Journal of Intelligent Information Systems, 17:2/3, 301–322, 2001

Mediator-Based Evolutionary Design and Development of Image Meta-Analysis Environments

JESPER FREDRIKSSON jesperf@nada.kth.se
Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden; Division of Human Brain Research, Department of Neuroscience, Karolinska Institute, Stockholm, Sweden

PER SVENSSON pers@nada.kth.se
Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden; Department of Data and Information Fusion, Swedish Defence Research Agency, Stockholm, Sweden

TORE RISCH Tore.Risch@dis.uu.se
Department of Information Science, Uppsala University, Uppsala, Sweden

Abstract. We discuss how emerging object-relational database mediator technology can be used to integrate academic freeware and commercial-off-the-shelf software components to create a sequence of gradually more complex and powerful, always semantically and syntactically homogeneous, database centered Image Meta-Analysis Environments. We show how this may be done by definition and utilization of a use-case-based evolutionary design and development process. This process allows subsystems to be produced largely independently by several small specialist subprojects, turning the system integration work into a high-level domain modelling task.

Keywords: information system architecture, evolutionary development, human brain imaging, problem-solving environment, mediator technology, object-relational database

1. Introduction

This paper, developed from Fredriksson and Svensson (2001), discusses the evolutionary development, based on object-relational database mediator technology, of system architectures and implementations for a class of research information processing environments in which neuroscientific image processing, analysis, exchange and databased storage and retrieval are to be performed.

A process of co-operation between subprojects set up to develop a large, complex and evolving system is proposed, where mediator technology provides the glue between software components developed by the subprojects. An advantage compared to traditional component-based development is that most of the development and system integration work can be based on the use of an extensible declarative database language common to all subprojects during each development phase. The process is particularly suitable for interdisciplinary scientific co-operation engaging several domain experts, who need a common language to describe, develop and test new ideas in an evolving environment. This approach is being made possible by the emergence of new distributed and composable database mediator designs which include an object-oriented data model, reconciliation facilities, extensible

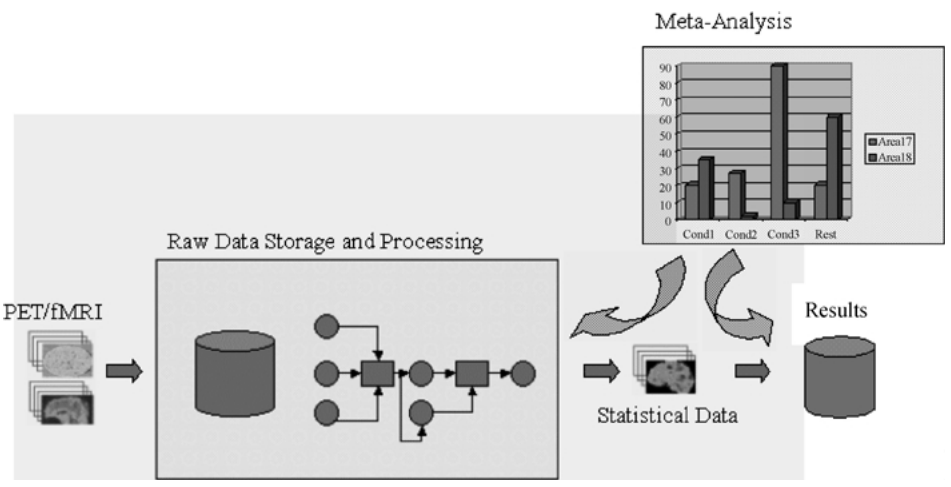

Figure 1. The BINS/NG facility as an instance of IMAE.

object-relational query language (ORQL) facilities, adaptive cost-based optimization techniques, and networked access to wrapped domain-specific and legacy database subsystems. We describe how this process is being applied to the creation of an instance of an *Image Meta-Analysis Environment* (IMAE), a database generator for multidimensional brain image data (Roland et al., 2001).

An IMAE is primarily responsible for management, processing, visualization, querying, and data mining on experiment data in several stages of refinement, see figure 1. The process of analyzing *inter-experiment*, even *inter-laboratory* data is referred to as *meta-analysis* and may include statistical tests, eigenimage analysis, data fusion of functional and structural data, and various other data mining and analysis techniques. The BINS/Neurogenerator (BINS/NG) system collects raw 3D brain image data in the form of PET and fMRI scans. A dynamic set of processing modules transform the raw image data into statistical images in a uniform manner. The resulting data homogeneity provides a stable ground for meta-analysis as well as a basis for testing and evaluating new image processing modules.

