wide replicated *DB* (*CDB*) on a Microsoft (*MS*) Windows NT (*MS-WinNT*) server used, where detected software bugs, changed requirements and new features are collected and stored for consideration in new releases. Customer inputs to the CAD-UL service hotline are also entered into the *CDB*.

Customer data is stored in a separate *Customer DB*.

The source were generally stored under a *UNIX* server and the generation, build and test processes were done under *UNIX*, *VAX-VMS* and *MS-WindowsNT/95/98* systems.

Some of the products were under revision control of a simply set up and used the Revision Control System (*RCS)* [4] with script boundary.

The other products, like the debugger, as location of the base line project for this CM project, were kept in separate file structures. The source code and the production location of each main release was stored separately. Based on this, minor bug-fixes, called patch releases, were carried out on the related major version of the source code. The patches were implemented incrementally, thus a patch release included all former bug fixes of that major release. The different major versions were stored as directory trees on the server
(*Fig. 3: Debugger Trees*).

1. *UNIX* operating systems from different manufacturers. (SUN, HP, Linux). *VAX-VMS* from (Digital Inc.). *Microsoft WindowsNT/95/98* operating systems

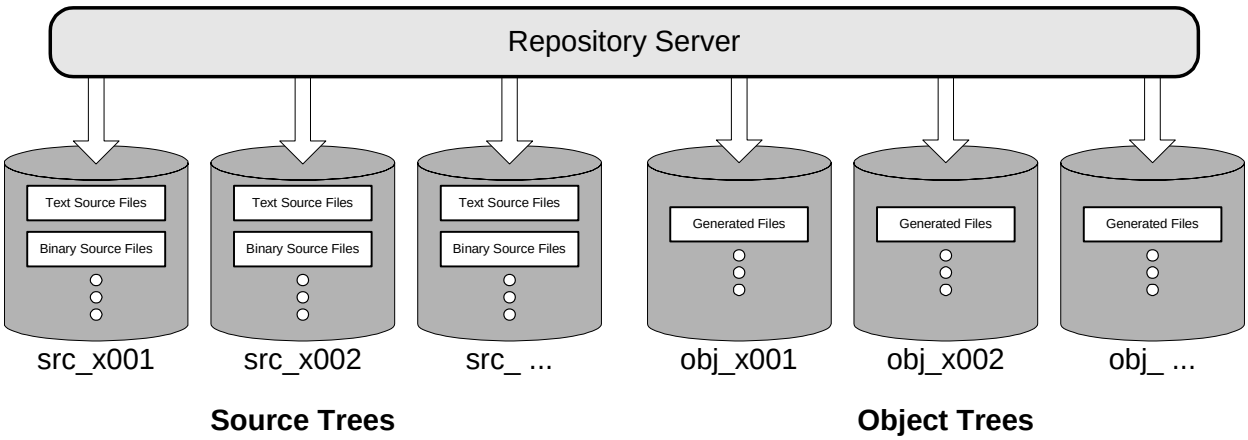

Fig. 3: Debugger Trees

3 OBJECTIVES

Technical Objectives

The old *CM* system has several weaknesses. Out of these weaknesses, came a list of technical benefits objectives for the improved configuration management created:

- Precise control of the effects of modifications or redesigns in all product versions. [PRECISE CONTROL]
- Reliable control over the consistency of the generated software configurations. [CONSISTENCY]
- Online access to any sub-release of any previously generated software configuration. For a specific base product, only the latest patch version of a major release was made available. [ONLINE ACCESS]
- Improved handling of the configuration sets for each host. Keeping the configuration files consistent was very error-prone. A change (or add) of a configuration switch automatically needed the migration into the highest (latest) major version tree. [HOST SETS]
- Improved documentation on needed changes for a particular action compared to CAD-UL's Customer, Call and Defect Database (*DB*) (*CDB*)[2]. [DOCUMENTATION]
- Creation of an efficient bug-fixing with reduced turnaround time for patches. Any bug-fix could be applied only to the latest patch release. This implied that the newly generated patch version would incorporate all patches between the version where the problem was detected, and the last fixed problem. It should be possible to fix only a specific problem of an old patch release. [TURNAROUND TIME]
- Establishing a central management of all configuration related data. [CENTRAL MANAGEMENT]

