A Decision Support Method for the Selection of Object Management Systems*

Sanjay Dewal, Wolfgang Emmerich, Karsten Lichtinghagen
University of Dortmund
Department of Computer Science
P.O.Box 500 500
4600 Dortmund 50
FRG

Abstract

With the increasing demand for highly complex, integrated and application domain specific systems engineering environments (SEEs) more or less specialized components of the SEEs are developed. An important component is the database management system (DBMS). It is generally accepted that conventional DBMSs are not useful to fulfill the requirements on highly complex, persistent data structures. Rather specialized DBMSs, namely object management systems (OMS), have been developed for fulfilling the enhanced requirements. An advantage of OMSs is that they further enhance the integration not only of data but also of processes.

Currently several specialized OMSs with significantly different properties such as the data model, architecture and performance are available. Thus it is very difficult for an SEE developer to select the most appropriate OMS for his SEE. In this paper we have proposed a decision support method which enables an SEE developer to identify his requirements and to compare the evaluation results of different OMSs. Additionally we present a practical experiment where we have applied the decision support method for comparing different OMSs. Experiences of the investigation are presented briefly.

---

*This work has been partially funded by the Esprit II Project ATMOSPHERE (No. 2565)

1 Introduction

Nowadays more and more highly complex, integrated and application domain specific systems engineering environments (SEEs) are developed. For such SEEs the development of more or less specialized components of the SEEs becomes necessary. These components are tailored towards solving problems of the particular application domain considered.

The demand for complex SEEs includes the necessity for managing persistent, complex data structures. It is generally accepted that such complex data structures cannot be managed by conventional database management systems (DBMSs) such as the well-known relational DBMSs ([GL85], [Mai89]). Non-standard database management systems, namely **object management systems (OMSs)** have been developed for managing the complex data structures.

For developing highly integrated SEEs it is necessary to determine the various interfaces of the components of an SEE to be integrated and to define particular integration mechanisms for achieving the integration. For example the ECMA reference model for SEEs [Ear90] identifies three interfaces of components, namely user, task and data management interface. OMSs are the one of the major vehicles to achieve data integration by enabling component interoperation via the programming interfaces of the OMSs (see fig. 1). Additionally particular trigger mechanisms enable task integration with OMSs to some extent.

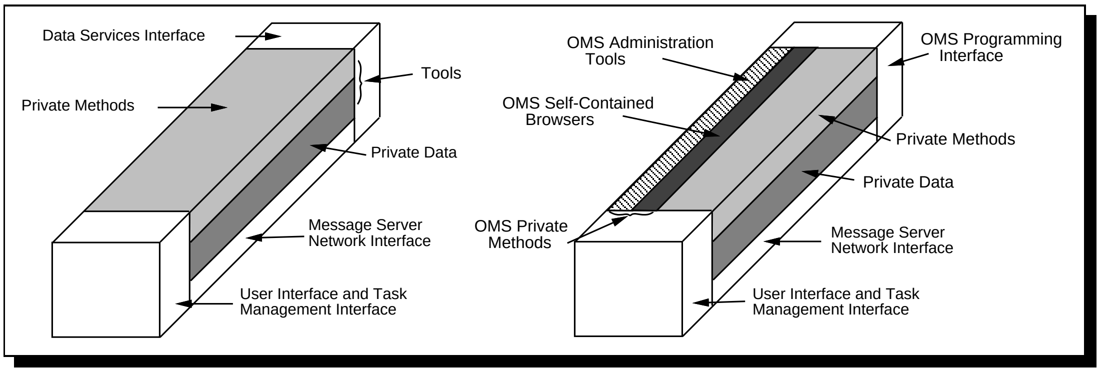

Figure 1: SEE-Reference Architecture / Data Integration by OMSs

The heterogeneity of the requirements on (1) the persistent complex data structures of the different application domains and (2) the degree of integration has resulted in the development of many more or less specialized OMSs. The problem for an SEE developer[1] is to select the most appropriate OMS among the existing ones, because the OMSs differ significantly regarding the data model, architecture, host language, etc.