Potential advantages of this development approach are:

- basing the architecture on object-relational mediator technology allows fast prototyping of data modeling, workflow management, data mining, and image processing functions, the latter through reuse of existing image processing program modules in a database-centric environment, the others through exploitation of functionality already present in the query language of the mediator system or added as domain-specific extensions
- the flexibility of ORQL-based design solutions makes the resulting system easily amenable to stepwise modification
- complex system development tasks may be managed as a collection of decentralized specialist subprojects, without compromising the semantic coherence of the evolving system

- different kinds of data can be represented in different kinds of repositories and mediating facilities can be used for abstracting, combining, and reconciling mediated data
- our experience indicates that development projects based on conventional technology would require considerable more system designer and programmer resources; also, tighter and therefore more costly project management would be necessary

Thus, our experience indicates that systematic use of distributed mediator technology with ORQL extension facilities permits very fast prototyping of Image Meta-Analysis Environments. In fact, we believe there is opportunity to formulate a high-productivity evolutionary system design and development process based on this technology.

Our work contributes to the SSF[1]-supported BINS supercomputer-based neuroinformatics analysis center and the EC[2]-supported NeuroGenerator[3] database generation and distribution system. These projects develop new methodology to improve management and use of large Image Meta-Analysis Environments. Organizations participating in BINS are the Division for Human Brain Research of the Karolinska Institute, Stockholm (HBR), the Department of Numerical Analysis and Computer Science (NADA) of the Royal Institute of Technology, Stockholm (KTH), and the Parallel Data Processing Center (PDC), also at KTH. In NeuroGenerator, the Uppsala DataBase Laboratory (UDBL) of Uppsala University (UU), Forwiss and Active Knowledge (AK) in Munich, Germany, also participate. Per Roland (HBR) is project coordinator for both projects.

In these projects new methodology is being developed to improve management and use of large IMAEs while addressing the following research issues in scientific database management technology:

- development of evolutionary architecture and system development methodology for heterogeneous distributed scientific databases using mediator-based integration of 3D spatial image datatypes
- management of process workflow in very large repositories of raw image data, subjected to many statistical and image processing steps
- advancement of the state-of-the-art in mediator technology, in particular query processing in large distributed mediated heterogeneous databases
- development of data mining models and algorithms to support meta-research within large inhomogeneous sets of images from brain activation experiments.

This paper deals mainly with the first two of these topics.

The article is organized as follows. Section 2 provides a brief summary of concepts for evolutionary design of systems architectures. Section 3 discusses requirements and use-case models for Image Meta-Analysis Environments and provides a schematic use-case-based architectural analysis for the BINS and Neurogenerator IMAEs. In Section 4, two of the use cases are analyzed in more detail. Section 5 briefly surveys research on the design of Problem Solving Environments and related classes of systems. Section 6 concludes the paper.

2. Basic concepts of evolutionary software design and database mediator technology

In general terms evolutionary system design and development (Coplien, 1999) may be described as a methodology where large development projects are partitioned into an organized collection of *separately agreed* subprojects or phases. Each phase is developed according to a predefined *design contract*, which can and must be *operationally verified* by a set of "users" representing the "customer" organization. If during verification the design objective of any phase is found not to be satisfied, the customer will have to choose between allocating more resources to the phase (possibly after replacing the organization responsible for the unsatisfactory development work), reduce the design ambitions for the phase, or cancel the entire project. In order for the design contracts to be operationally verifiable by users, it should be expressed in terms of *use-case specifications* (Jacobson, 1994).

The rationale behind evolutionary design and development is to facilitate close customer and end-user involvement in the development process. By employing this approach the customer may avoid many of the risks involved in procuring the development of a large software system as a single item on the sole basis of possibly poorly understood paper specifications. Furthermore, it constitutes a process by which the customer can decide gradually which features need to be further developed as and when his experience with the system concepts grows.

Evolutionary design and development may be the most rational approach known for building large and unique software systems, but will not provide much leverage unless based on pervasive reuse of software tools and components. We claim that crucial software reuse opportunities may be offered by use of middleware, in particular by database mediator technology. Also, it is evident that as information system development migrates from a strictly controlled sequential in-house process towards a collection of concurrent activities performed by semi-autonomous cooperating groups, new methodologies need to be developed and employed which can initially establish and successively evolve an architectural "backbone" for the total process. In this paper, we present an outline of such a methodology.

2.1. Mediators

The purpose of the *mediator design pattern* (Gamma et al., 1995; Lévy et al., 1998) is to define a software component that encapsulates how a set of other components interact. Mediators promote a desirable weak coupling between components by keeping them from referring to each other directly, so that each component's interaction may be modified independently.

The specialization of the mediator concept to information and database systems, originally proposed by Wiederhold (Wiederhold, 1992; Wiederhold, 1995a), is perhaps the conceptually most complex mediator application yet attempted and has been studied in a number of research projects during the 90's (for a recent list, see Josifovski and Risch, 1999; Risch and Josifovski, 2001). In Wiederhold (1992), a (database) mediator is defined as "*a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications*". Such a software module should be small and simple so that it can be maintained by one expert or at most a coherent group of experts. Furthermore, mediators should provide data about themselves in response to requests by the potential users and, in distributed mediated systems, by other mediators.