2. Former a stand-alone Change-Request *DB* and a stand-alone Customer *DB*.

- Creation an uniform production environment for all products to minimize administration overhead. [UNIFORM PRODUCTION ENVIRONMENT]
- Improvement of the control and documentation of software changes and generated releases for books, release notes and for the IS0-9001 project documentation. [IS0-9001]
- Creation of web access to the Client-Server (*CS*) Customer, Call and Defect database (*CDB*)[1]. [WEB ACCESS]
- Query and Report functionality in *DB* from the customers (Sales section) over defects (Support section) and releases to the source revisions (Support, development, QM section). [QUERY AND REPORT]
- Integration of the Customer, Call and Defect *DB* (*CDB*). [INTEGRATION]
- One single definition point in the *CM* system for naming of packages of products modules, etc. which defines. [ONE DEFINITION POINT]
- Server-Server coupling and strict Client-Server (*CS*) architecture. [ARCHITECTURE]
- No modification of the base tools GNU[2] Concurrent Versions Systems[3] (*CVS*) [2] and the customer information system from Platinum Software Corp. (*Clientele 3.0*)[4]). [NO MODIFICATION]

Social Objectives
The technical benefits should also have an impact on the motivation and satisfaction of the employees involved in the improved *CM*:

- Reduced efforts for administrative actions for the software development process. [ADMINISTRATION]
- More time for work related to technical problems. [MORE TIME]
- Benefits of the increasing understanding of state-of-the-art software development techniques and strategies. [BENEFITS]

Conservative Objectives
The old *CM* system also has strengths, which should be still remain:

- The old system was very flexible for modification. Due to the fact that it was based on storing different versions as different directory trees, creating a new version was quite simple. [FLEXIBLE]
- Creating a new configuration was done by simply copying and modifying build scripts. These script languages were simple and well-known and, therefore, don't require special training efforts. [SIMPLE]
- Because of the fact that no commercial *CM* tool was used, there are no restrictions affecting the software (*SW*) production process. [NO RESTRICTIONS]

1. Added or refined while processing the project.
2. "GNU is Not *UNIX*" and stands for software under General Public Licence (*GPL*) of the Free Software Foundation.
3. An *CS* multi-user version control system under *GNU* licence for several operating systems at CAD-UL is used in *CS* configuration
4. Not possible to modify.

4 CM SYSTEM

System Overview
The System is based on two major tools. *CVS* (*CMS* and *CMC*) as base revision system and *Clientele* 3.0 as Customer, Call and Defect System (*CDB(s)* and *CDC(s)*) based on *MS-SQL-Server*[5] (*see Fig. 4: System Overview*).

Added to the concept are *PostgreSQL*[6] *DB*s, the Apache Web Server[7] [6], Adobe Acrobat 4.0[8], an *RSH*-Daemon[9] for Windows NT and the Emacs[10] [3] editor.

The glue-scripts, customizations, web server, (cgi-) scripts and the Graphical User Interface (*GUI*) are done with *Perl*[11] [5].

The *CS* connections are done with the internal *pserver* system of *CVS*, the *CS* system of *Clientele 3.0* and with *RSH*.

The *DB* access in the glue-scripts are done with *ODBC* to the *MS-SQL-Server* and with the *Perl* interface to the *PostgreSQL DB*.

5. Former based on *MS-Access* which was dropped because its performance was poor.
6. A Standard Query Language (*SQL*) *DB* system under a licence similar to *GPL* by the University of California.
7. A web server for *UNIX* and *MS-WinNT* under *GPL*.
8. Provides a portable document format.
9. Remote Shell Daemon for *MS-WinNT/95* version 1.3. A program running on *MS-WinNT/95* which allows to execute programs from remote e.g. *UNIX*, workstations with the *RSH* (Remote Shell) functionality under *GPL*.
10. A very powerful editor under *GNU* licence, which is available under most currently used operating systems with an internal customization scripting language based on Lisp
11. A very powerful scripting language which is available under most currently used operating systems for operating system access and text matching.