This paper focusses on defining a method to select an OMS for an SEE developer. In this context selection means identification of the requirements on an OMS, the evaluation of OMSs regarding these requirements and the comparison of the evaluation results of the different OMSs in order to derive a purchasing decision. In section 2 of the paper we present the particular goals of the decision support method in detail. In sections 3 and 4 the decision support method and the decision process for OMSs are presented. In section 5 the results of an evaluation of OMSs performed in the project ATMOSPHERE[2] by applying the decision support method analyzing various OMSs are presented.

2 Goals of OMS Evaluation

In the project ATMOSPHERE existing OMSs are used for the development of SEEs and components of SEEs, respectively. Our method therefore focusses on the selection of an OMS for an SEE developer.

The scope of a selection of OMSs significantly differs from the scope of such a selection of conventional DBMSs. In the latter case the scope is on comparing the implementations of the same interface, optimization and tuning of the DBMS or performance estimation for predefined loads. For the selection of OMSs we propose the following scope:

- the description of the *technical properties of the product* (architecture, performance, functionality etc.)

[1] i.e. a person or institution developing SEEs

[2] ATMOSPHERE is a project that focusses on the development of a complex integrated industrial SEE

- the description of the *non-technical (commercial) properties of the product* (supplier, price, documentation, maintenance, support etc.)
- the *assessment of the product*, including an assessment of the adequacy and usefulness of the technical concepts, the performance and usability of the product.

Selecting an OMS means to define requirements on the OMS. The requirements on an OMS are obviously dependent of a particular SEE to build. The aforementioned heterogeneity of the applications and the resulting heterogeneity of the derived requirements aggravates a general assessment of an OMS in such a way that either the decision support method has to be restricted on assessing very common features of the OMSs or has to use very precise requirements derived from a particular application. Nonetheless the decision process may be very difficult for several reasons.

- The differences of the various OMSs complicate the comparison of the evaluation results.
- Due to the differences in philosophy and documentation of the various OMSs it is even not trivial to identify common basic features.

We present a decision support method which overcomes the aforementioned difficulties. The method presented in this paper focusses on the definition of such an decision framework providing a uniform grid for the comparison and evaluation of OMSs with the intention to be appropriate for many different requirements and OMSs. In particular our method allows to

- cope with the heterogeneity regarding OMS properties by defining a common view on the different systems using a classification scheme,
- simplify an individual selection by allowing to check the application specific requirements.
- assess the performance of the different OMSs using benchmarks.

3 Decision Support Method for Selecting OMSs

An SEE developer applies the decision support method for identifying the most appropriate OMS for the currently developed system. A faulty decision may be disastrous for the SEE developer, as the development schedule can be delayed or the funding does not allow to buy another OMS. Thus the decision must be comprehensive. In particular it is not sufficient to focus on some aspects of an OMSs only, but to consider various aspects, such as data model, architecture, performance, etc.

For a comprehensive decision the SEE developer has to define his requirements. Such requirements describe the role which an OMS plays as part of an SEE. However, the various characteristics dependent on the role of the OMS are quite difficult to derive. The problem is that for identifying such characteristics an SEE or parts of an SEE must be completely developed. In particular it is important to define the requirements of an SEE user[3] and to define the architecture of the SEE on the basis of the SEE user specific requirements in order to derive the particular role which the OMS plays as a component of this SEE.

Such an approach is very complex and highly time- and effort-consuming. The variety of existing requirements catalogues (see [GPI89], [GIE88], etc.) and the variety of the existing OMSs [SH90] point out the difficulties defining SEE user specific requirements catalogues. For solving the problem we have defined a **classification scheme**. The classification criteria identify and address the various aspects of OMSs which are summarized in the following.