2.2. Extensible ORDBMS and AMOS II database mediator technology

Object-relational database management systems (ORDBMSs) (Stonebraker and Moore, 1995; Carey and deWitt, 1996) allow extensions to be made on several levels:

- **Explorative level:** some model extension aspects may be explored and resolved entirely on the query language level (Orsborn and Risch, 1996)
- **Functional level:** by addition of specialized ADTs for specific domains (e. g. image data, spatial data, matrix data) the core database model may be extended to new application domains, such as *Finite Element Analysis* (Orsborn and Risch, 1996) and *Spatial Analysis* (Oukbir, 2001)
- **Performance level:** when the domain model has become linguistically and functionally well integrated, efficient access paths in the form of domain specific indices and optimization rules may be introduced to obtain scalable query evaluation performance (Stonebraker and Moore, 1995).

As realized in the extensible object-relational mediator system *AMOS II* (Active Mediator Object System II) (Risch and Josifovski, 2001),[4] database mediator technology allows pre-existing stand-alone DBMS's and other data sources (in BINS DB2 and files, in NeuroGenerator also RasDaMan (Baumann et al., 1997b)) to be integrated in an evolutionary fashion. This technology exploits powerful encapsulation capabilities of AMOS II to achieve simplicity, homogeneity and continuity of user conceptualizations throughout a planned succession of executable and verifiable prototypes of the final systems.

AMOS II is a *distributed database mediator system* allowing several AMOS II mediator servers to communicate over the Internet (Josifovski et al., 1999; Risch and Josifovski, 2001). The core of each server is an extensible, open, light-weight, main-memory ORDBMS. The mediator servers appear as virtual database servers having data abstractions and an SQL-99-like object-relational query language *AMOSQL*. The system can be used as a single-user database or as a multi-user server to applications and to other distributed AMOS II systems.

The distributed database mediator concept allows stepwise development of a syntactically and semantically homogeneous system out of several originally unrelated heterogeneous components. Intelligent wrappers, called *translators* (Josifovski and Risch, 1999), hide the heterogeneity from the information system developer, and *a fortiori* from the end user, contributing to increased software development productivity. In particular, this technology permits early experimentation with domain-specific query language concepts, in this case used for raster data management integration. In Fredriksson et al. (1999), the lack of such a facility in mainstream DBMS architectures was found to be a potentially serious shortcoming when using existing raster database extensions in IMAEs.

3. Application requirements and use-case models for an Image Meta-Analysis Environment

In this section, we discuss the system design task of building an IMAE. In the development of IMAEs, user involvement is particularly important, since IMAE users may frequently be experts in methodologies relevant to the system design. Extensive reuse of subsystems, whether commercial off-the-shelf systems or academic freeware, is critical to successful construction of these systems. The commonly employed decentralized academic project structure based on several largely independent supervisor-student Ph.D. subprojects emphasizes the need for a development process where subsystems can be independently produced and successively integrated into new system versions.

We make a schematic domain characterization and then successively condense this knowledge into use-cases and eventually an evolving system architecture. A sequence of overlapping use-cases is modelled in increasing detail while associated, increasingly stringent requirements on expressibility and performance are formulated. By repeating this process until a sufficiently significant subset of customer requirements have been met, we have established the basis of an evolutionary development methodology, which is illustrated by means of a few examples in Section 4.

A brain image database system, based on the RasDaMan raster database manager and the O_2 object-oriented database management system, was developed for ECHBD (Fredriksson et al., 1999), a precursor project to BINS and Neurogenerator. In ECHBD, only statistical population images are stored and no image processing control can be carried out by the database user. Thus, all image processing is carried out off-line to produce population average images which are then loaded into the database. Experience from this project was exploited in the development of the BINS and Neurogenerator architectures, in particular with regard to type integration between general purpose DBMSs and special-purpose raster databases, cf. Fredriksson et al. (1999).

BINS and Neurogenerator both address data analysis over a database containing raster data for 3D brain images as well as associated meta-data describing properties of the images and how they are produced (experiment setups, etc.).

3.1. Characteristics of Brain Image Analysis in a database-centered environment

Here we focus on the analysis of 3D functional brain images from fMRI and PET experiments. As an independent source of information, we consider structural 3D brain images of cytoarchitecture.[5] Images are associated with experiment descriptions called *image metadata*. More information on brain images and their metadata can be found in Fredriksson et al. (1999).