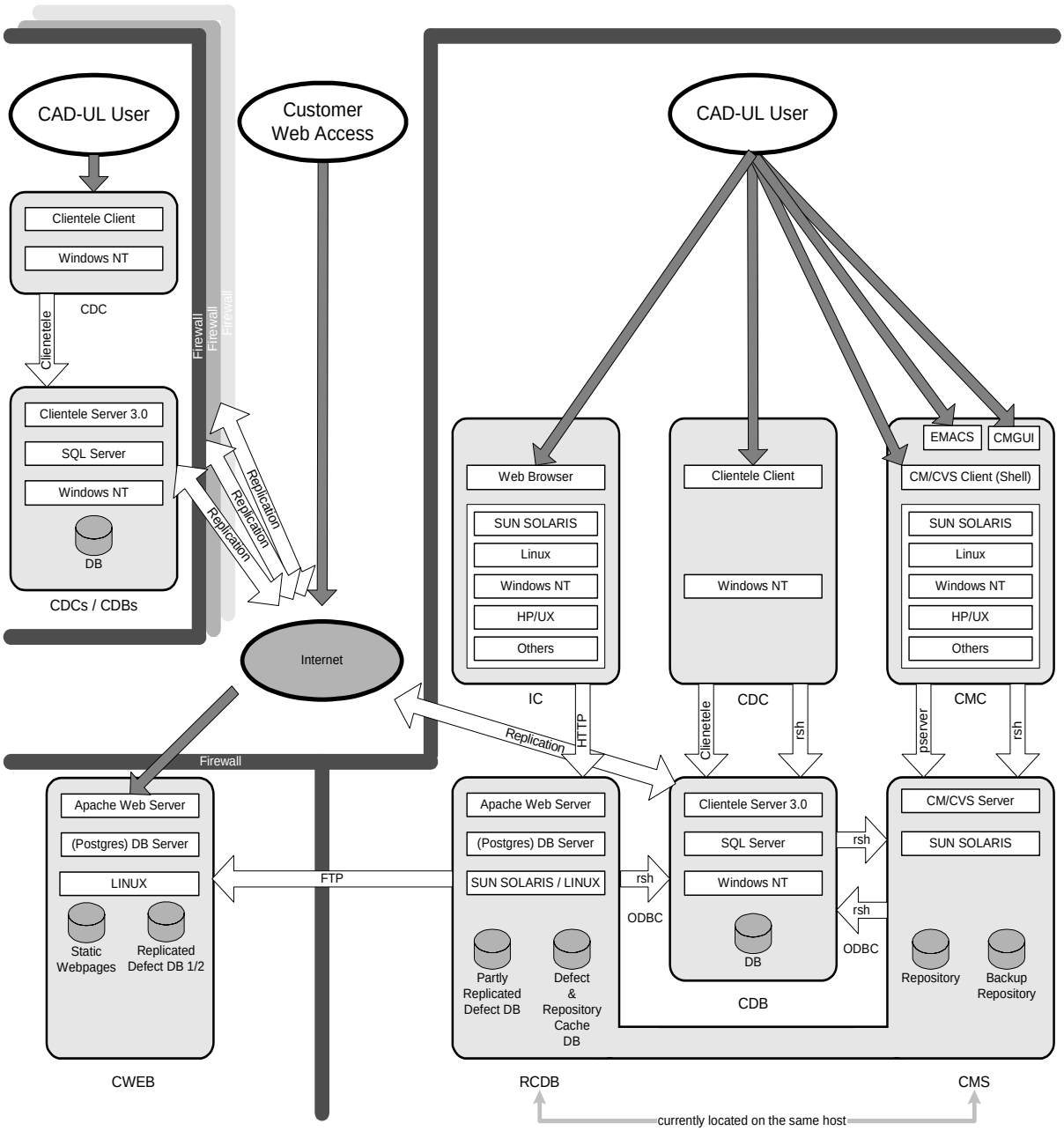

Fig. 4. System Overview

The main elements of the system are:

- *CMS*
  The customized *CVS CM* (repository) server with additional glue-scripts (e.g. with *RCS* calls). All sources are stored grouped by products in different trees in the repository (*see Fig. 6: Repository Layout*).
- *CDB*
  Customer, Call and Defect *DB* server. All information about customers, calls, defects, licences, products, packages, etc. are stored in an *SQL DB* which is replicated with and against the subsidiary organizations
- *CMC*
  The *CM* client (*UNIX* and *MS-WinNT*). Glue-script on top of the *CVS* client and as client to the additional glue-scripts of the server.
- *CDC*
  The Customer, Call and Defect *DB* client (*MS-WinNT*). Graphical User Interface (*GUI*) interface of the *CDB* server.
- *IC*
  The Web browser clients for the access of the *Defect & Repository Cache DB*. Inside this DB, the defect data and the log data of the repository are cached and linked to each other.

- *RCDB*
  The replicated Customer, Call and Defect *DB*. Inherits the Defect Table and abstracted licence information. Acts as data preparing interface for the customer access to the *DB*.
- *CWEB*
  The Web Site server of CAD-UL, with access to one of two copies of the replicated *CDB* (Defect Tables) which is transferred from the *RCDB* by File Transfer Protocol (*FTP*) for transferring files between hosts. The web interface connects always to the current one of the *DB*s.
- *CDBs*
  Replicated of the *CDB* at the subsidiaries.
- *CDCs*
  Replication of the *CDC*s at the subsidiaries.
- *CMGUI*
  A *GUI* front-end for the shell *CM* interface.
- *EMACS*
  The Emacs editor with a customized interface to the shell *CM* interface.

Version Control
The *CVS* version control system bases on the *RCS* repository text and binary files with a version information header.

Binary files (e.g. *GUI* Bitmaps) and text files (which aren't allowed to be modified) are also stored in the Repository with one difference. For each file there is a so called shadow-file generated and maintained by the *CM* system (*Fig. 5: Text and Binary Files – From step 1 to 2.*). This shadow-file inherits the version information header which is normally added to the text file. By including this shadow-file into the program sources, the version informations of binary files also are stored in the executable (library, etc.).

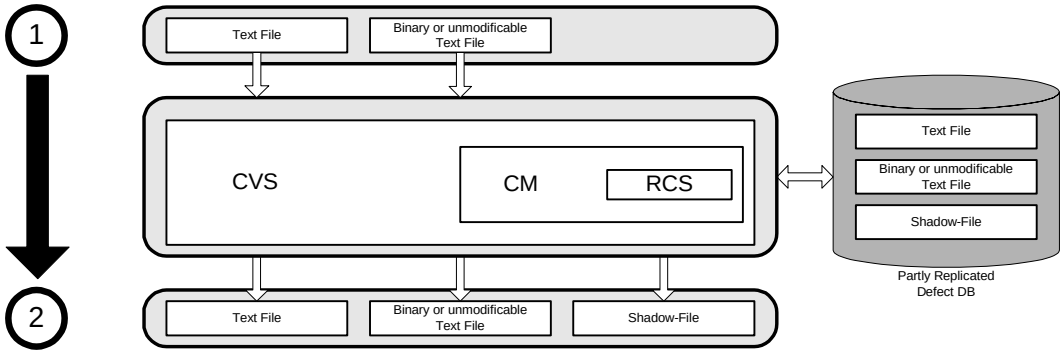

Fig. 5: Text and Binary Files

Release Management
In our first step of release management, (*see extensions: "Future Actions Planned"*), we mark (tag) the file versions of a release file set due to product, modules and target entries in the *CDB*. These marked file sets may will be retrieved later.

Interaction with the Call and Defect Management
Main point for this besides the technical infrastructure is the method to store the project name and optionally the defect number in the description of a new file version. On base of these entries, relations from the ISO-9001 project documentation and the *CDB* to the sources and back are possible.

Web Access
With the Web Access, the customer gets access to defects listed in the defect table of the *CDB*, which are related to his licensed tools.

This is done by transferring the replicated defect tables and licence informations to the *CWEB DB* which is generated in the *RCDB DB* system. It is planned to integrate the *RCDB DB* in the *CMS* server.

Web access is expected to be available for customers in spring 2000. Currently the web access is in test phase in CAD-UL.