- **General (non-technical) product information:** name of the OMS, supplier, hardware platform, price, documentation, etc.;
- **Architecture:** in particular the functions, internal interfaces, distribution concepts, extensibility and openness;
- **Functionality of the programming interface:** the data model, identification/navigation, versioning, external views, etc.;
- **System Features:** segmentation and clustering, transaction concepts and mechanisms, access control, security, etc.;

[3]SEE user is a person or institution developing systems using an SEE

- **Standard browser:** required hardware/software, functionality, data model in comparison with the programming interface, etc.;
- **OMS administration:** Installation, crash recovery, administration of distribution, performance monitoring and tuning, etc.;
- **Performance issues:** execution time of the OMS functions, execution conditions, etc.

The evaluation of OMSs regarding a classification criterion is highly dependent on the particular aspect addressed. Whether a particular concept is supported by the data model can usually be decided on the basis of the documentation of the SEE. However, aspects such as performance can be decided after practical use of an OMS only. Therefore we distinguish **analytical evaluation** and **experimental evaluation** which are described in the following sections.

3.1 Analytical Evaluation

During the analytical evaluation OMSs are evaluated on the basis of the documentation or any marketing material available. For most of the classification criteria it is very difficult to define formal evaluation scales for the evaluation results. Thus we have described the evaluation results of an OMS regarding each classification criteria informally.

With the analytical evaluation OMSs are evaluated regarding the classification criteria which consider aspects such as non-technical aspects, data model, architecture, application interface, etc.

3.2 Experimental Evaluation

Experimental evaluation is typically applied for evaluating OMSs regarding performance aspects. During the experimental evaluation a particular experiment further called **benchmark** is implemented and executed on the OMS. Benchmarks known for conventional DBMSs are usually defined in form of a source program (e.g. implemented in C or SQL). As the OMSs differ regarding their data model, interfaces and host languages, it is not possible to define a benchmark in form of a source program. Rather it is important to define the benchmark in a highly abstract way, such that it can be implemented on top of a large variety of OMSs. Here the term "abstract" means that the particular data storage problem is described on a highly abstract level, e.g. based on an entity relationship model, in order to be implementable on various OMSs. For distinguishing the benchmarks for conventional database systems and OMSs, we call the latter benchmark **abstract benchmark**.

For deriving any decisions on the basis of the abstract benchmark it is important that the abstract benchmark is SEE user specific. As an SEE usually works on complex data structures and uses many OMS functions which access the data, the OMS benchmark must ideally simulate the behaviour of an SEE or parts of an SEE (i.e. define may complex operations for complex data structures). An example for such a **complex benchmark** is the HyperModel benchmark (see [BAM88, ABM$^+$90]).

However, a complex benchmark is always only appropriate for a certain class of applications and its implementation is fairly expensive. Moreover, the implementation of a complex benchmark is complicated in so far that the choice of a particular data structure is usually not reproducible. Rather there exist several alternatives for the implementation of a complex benchmark which produce significantly different performance results. Thus the implementation must be optimized in order to use the appropriate OMS functions with good performance only. The paradoxon is that we must know the performance results of the OMS functions for evaluating the performance of the OMS functions.

The solution of this problem is a **two-step approach**. In the first step we define an OMS benchmark for elementary OMS functions using simple data structures. The main idea is that these operations should be easy to implement and valid for all OMSs constituting the basis for more complex and application dependent operations. The performance results of this benchmark, further called **simple benchmark**, may be afterwards used for implementing a complex benchmark. Such a simple benchmark has been defined in [DHS$^+$91]. Figure 2 depicts the data structure of the simple benchmark using a very simple entity relationship model. Figure 3 sketches the operations defined for the simple benchmark. The data structure of the simple benchmark is described in the following.

