A Case Study in Repository Selection for a distributed Software Engineering Environment

Jan Neuhaus
Fraunhofer-Institut für Software- und Systemtechnik ISST
PF 52 01 30
D-44207 Dortmund, Germany
neuhaus@do.isst.fhg.de

Wolfgang Janzen
Oberfinanzdirektion Hannover
Lavesallee 10
D-30169 Hannover, Germany

Andreas Bäcker
Partner Consult
Wandsbeker Zollstraße 19
D-22041 Hamburg, Germany

Abstract

The German Ministers of Finances have set up a large scale project for the development of a nationwide uniform tax management system. One phase of the initiation of this large software development effort was the construction of a distributed software engineering environment. The Fraunhofer ISST and Partner Consult were involved in the selection process of the repository and the change and configuration management tool for this environment. In this paper we describe the selection process for the repository. We present a structured approach which tries to minimize the selection costs and we discuss some of the lessons we learned about the process itself and the state of the market in repository technology.

1. Introduction

There are a lot of problems in constructing a software engineering environment, but it is also very difficult to select or compose such an environment for the support of a strategic software development project. In the context of this paper the main selection criteria are first whether it is effective, i.e. does it fulfill the needs of the software developer and the project management, and second whether it is supported well now and in the future. The software engineering environment is one important cornerstone of a very complex system. It is well known that nobody should build a house on sandy ground, thus the selection process is very important and heavily influences the probability of the success of the entire project.

The German Ministers of Finances are preparing a large software development effort, called FISCUS, to completely redesign and re-implement most software components involved in the tax management process. Modern techniques will be applied to construct a layered basis architecture to enable the large set of applications to work together in a distributed multi-user environment. The software is supposed to be developed mainly with resources of the financial administrations of the "Bundesländer". Hence, a distributed software engineering environment has to be instantiated. We participated in an early phase of the FISCUS project, in which the tool selection was the main goal. It was already decided that the environment will be composed of different tools from different vendors and multiple groups were assigned the selection of the components. In this paper we will concentrate on the repository selection process.

Before the process is explained in detail in section 2, we briefly describe the main requirements and give some pointers to related work. In section 3 we summarize what we learned about the process, followed by a section concentrating on our experience with the repositories we evaluated.

1.1. The requirements

The requirements analysis is an important first step in any evaluation process. We will not discuss all the requirements on a repository for a software engineering environment, which can be derived for example from [9], but only stress the special aspects of the FISCUS project. Some of these were not finally set, but we identified the following requirements as a basis for the evaluation.

1.1.1. Hard- and software infrastructure. Most of the workstations in the FISCUS development environment use Windows/NT as operating system. The servers will be Unix machines, using AIX or HP/UX. In our test scenarios we limited ourselves to AIX. There will be more than 16 locations distributed all over Germany which are interconnected via ISDN lines. This technology will probably change over time, but it will limit the bandwidth to about 64 Kbits/sec for some nodes at least for the first project phase.

1.1.2. Preselected tools. Some tools which should be incorporated into the environment had already been fixed. For example, there has been a decision that Paradigm Plus will be used for the analysis phase. This tool stores its data in an object-oriented database. As an integration facility the tool offers sufficient import and export capabilities.

Throughout the evaluation process there was no decision, concerning whether the software should be written in Smalltalk [12, 11] or in C++ [10], or which C++-compiler will be used.

1.1.3. Administrative requirements. It is very important that the software development to be performed is in the public interest. Faults in the software, whether produced by humans or by tool failures, can have severe impacts. So one important requirement was the security of the system, and another one the possibility to install a revision secure environment. This means that for any action or any change that has been made it must always be possible to detect: what has been done, who has done it, when was it done and why was it done. This can only be achieved with a secure system which prevents any illegal modification of the data.

1.1.4. Maintenance. The FISCUS project has an estimated duration of 10 years. This implies that maintenance of the software engineering environment is a crucial problem. Only products which will probably be on the market for a long time can be chosen. Furthermore, there will be no large staff to support the environment itself. This has to be done mainly by the tool supplier, thus requiring a very capable and reactive support staff.

1.1.5. Tasks of the repository. The tasks for the repository in the FISCUS environment are twofold. On the one hand it was estimated that only the elaborated data dictionary functions of a repository are able to form the basis for the cooperative design efforts and later for the reuse management. On the other hand the distribution of the development team will need some facilities to exchange information over a common infrastructure and a repository can provide this, too. It was not the primary goal to find tools that store all their data in a single repository, achieving full data integration. This was considered impossible because tools have already been selected which use different data storage mechanisms.