Customer Sales and Support Management
With the link from the *CDB* to the *CMS* and with the queries inside the *CDB* we have now, e.g., the possibility to retrieve very easily which customer has which configuration of our products.

Integration of ISO-9001 Project Documentation
The integration of ISO-9001 project documentation is done with Adobe Acrobat (Version 4.0) which provides platform independency (*UNIX* and *MS-WinNT* and *MS-95/98* systems) and a digital signature functionality[1].

At least this digital signatures let us now integrate the project documentation in the *CM* system, by treating the

1. Currently only on *MS-WinNT/95/98* systems available.

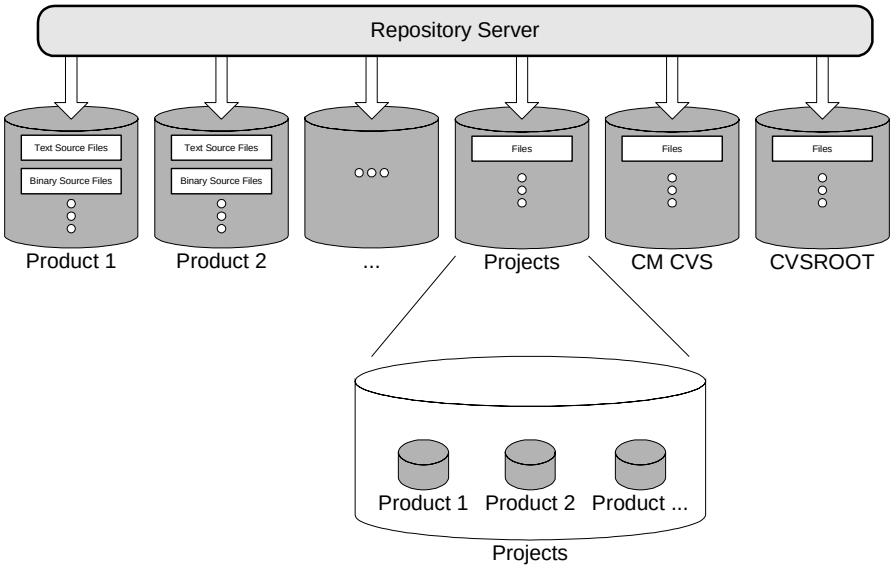

Fig. 6: Repository Layout

project documentation equal as source files and store them in the repository in a directory tree called Projects where all products are listed (*see Fig. 6: Repository Layout*). As one additional benefit to this we have now the possibility to send the documents in-house or even external per email.

The link from the sources to projects and back is done by the project name.

5 RESULTS

As a general result, it could be noted that the main achievements are qualitative with a long term characteristic which can't be measured easily.

Qualitative Results

In contrast to the initial CM system.

- *The modification of the sources are much better and with a much greater granularity and automatically documented.*
- *The modifications could be done much better distinguished from each other.*
- *There are lots of functionalities to retrieve information out of the CM system, the files and the generated products.*

Quantitative Results

Besides to these qualitative results were several measurements defined for being performed after and while transformating the base line project to the *CM* system.

Number of Change Requests / Year (long-term)

A quality effect can be measured by the number of change requests and by the number of patch releases of a specific product per year.

Results will not be available before end of 2000 or beginning of 2001

Number of Patch Releases of a Specific Product / Year (long-term)

Average Number of Defect Reports / Patch Releases (long-term)

An easy, automated process of generating a new software release tends to increase the number of releases. This is not intended because it increases the efforts in testing, verification and documentation.

Results will not be available before end of 2000 or begin of 2001

Transformation Process Effort (short-term)

Migration of a software product, from the old system, to the new *CM* system.

Our initial migration of the debugger requires one man day net time. The gross time was 3 man-days, because of bug-fixes and the glue-scripts. To build up the full functionality, there is an additional second step in migration[1] *necessary which is currently in progress.*

An additional initial migration of the assembler (a former RCS based project) needs 1/2 man day net time.

Generation Process Effort (short-term)

The effort required for generation of a new release based on a specific source version. The time from starting to finishing a release job, compared it to former generation processes.