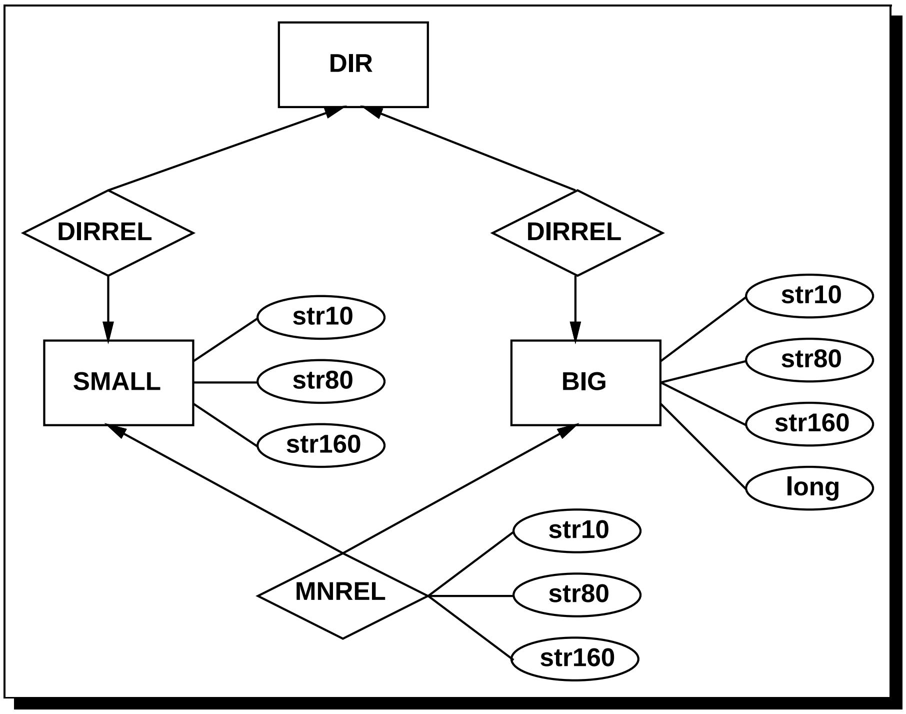

Figure 2: Data Structure of the Simple Benchmark

- Initialization, Opening and Closing of the database.
- Creation and Deletion of objects without initialization of the attribute values.
- Write and Read of "small" attributes of an object.
- Write and Read of "big" attributes of an object.
- Write and Read of all "small" attributes of an object.
- Write and Read of a relation.
- Write and Read of a "small" attribute value of a relation.

Figure 3: Operations of the Simple Benchmark

The data structure of the simple benchmark consists of three different object types, namely *DIR, SMALL, BIG*. For each of the two object types *SMALL* and *BIG* we define three attributes of the type *STRING*. With the attributes it is possible to store values (i.e. byte strings) with a maximum length of 10 Byte, 80 Byte and 160 Byte ("small attribute values"). For the object type *BIG* we define an additional attribute of type *LONGFIELD* where values up to 10 KByte and 128 KByte can be stored. The object type *DIR* has no attributes.

For each pair of object types *DIR/BIG, DIR/SMALL* and *SMALL/BIG* we define relation types. The first two relation types are $1 : n$ $(n \geq 1)$ relations without attributes, for the third relation type that is of the kind $n : m$ $(n, m \geq 1)$ three attribute of the type *STRING* are defined which are used for storing small attribute values.

This entity relationship model is very simple but it includes the basic object types and relationships that are typical within SEEs. Furthermore it can be implemented using almost every OMS. On this basis it enables a measuring of the elementary operations mentioned above that constitute the basis for all kinds of complex operations in SEEs. Before starting the measuring an initial database has been created for all tested OMSs that is based on the described entity relationship model.

4 Decision Process

The SEE developer identifies the most appropriate OMS by applying the decision support method. However, we claim that the identification process can be performed differently to the decision process. The idea is to classify all existing OMSs by using the classification scheme and not to classify an OMS for a particular SEE developer only. The advantage of such an approach is that it is possible to store the classification scheme and the results of all OMSs in a database and to provide tool support for the identification. Such a tool is called **decision support system** (**DSS**).