1.1.6. User interface and documentation. Most users of the software engineering environment will be recruited from the current staff. This is necessary to retain the application domain knowledge needed but will limit the technical background of the users of the environment. Therefore, the documentation, the user interface, and the training should be easy to understand and available in German.

1.2. Related work

There is some work on repository selection published in the literature, but in most cases the results are not very useful for the public. It is nearly impossible to publish work revealing detailed information about capabilities of commercial tools. Most license agreements simply disallow the publication of any information obtained by the use of the tool. Furthermore, in most cases the applied requirements are so specific that they make the transfer of test results nearly useless. This is another reason why such reports are not so wide spread. Our paper outlines a generic process, and explains its application in a real world project. It is intended that other people are enabled to perform their own evaluation with their own set of requirements.

There was quite a similar task in the context of the ATMOSPHERE project [8]. Their evaluation concentrated mainly on technical features and completely omitted the support, the stability of the vendor and other more organizational but nevertheless important aspects.

We hold the view that benchmarking is only a small piece in the selection puzzle, but there is a lot of work done in this special area [16, 6, 7, 3, 4, 1, 18, 2].

2. The evaluation process

In addition to all the requirements sketched above, there was one more important constraint. The evaluation was considered important, but nevertheless, the costs should be kept as low as possible.

To achieve this goal a multiple stage process was defined. Every stage went more into the details of the products and consumed more time. To reduce the needed resources, a sieve approach was chosen. At the end of every stage a preselection was made and tools which were recognized as to be inappropriate according to our previously defined selection criteria were excluded from the tests.

There is one problem with the sieve approach. If the raster is badly chosen, not a single tool will survive. This is an acceptable result from the theoretical point of view, but is void in the context of a tool selection process. To avoid this problem we defined only a small number of hard criteria and based the final decision about the exclusion of a tool on the comparative results of all tools.

Figure 1 shows the different phases of the selection process, which will be explained in the following sections.

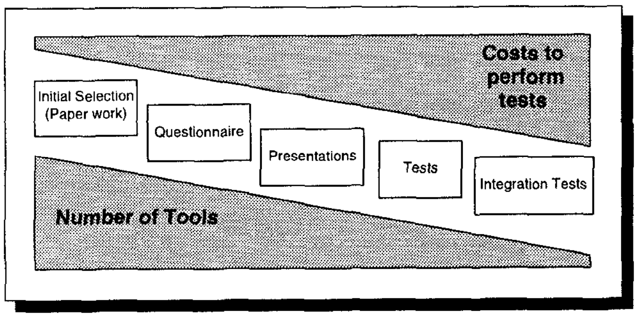

Figure 1. The phases of the repository selection process

2.1. The paper work

The first step totally depended on paper work. To guide the further process a requirements definition, more complete than the short sketch above, was developed. Based on a market survey four repositories were preselected which seemed to be promising candidates for the FISCUS software engineering environment.

2.2. The questionnaires

The questionnaires were derived from the requirements definition and mapped these onto possible product and company features. Figure 2 shows an abbreviated table of contents. It was prepared as an editable document and sent to the selected vendors giving them a deadline for the submission of the completed text.

The questions started with some general information about the company, their representative in Germany and the product context. In the next part we asked for reference projects and customers which were possible candidates for a visit to a reference installation. These parts were important because the strategic position of the company in Germany and the position of the product inside their portfolio indicates the possibility of a long term cooperation.

The next sections in the questionnaire were directed towards the product itself. A first block collected the information about the availability of the product and the product family, if the repository was part of a larger collection of tools. Within the second block we checked for the applicable standards and for the available interfaces. Next we continued with questions about the information model of the repository. Furthermore, we analyzed whether the repository has a meta model and if it is able to support distribution and replication.

The supported operation systems and network protocols formed another topic before the features of the product were checked in more detail. First the meta model was checked for modelling capabilities and administration features. The standard database features like transactions, concurrency, security, logging, performance characteristics, 24 hour availability, and backup and recovery features were requested next. The repository features were partitioned into the functions of the graphical user interface, the retrieval mechanisms, the impact analysis, workflow support and the support for reuse.