A first generation process was done, but a clear quantitative result in contrast to the qualitative result can't be seen yet. Better results will be available after having passed more generations processes.

1. The source files are extended by a *version information header* (see section 3.1).

Modification Effort (short-term)
Time for implementing new features and bug-fixing activities, excluding the efforts for problem solving.

Similar to the "Generation Process Effort", see "Qualitative Results".

Reintegration (short-term)
Merging source-code revisions created in separate branches into the main branch.

Similar to the "Generation Process Effort", see "Qualitative Results".

Organization Impact
An impact on the organization can't be seen yet.

Culture Impact
A clear impact on culture or the existence of resistances can't be seen yet.

Skills Impact
Different styles of skills had been generated by the project. These skills open objectives of usages also after the ESSI-PIE project itself.

- *CM* systems
- *Perl*, *MS-Visual Basic*, Emacs Lisp, (Tcl) Perl/Tk[1], *PostgreSQL* and *Clientele* programming
- *CVS*, Apache Web Server and *PostgreSQL* configuration
- *UNIX-MS-WinNT* connectivity
- *DB* access with ODBC (with *Perl*)

6 RESULTS RELATED TO THE OBJECTIVES

Technical Objectives
- [CONSISTENCY]
  Consistency could be checked even in the end-product[2] at any time in the future.
- [ONLINE ACCESS]
  All major and patch releases and any intermediate file versions are accessible.
- [HOST SETS]
  The configurations are now handled equal to source files and are included in releases and patch releases.
- [DOCUMENTATION]
  These changes are automatically accessible via the defect number.
- [TURNAROUND TIME]
  Bug-fixes and patch release could be done on any major or patch release.
- [CENTRAL MANAGEMENT]
  Based on *CMS* and *CDB*.
- [UNIFORM PRODUCTION ENVIRONMENT]
  Will be done after migration of the other tools. Planned in Q1/2 of 2000.
- [ISO-9001]
  The ISO-9001 project documentation is now integrated in the *CM* system as Adobe Acrobat files.
- [WEB ACCESS]
  Started for internal test use in Dec. 1999. Customers are planned to have access in spring 2000.
- [QUERY AND REPORT]
  Could be done over the interfaces between the systems.
- [INTEGRATION]
  Queries are from the *CMS* to the *CDB* and back possible.
- [ONE DEFINITION POINT]
  In the *CDB* are the products, modules, etc. defined.
- [ARCHITECTURE]
  (*See Fig.4: System Overview*)
- [NO MODIFICATION]
  A modification of the base tools wasn't done.

Social Objectives
- [ADMINISTRATION]
  Will be achieved when all products are switched to the new CM by having one standardized system.
  Retrieving of version, configuration, defects, ISO9001-documentation is much more easily through relations between these parts of the CM system.
- [MORE TIME]
  More time for essential work is available on the fly of software development, generation and maintenance and not only or particularly on single special points of the processes.
- [BENEFITS]
  See "Skills Impact".

Conservative Objectives
- [FLEXIBLE]
  The creating of a new versions is now even simpler than before. All versions lie on each other in one tree.
- [SIMPLE]
  Still true with the improvement that build scripts are now handled exactly as other source files.
- [NO RESTRICTIONS]
  Still true.

7 CONCLUSION

Technological Aspects
- There is no alternative to this improved *CM* system in getting the overview over configurations, even when not every feature went in production.
- The *UNIX* philosophy in combining single tools with glue scripts is still a quite powerful way of generating functionality and interoperability.
- A commercial base tool in contrast to *CVS* would have provided a so-called high level system in form of a proprietary black box *DB* where everything happens inside. The disadvantages of such black box *DB*'s were

1. A module for the extension of Perl with the Tk GUI interpreter.
2. For text source files. Binary files, production scripts, etc. could (and will) be added later (*shadow files*).

more than once shown by other tools used by CAD-UL. It can be regarded an advantage that in the case of having problems with a tool, a software engineer has a chance to look or modify the tool itself instead of depending on the support of the manufacturer.

- *Perl* is a much more powerful and easy to use tool than e.g the particularly tested *MS-Visual Basic*.