An SEE developer (1) views the existing classification criteria of the classification scheme, (2) defines the criteria of interest and (3) indentifies the importance of the selected criteria using the DSS. The DSS selects the evaluation results of the

OMSs stored in the database regarding the identified classification criteria. As the evaluation results are defined informally, the SEE developer has to view and compare the evaluation results thoroughly in order to identify the most appropriate SEE.

In case classification criteria are missing, the SEE developer has to add these criteria to the classification scheme. It is obvious that the OMSs must be evaluated regarding the newly defined criteria using analytical or experimental evaluation. The addition of new criteria to the classification scheme allows to improve the "quality" of classification scheme during repeated usage of the DSS.

Nonetheless the advantage of the DSS is that it is possible to derive decisions not only once, but several times. Furthermore the effort is reduced significantly as it is not necessary to perform the benchmarks repeatedly.

5 A Practical Example

We have applied the decision support method several times in different context (see [DHS+91], [DELS91]). In this paper we present the evaluation results produced in the context of the ATMOSPHERE project ([DELS91]), where we examined four OMSs, namely PCTE/OMS ([GMT87]), CADLAB/OMS ([MGSW89]), GemStone ([BMO+89]) and Damokles ([DGL86]). PCTE/OMS and CADLAB/OMS are both used within the ATMOSPHERE project, while the examination of GemStone and Damokles serves for comparison purposes. GemStone is an example for a wide-spread used commercial US-system, while Damokles is rather a research-oriented system than a product. However, Damokles has very good and interesting concepts.

As the existing classification schemata were not useful for the ATMOSPHERE context and the OMSs to be evaluated within were different, we had to perform the decision process "from scratch" (i.e. to define a new classification scheme and to evaluate the OMSs of interest). Thus in the first step we have modified the classification scheme defined in [DHS+91] according to the particular needs of ATMOSPHERE. In the second step we have evaluated the OMSs regarding the classification scheme. In the third step we have implemented and executed the simple benchmark on the different OMSs. For the performance analysis we have choosen several test frames that cover the most important execution conditions and enable to check the OMSs according to these conditions. Examples for these test frames are executions

- with direct and navigational access,
- with and without transaction mechanism,
- in a "cold" database state (i.e. read and manipulation of objects directly after opening the database) and in a "warm" database state (i.e. read and manipulation of an object immediately after reading the object),
- with different contents and sizes of the database.

Each benchmark has been performed using the same hardware platform (i.e. computer, primary and secondary storage) and software platforms (i.e. operating system, load). A constant load was achieved by guaranteeing exclusive access to the computing system during the execution of the benchmark. Furthermore we have repeated each operation and each benchmark several times in order to overcome unexpected side-effects on the performance results.

The evaluation has shown that the evaluated OMSs differ tremendously regarding their data model, degrees of object orientation, transaction concepts and many other aspects. In the remainder of this section we will only give some examples of our evaluation results. Due to space limitations it is not possible to present all the evaluation results in detail. The complete evaluation report can be found in [DELS91].

Data Models. Concerning the data models of the programming interface the OMSs differ in the underlying abstract model. Thus PCTE/OMS and Damokles are based on an *extended entity relationship* model, CADLAB/OMS has a *graph-based* and GemStone a *set-theoretical* data model.

Object Orientation. Except GemStone that may be called *fully* or *behaviourally object-oriented*[4] (i.e. it enables the definition of class specific methods) all other OMSs are only *structurally object-oriented* (i.e. methods are always predefined).

[4] Concerning the different degrees of object-orientation c.f. [Dit86]

Transactions. We have identified great differences in the supported transaction concepts. Damokles e.g. supports mechanisms for *design transactions* and PCTE/OMS covers *short* and *nested transactions*. The release of CADLAB/OMS which we have evaluated, however, has only a simple locking mechanism but no real transaction support, not even for short and flat transactions.

Standard-Browsers. Capable browsing facilities for comfortable interactive access to the database are actually imperative for OMSs. It is desirable for an OMS to provide at least one *textual browser* with full functionality or even better a *graphical browser*. Damokles, however, provides no browser, while CADLAB/OMS only provides a very low level textual browser (similar to a debugger) with restricted functionality. PCTE/OMS and GemStone provide full browsing facilities.