The questionnaires were analyzed and ranked according to a numerical scale. The numbers were based on a rating in the range of 0 to 5 and then multiplied by a factor repre-

1. Supplier
   1.1 Name, representatives in Germany, contact person, hotline support
   1.2 Other products, product family
2. References
3. Product
   3.1 Availability of the product, the product family
   3.2 Applicable repository standards, Interfaces (API, SQL, CASE-Tools, ...)
4. Information model
   4.1 General information model
   4.2 Supported development methods
   4.3 Distribution
   4.4 Replication
5. Operating systems (platforms and communication facilities)
6. Meta model
   6.1 Object-oriented features, Integrity constraints, Complex objects, Relations, Naming
   6.2 Administration (graphical interface, versioning, accounting)
   6.3 Installation
7. Database features
   7.1 Backup and recovery, Availability, Transactions, Concurrency
   7.2 Security, Authorization, Logging
   7.3 Performance
8. Repository features
   8.1 Graphical user interface
   8.2 Query capabilities
   8.3 Impact analysis
   8.4 Workflow support
   8.5 Reuse support

Figure 2. Abbreviated contents of the questionnaire

senting the importance of the feature. The results lead us to the exclusion of one of the tools because it violated an important criteria: the availability of the product in Germany.

The response time of the company and how they filled in the form also gave some indication about their attitude towards their customers.

It was important that a detailed test plan existed before the first tool was installed in our test scenario. A lot of work was put into this test plan which will be explained briefly in section 2.4, but is too complex to be included in this paper. This test plan had to be worked out before the first presentation started and was done in parallel to the evaluations. After the presentations it was very important that everything was ready and defined, because the vendors tend to blame any fault or misunderstanding on the testers.

2.3. The presentations

The three vendors whose tools survived the first phase, and one that joined in late, were invited to presentations. They all got the same letter stating a set of questions which we collected after the evaluation of the questionnaires. They were given an agenda which allocated 45 minutes to a presentation of the firm and their strategy, and 90 minutes to give a live presentation of the repository, if possible answering most of the proposed questions, and 90 minutes for questions and further discussions. It was clearly stated that we would like to reuse the scenario of this presentations for our own first tests and that a test installation was needed.

In spite of our detailed requirements of the presentations, most of the information could only be gathered in the discussion part. Some companies had not been able to give live presentations of their tools and only showed slides. The questions had been prepared by us using a word processor and the answers were noted directly on laptops, by different persons. The results were discussed and merged later.

After this phase, we excluded one further candidate because it became evident that their tool was just ported from the mainframe world and that the Windows/NT platform was not supported well enough.

All the others were given a detailed hard- and software description of the test scenario and were asked to prepare offers for test installations.

2.4. The testing spiral

The main part of the tests resembled a spiral because many topics were dealt with different times but on several levels. As shown in figure 3 we started with an observation of the installation process. Next we looked through the documentation as a special test case. We planned to play with any delivered tutorial, but no company supplied a ready-to-go tutorial, so we started directly to perform tests with the installed data sets. In spite of our request not every supplier had installed the data set of the presentation.

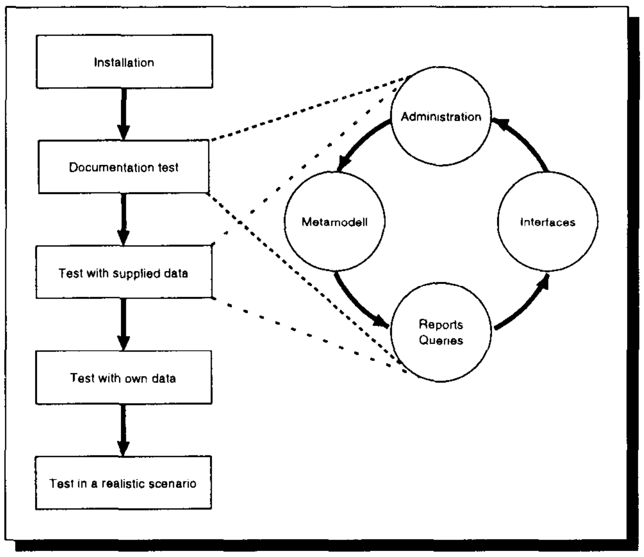

Figure 3. The testing spiral

After this phase we were able to exclude two more candidates. One was not able to deliver the needed interfaces and meta model modification facilities and one supplier did not provide any high-level access to the repository functionality. Up to this step the main work was spent in preparing the test plan and supporting the installation processes. The tests themselves had not been so time-consuming. The next turn in the spiral looked at the candidates with data that was prepared by ourselves. We tried to import our own authorization structures and meta models and finally ported an existing repository application. It was a repository benchmark we implemented for another project [16], but we did not want to use the results of the benchmark. This approach was chosen to reuse as much code as possible. Even this would have been too time-consuming, if more than two products had survived so far. Unsurprisingly we found some more weak points in the products while trying to use them our way. To accompany this phase we also visited an existing installation of one repository to verify our impressions on a working system.