Business Aspects

- With the original selected *CM* Tool ClearCase (from RATIONAL Software Corporation), it would have been difficult to expand the results to the whole company because of the financial impact of the licence and additional hardware costs besides the technical described.

Strengths and Weaknesses of the Experiment

- The strength of the experiment is that CAD-UL will get a *CM* solution of a style and complexity that will create everyday benefits in technical and human aspects with positive effect of the business goals.
- The *CM* system open to extend inside (distribution repository) and outside the software engineering process like, e.g., marketing or to other profit centers.
- The most important weakness at the beginning , depending on the point of view, while performing the project program was that is wasn't calculated that the market leader ClearCase (RATIONAL Software Corporation) wasn't able to support at least CAD-UL's most important host systems.
- Another "weakness" is that the *CM* system was originally planned more as a stand-alone solution in the development division and the interaction with the defect and customer data weren't planned in detail. This was due to the situation in 1998, where e.g. only a remote unacceptable powerless defect *DB* exists and the customer data were stored in an other *DB*.

8 Future Actions Planned

- Transformation of all other tools into the new *CM* system.
- Transformation of the product user documentation into the new *CM* system.
- Building up a second repository for the distributions (*CMD*) and connecting this repository to the *CMS-CDB* system (*see Fig. 7: Connecting a distribution repository to the system - Compare with Fig. 4*).

Future Options

- Remote access from other locations, e.g from the subsidiaries.

REFERENCES

Novice Readers

1. Balzert, H. Lehrbuch der Software-Technik : Software Management,Software-Qualitäts-sicherung, Unternehmensmodellierung,Spektrum Akad. Verlag, 1998

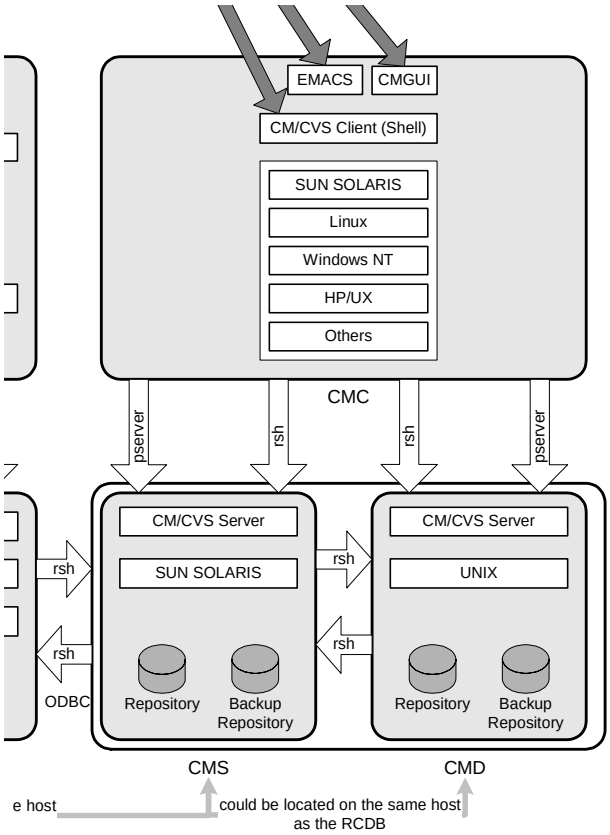

Fig 7. Connecting a distribution repository to the system - Compare with the right half of Fig. 4

Other Readers

2. Cederqvist, P. Version Management with CVS - for CVS 1.10, Signum Support AB, Available at <http://www.cyclic.com>, 1998
3. Glickstein, B. Writing GNU Emacs Extensions, O'Reilly & Associates Inc., United States of America, 1997
4. Herold, H., Meyer, M. SCCS and RCS - Versionsverwaltung unter UNIX,Addison-Wesley, Germany, 1995
5. Siever, E., Spainhour, S., Patwardhan, N. Perl in Nutshell, O'Reiglly, United States of America, 1999
6. Stein, L. D. How to Set Up and Maintain a Web Site, 2. Edition, Addison-Wesley, United States of America, 1997
7. Wall, L., Schwartz, R. L. Programmieren in Perl, Hanser, Germany, 1993