Administration Tools and Documentation. Not all of the OMSs show sufficient administration capabilities. GemStone and PCTE/OMS, however, provide specific tools for administration support. Documentation of the administration is also a general frailty of many systems.

Archivation and Recovery. Recovery and data replication is another weakness of most OMSs. GemStone and PCTE/OMS, however, supply tools or respectively procedures for replication and backup in case of system or media crashes.

Performance Concerning our performance measurements we have again identified tremendous differences between the OMSs. The execution times of the benchmark for the different OMSs have elucidated that each of the OMSs has its strength concerning particular execution conditions and hence is obviously not appropriate in all application areas. A general result is that there is one group of OMSs (PCTE/OMS, Damokles) that is more appropriate to handle big objects, i.e. flat files like textual or multimedia documents, while the other group of OMSs (GemStone, CADLAB/OMS) is more appropriate for small objects like e.g. syntax graphs or CAD objects.

Besides the mentioned results there are a lot of further interesting differences between the various systems, e.g. concerning versioning, security and distribution.

6 Summary and Conclusion

In this paper we have defined a method for selecting OMSs and we have presented evaluation results of OMSs which we have evaluated by applying the decision support method. The practical experiment with the decision support method has further pointed out the strength and weaknesses of the method.

In particular the experiment has shown that the classification scheme may be used for not only describing the various aspects of OMSs, but also for a comparison in order to identify the most appropriate OMS.

The major disadvantage of the classification scheme is that the definition of the classification criteria is not reproducible. In particular it is difficult for other persons to use the classification scheme as there does not exist a glossary where the different terms are defined.

For the experimental evaluation the simple benchmark allows to derive the performance of an OMS for simple OMS functions accessing simple data structures. We had lots of difficulties for ensuring the same hardware and software platforms and the same conditions such as work load in order to keep the results comparable. The major problems of the benchmark are that it focusses on the simple OMS functions only and that it does not cover an examination of concurrent execution.

In a next step, we are currently implementing a more complex benchmark for accessing more complex data structures. Furthermore we will extend the focus of the experimental evaluation on aspects such as distribution, multi-processing, etc. Additionally the introduction of a glossary for the classification criteria may allow different persons to work with the DSS.

Our experiences with OMS evaluation have shown that a single OMS can perhaps be sufficient for a certain application, but not for a number of different tasks within a complex environment like an SEE. For instance, a common data repository within a software development environment has to cope with fine grained objects (e.g. for the syntax graph of a syntax-directed editor) as well as with big objects (e.g. storage of the modular structure of a software system). Moreover a project manager could perhaps wish to store administrative data within a relational database system, especially if we consider that he uses a tool that internally handles relational data.

The simple example depicts the demand of complex applications for various highly heterogeneous requirements regarding different features of an OMS. This observation means that an OMS is quite useful to play a role in an SEE for integrating data and processes. However, for a highly complex SEE it is important to integrate several OMSs in order to support the different application specific requirements appropriately.

Our future activities in the context of OMS evaluation and analysis focus on the development and definition of a framework for integrating different OMSs in order to fulfill the complex application specific requirements. This integration framework is important, because the development of single overall optimal OMS fulfilling all requirements seems to be technically unfeasible for the near future. The coupling of different suboptimal OMSs is the straight-forward approach to combine the features of these OMSs.

The concept of such integration framework is based on the ESF-Software Bus in many aspects. So the framework should be open and extensible in the sense that an integration of OMSs has to be most easy by providing an appropriate plug-in concept. OMSs will be horizontally integrated below an OMS interoperation mechanism that defines different qualities of interoperation and due to this different degrees of integration. We will use our OMS evaluation method for an examination of different OMSs concerning their features and suitability for an integration into the framework. In this context the OMSs have to be analyzed concerning their interfaces, the different data models and the corresponding query languages with respect to OMS integration. The examination will be based on the descriptions of the OMS products.