When we started to prepare the next phase, the evaluation of a realistic scenario, it became clear that only one supplier was really able to support the interfaces that we needed, the other one failed to install the import filters and could give no indication of how expensive the creation of such filters would be.

From the beginning a hotline test was a part of our test plan. We even predefined error conditions and how we could enforce them. Sadly enough, but we did not need these tricks. With every repository we were forced to call the hotline, sometimes more than once. But we also learned a lot form these hotline contacts. They gave a good insight into the company and how able and willing they are to help their customers. One repository had such a bad user support that this alone would have been a reason to discard the product.

The test plan behind the above process was very detailed. It contained about 140 test items. The tests were numbered and the results were collected directly into the document. Figure 4 shows a (translated) extract from the test plan.

2.5. Integration tests

The last part of the test plan referred to the integration capabilities of the repositories in our test scenario. The test environment should be populated with real tools and, most importantly, with the change and configuration management tool that has been selected by that time. We could have stopped our efforts before that, but we continued in order to generate input for the tool introduction phase. As we expected, the integration tests were very time-consuming, because in most cases the integration actually has to be performed, since faults and traps can only be seen if substantial parts of the integration have been performed. The result of this phase was not a better selection but a paper that should help to perform the integration task for the FISCUS scenario.

3. Lessons learned about the process

Looking back on the process we found that the first impressions after the presentations had been right. If the supplier is forced to give a detailed presentation, including a live demonstration, a good basis for decision making is established.

It is obvious that test installations are needed, but the resources we needed to enable these test installations were huge. We had prepared a common hard- and software test environment and given a detailed description to the suppliers, but most of the installation teams did not know these descriptions. They had wrong installation media or versions for different operating system releases. We had to provide alternative hard- and software on-the-fly, trying to keep to the timetable.

This brings up another crucial point: In planning such an effort, coordination delays are a big problem. We planned and distributed deadlines early, but because it is hard to exclude a tool only based on a broken deadline, we had to cope with a lot of delays.

What we learned was that suppliers do not invest in "homework". They do presentations, but do not customize them much, even if they are asked to do so. Even the chance

4.2.7 Failure resistance

The integrity of the data must be preserved under all possible circumstances. It will be checked how...

Tests:

Test 71: Forced abort of a client
Test 72: Forced abort of the server
Test 73: Reboot of the server hardware
Test 74: Complete power failure
...

Test procedure: Provoke the condition of the test case while running at least one client program. Check the...
KO-criteria: The data cannot be recovered, even with the help of the supplier...

4.2.8 Hotline

A functional and responsive hotline support is essential for the environment...

Tests:

Test 77: Reachability
- when is the hotline reachable
- do they call back
- ...

Test 78: Procedure
Test 79: Provoked problem
...
Test 81: Compare with questionnaire (**F1**, **F29**)
...

Test procedure: ...fill up all available disk space and start some operation. Call the hotline with an error report.... Call the hotline on Friday 5 o'clock...
KO-criteria: The hotline is not reachable, trying at least three times a normal day...

4.2.9 Security

A secure data management is very important. In addition to the mandatory security features...

Tests:

Test 83: Is there a login procedure for the tool?
Test 84: What is the password policy?
Test 85: Can unauthorized persons:
- read data
- change data
- change access rights
- ...

Test 86: Is data encryption supported?
...

Test procedure: These tests will be performed with different operating system privileges...
KO-criteria: Access to data is possible...

Figure 4. Excerpts from the test plan

of a large order did not change this attitude. Our request for special data sets was ignored in all cases. Thus, in repeating such a task a test plan that includes the generation of test data is needed.

To find the weak points of a product, paper work is a first step, but only doing some substantial work will reveal information about problems. For this task, a set of implementations which can be ported is needed.

Going through the spiral and learning more and more details about the remaining tools, we had to adapt our requirements, because we had to omit hard criteria that were not met by any tool.

The questionnaire has already showed that a list of reference installations is a valuable selection criterion. If it is possible to visit an existing installation it can give inside information not otherwise obtainable. But this visit should be delayed as much as possible because in most cases the questions asked determine the quality of the information.

4. What we learned about the repositories

The Fraunhofer ISST has a research background in database support for engineering environments and so we were curious to learn what today's commercial repositories can supply. The sad truth is that no one is capable of providing all the features that have been known to the research community for many years. We missed good security concepts, there was no support for nested transactions or design transactions. Versioning was not always handled in a consistent manner or not handled at all. There is a big difference between the repositories that started as object management systems (OMS) for a tool or a tool family and the ones that were designed as data dictionaries, mainly for the mainframe world. They both tend to favor the functions they were initially designed for and do not cover both aspects well.

For industrially available repositories consultancy seems to be the main source of profit.

Security is a big problem for repository suppliers and also the use of repositories over a WAN or Internet is not treated well enough.

There is no object-orientation, even if the support of tools for Object-Oriented Analysis or Design (OOA/OOD) methods is traded as such.

4.1. Interpretations of the term "Open"

All suppliers called their repository "open", but this provides not much information. In most cases practically everything is possible, but how it can be done and with what efforts and costs is left to the consumer. In some cases openness is achieved by providing means to directly access the underlying data structures and thus enforcing the violation of security and encapsulation. There is no real metric for openness, but we defined as our metric the effort that is needed to adapt an existing tool to the repository. This definition of openness should enforce the implementation of such an integration, but a lot of information can be derived by scanning the documentation. For instance, if the documentation presents no example for the integration of a tool, it can be expected that integration is not trivial.

4.2. Comparison with the PCTE standard

Due to our experience with PCTE [13, 20, 16, 15], we were tempted to compare the repositories with the PCTE standard [14, 19], and our experiences with PCTE implementations. None of the repositories supplied a PCTE interface or mapped their data model to the one described in the PCTE standards.

There are a lot of aspects which are addressed by the PCTE standard and which are not available in current repositories. The security system of PCTE, even that of PCTE 1.5 [17], is not reached by any repository, neither are the distribution and replication facilities. For instance, the automatic reconnect after a server or network failure was a big problem for all tested repositories. Because PCTE is a standard, its interface is very well documented, this is not the case with the repositories. Their interfaces tend to change from version to version and most of them hide the basic interface from the user. In PCTE the processes and the communication means are defined in the standard, most repositories handle these aspects only in the context of the hosting operation system.

The frontends for commercially available repositories are good, but sometimes they encapsulate a lot of functionality that should be on server side. There are a lot of existing import and export filters and other kinds of bridges available for most of the repositories, more than for PCTE. The text-based interfaces are the most common ones, but for PCTE there is only a textual representation for the DDL, not for the data itself. It might be a good thing to adapt CDIF [5] or something similar to PCTE.

5. Conclusions

We have defined a structured process for repository evaluation and given some information about our experience with that approach. It is our belief that this process can be adapted to different requirements easily and might help others and us to solve similar tasks with less costs in shorter time.

In this project we have also learned that state of the art and state of the market in repository technology are still far apart and that both could benefit from one another. The technology transfer problem is very evident in this area.

Acknowledgements

We would like to thank all the people who contributed to the evaluation process: From OFD Hannover: Thomas Jahn, Hr. Rey, Stefan Schröder, and Hr. Weihrauch. From OFD Magdeburg: Ralf Graff, Holger Keil, and Frank Obermeit. From Partner Consult: Hans Hoffmann. From Fraunhofer ISST: Gregor Badura, Lars Heete, Bernhard Holtkamp, Andreas Kiraly, Michael Lietmann, and Norbert Weißenberg.

It was fun working with them and they all contributed to the experiences we shared in this paper. Our thanks goes also to the people of the repository suppliers, for their presentations, their technical aid and all their other efforts.

We would also thank the anonymous referees, Michael Kuchenbecker and Nick Grindell for their help in improving this paper.

References

[1] A.-J. Berre and T. L. Anderson. The hypermodel benchmark for evaluating object-oriented databases. In R. Gupta and E. Horowit, editors, *Object-Oriented Databases with Applications to CASE, Networks, and VLSI CAD*. Prentice Hall, 1991.

[2] M. J. Carey, D. DeWitt, and J. Naughton. The oo7 benchmark. In *ACM SIGMOD International Conference on Management of Data*, volume 22 of *SIGMOD Record*, pages 12–21, Washington DC, USA, 1993. ACM Press.