References

[ABM+90] T. L. Anderson, A. J. Berre, M. Mallison, H. H. Porter, and B. Schneider. The hypermodel benchmark. In F. Bancilhon, C. Thanos, and D. Tsichritzis, editors, *Proceedings of the International Conference on Extending Database Technology – LNCS 416*, pages 317–331. Springer, Mar. 1990.

[BAM88] A. Berre, T. L. Anderson, and M. Mallison. VBase and the HyperModel Benchmark. Technical Report CS/E-88-031, Oregon Graduate Center, 1988.

[BMO+89] R. Bretl, D. Maier, A. Otis, J. Penney, B. Schuchardt, J. Stein, E. H. Williams, and M. Williams. The GemStone data management system. In W. Kim and F. H. Lochovsky, editors, *Object-Oriented Concepts, Databases and Applications*, pages 283–308. Addison-Wesley, 1989.

[DELS91] S. Dißmann, W. Emmerich, K. Lichtinghagen, and L. Schöpe. OMSs Comparative Study. Technical Report D2.4.3-rep-1.0-UDO-EL, ATMOSPHERE, 1991. To appear.

[DGL86] K. R. Dittrich, W. Gotthard, and P. C. Lockemann. Damokles – a database system for software engineering environments. In R. Conradi, T. M. Didriksen, and D. H. Wanvik, editors, *Proc. of an Int. Workshop on Advanced Programming Environments, LNCS 244*, pages 353–371. Springer, 1986.

[DHS+91] S. Dewal, H. Hormann, L. Schöpe, U. Kelter, D. Platz, and M. Roschewski. Bewertung von Objektmanagementsystemen für Software-Entwicklungsumgebungen (in German). In *Proc. of the GI Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft*, 1991.

[Dit86] K. R. Dittrich. Object-oriented database systems: the notion and the issues. In K. Dittrich and U. Dayal, editors, *Proc. of the 1986 Int. Workshop on Object-Oriented Database Systems*. IEEE Computer Society Press, 1986.

[Ear90] A. Earl. A Reference Model for Computer Assisted Software Engineering Environment Frameworks. Technical report, Hewlett Packard, August 1990.

[GIE88] GIE Emeraude. Requirements and Design Criteria for Tool Support Interface. Technical Report 15, ECMA TC33, 1988.

[GL85] W. Gotthard and P. C. Lockemann. Datenbanksysteme für Software--Produktionsumgebungen – Anforderungen und Konzepte (in German).

In W. E. Proebster, R. Remshardt, and H. A. Schmid, editors, *Methoden und Werkzeuge zur Entwicklung von Programmsystemen – Fachberichte und Referate Band 16*, pages 185–210. Oldenbourg, 1985.

[GMT87] F. Gallo, R. Minot, and I. Thomas. The object management system of PCTE as a software engineering database management system. *ACM SIGPLAN NOTICES*, 22(1):12–15, 1987.

[GPI89] GPI – The German PCTE Initiative. Requirements for the enhancement of PCTE/OMS, version 2.0. Technical report, Nixdorf Computer AG, Berliner Str. 95, D-8000 München 40, Mar. 1989.

[Mai89] D. Maier. Making database systems fast enough for CAD applications. In W. Kim and F. H. Lochovsky, editors, *Object-Oriented Concepts, Databases and Applications*, pages 573–582. Addison-Wesley, 1989.

[MGSW89] J. Miller, K. Gröning, G. Schulz, and C. White. The object-oriented integration methodology of the cadlab work station design environment. In *Proc. of the* 26^{th} *ACM/IEEE Design Automation Conference*, pages 807–810, June 1989.

[SH90] L. Schöpe and H. Hormann. Übersicht über Nicht-Standard Datenbanksysteme. Internal Report 42, University of Dortmund, Dep. of Computer Science, Software-Technology, FRG, Jan. 1990.