[3] R. G. G. Cattell. An engineering database benchmark. In J. Gray, editor, *The Benchmark Handbook for Database and Transaction Processing Systems*. Morgan Kaufmann Publishing Inc., 1991.

[4] R. G. G. Cattell and J. Skeen. Object operations benchmark. *ACM Transactions on Database Systems*, 17(1):1–31, 1992.

[5] Case data interchange format interim standard. EIA/IS 106, Electronic Industries Association, c/o Patti Rusher, 2500 Wilson Blvd., Arlington, VA 22201, U.S.A, 1994.

[6] S. Dewal, W. Emmerich, and K. Lichtinghagen. A decision support method for the selection of omss. In P. A. Ng, C. V. Ramamoorthy, L. C. Seifert, and R. T. Yeh, editors, *Proceedings of the Second International Conference on Systems Integration*, pages 32–40, Morristown, New Jersey, USA, June 15–18 1992. IEEE Computer Society Press.

[7] S. Dewal, H. Hormann, L. Schöpe, U. Kelter, D. Platz, and M. Roschewski. Bewertung von Objektmanagementsystemen für Software-Entwicklungsumgebungen. In *Proceedings of the „Datenbanksysteme in Büro, Technik und Wissenschaft“ (BTW) conference*, number 270 in Informatik Fachberichte, pages 404–411, Germany, 1991. Springer Verlag.

[8] S. Dißmann, W. Emmerich, B. Holtkamp, K. Lichtinghagen, and L. Schöpe. OMSs comparative study. ESPRIT II - ATMOSPHERE Report D2.4.3-rep-1.0-UDO-EL, Lehrstuhl Software-Technologie, Fachbereich Informatik, Universität Dortmund, Dortmund, 1991.

[9] Reference model for frameworks of software engineering environments. Technical Report ECMA TR/55 and NIST Special Publication 500-201, ECMA/NIST, Dec. 1990.

[10] M. A. Ellis and B. Stroustrup. *The Annotated C++ Reference Manual*. Addison-Wesley Publishing Company, Reading, MA, 1990.

[11] A. Goldberg. *Smalltalk-80: The Interactive Programming Environment (The 'Orange Book')*. Addison-Wesley Publishing Company, 1984.

[12] A. Goldberg and D. Robson. *Smalltalk-80: The Language and its Implementation (The 'Blue Book')*. Addison-Wesley Publishing Company, 1983.

[13] W. Goldschmidt, editor. *Proceedings of the Workshop on the Intersection between Databases and Software Engineering*, Sorrento, Italy, 16.–17. May 1994. IEEE Computer Society Press.

[14] ISO/IEC 13719-1: Information technology – Portable Common Tool Environment(PCTE) – Part 1: Abstract Specification. International Standard, ISO, 1995.

[15] J. Neuhaus and X. Wu. A smooth migration path towards a fully object-oriented PCTE. In I. Campbell, editor, *Proceedings of the PCTE'93 Conference*, number 1 in PCTE Technical Journal, pages 352–376, Paris, France, Nov. 17–18 1993. Syntagma Systems Literature.

[16] J. Neuhaus and X. Wu. Implementing a general OMS benchmark on PCTE and ECMA PCTE. In T. Lindquist and H. Koehnemann, editors, *Proceedings of the PCTE'94 Conference*, number 2 in PCTE Technical Journal, pages 5–20, San Francisco, CA, U.S.A., Nov. 29 – Dec. 1 1994. Mark V Systems Limited, CA, U.S.A.

[17] PCTE Interface Control Group. PCTE – a basis for a portable common tool environment, Functional Specifications. Technical report, Commission of European Communities, 1988. Version 1.5.

[18] H. Schreiber. JUSTITIA: A generic benchmark for the OODBMS selection. In *Proceedings of the Fourth International Conference on Data and Knowledge Systems in Manufacturing and Engineering*, Hong Kong, 1994.

[19] L. Wakeman and J. Jowett. *PCTE: The standard for open repositories*. Prentice Hall, 1993.

[20] X. Wu and J. Neuhaus. Extending PCTE with object-oriented capabilities. In V. Mařík, J. Lažanský, and R. R. Wagner, editors, *Proceedings of the 4^{th} International Conference on Database and Expert Systems Applications*, number 720 in Lecture Notes in Computer Science, pages 681–684, Prague, Czech Republic, 6–8 Sept. 1993. Springer Verlag.