The reason for creation of a collaborative distributed system is that experiments are time-consuming to perform and frequently generate information of long-term value to the brain-imaging research community. Such information obviously needs to be shared between research groups. In fact, each experiment typically concerns only a small part of the total structure and function of the brain, leaving the synthesis of an overall model of the brain as a huge "meta-analysis" research task, based on results of numerous imaging experiments used collectively as basis for the characterization and mapping of generic classes of brain activity.

Other domain characteristics with architectural and developmental implications are listed below:

- **data sets contain specialized and complex data types.** 3D raster image data and metadata have very different structure. The analysis of these kinds of data needs to utilize their peculiar structural and semantic characteristics. Mining of brain image databases will require execution of complex queries over very large databases, which involve several kinds of data and metadata. In the NeuroGenerator project, mediator-based integration of specialized raster database technology (Baumann et al., 1997a) is expected to provide the performance necessary to allow such data mining, while overcoming the extensibility problem discussed in Fredriksson et al. (1999). Raster database storage and access technology will be used as a plug-in performance booster, while modelling and language characteristics of the raster data type will be integrated already on the mediator level, thus available to the BINS/NG systems independently of which raster data storage and access technology is adopted.
- **databases contain large data sets.** An fMRI experiment typically consists of scans from about 10 subjects during a single set of experimental conditions. Each subject is scanned approximately 1000 times, during each of which a 3D reconstruction of activations within the brain is acquired. Typically, each scan generates about 150^3 voxels. Thus, *in a single fMRI experiment*, approximately 3.4×10^{10} voxel values are measured and registered. This accounts for raw data generation only, and further processing of each image is necessary.
- **heterogeneous and complex information processing.** Many different levels of image refinement and intermediate results are created which need to be classified into *transient* and *persistent* objects. Analysis modules may themselves be very complex pieces of software, which are also subject to evolution. Thus, a homogeneous environment is needed to support independent comparison of analysis software.
- **compute-intensive information processing.** Some of the computational subprocesses involved require several hours *per image* on a typical Sun Ultra Sparc family type of computer. One single fMRI experiment is usually analyzed interactively using a large number of program modules during the run of a week for a researcher.

3.2. From domain characterization to use-cases

Since the system is expected to evolve in step with a still young field of research, system extensibility on several levels is a key requirement. This matches well with the concept of evolutionary design, as will be demonstrated below. A number of generic user requirements can be identified as a first step towards identifying the major use-cases listed and characterized in Section 3.3:

- well-integrated computational modules for sequential application to images, under control of an evolving set of workflow process networks
- querying functionality on image metadata for selection of indata to processing modules

- workflow descriptions and processing pedigrees will be subjected to archiving, retrieval, querying, reuse, and other management operations, as important subsets of the evolving system's metadata
- effective and efficient world-wide data distribution and, at least locally, shared access to images
- efficient content-based query functionality on large sets of 3D image data in combination with metadata
- *extensible* high-level query language to support querying and data mining on all types of data.

To satisfy the extensibility and evolutionarity requirements we have concluded that an extensible object-relational database mediator system provides an appropriate foundation for the architecture. The extensibility property allows both system developers and advanced users to add functionality to the system in a seamless and continuous manner, while mediation capabilities allow successive addition of new independent components as shown below. In keeping with the evolutionarity concept, we compose the above user requirements into a sequence of use-cases that lead directly to an evolutionary development process.

The overall system goals of the BINS/NG IMAE are to store, manage, and analyze raw brain imaging experimental data and workflows defined on these data. It should also be possible to query data from intermediate processing steps and to perform more computationally intensive data-mining tasks. The data are initially collected from various collaborating brain imaging laboratories and inserted semi-automatically with the help of a database administrator. The collaborating partners will be offered access to homogeneously processed data through an Internet accessible interface or from a local copy of the database.

We can thus identify the following actors in the system:

- the raw data provider
- the user; privileged or not; privileged users can execute processing chains
- a database administrator

Some candidates for use cases are:

- workflow creation/execution by local analyst
- raw data submission, involving the raw data provider and the database administrator
- raster data querying from a local or remote user

For illustration of our proposed method of object-relational, mediator based system design, we decide on four more refined use-cases (see Section 4) based on the latter two examples. They will be formulated from an evolutionary point of view, where each new use-case corresponds to increased demands on functionality or performance of the system.

3.3. Use-case-based evolutionary architectural analysis

By modelling typical use cases in steps satisfying increasingly higher demands on performance, scalability and system expressiveness, an evolution path is established for the system. The use cases can be found in figure 2 and the corresponding architectural solutions for each step are described in figure 3. The labels KI and PDC below denote the neuroscientific research laboratory at the Karolinska Institute and the supercomputing center at the Royal Institute of Technology, respectively, located a few kilometers apart and connected by a high-speed data link.

(a) Local execution of one processing module: The privileged user browses through databases and selects a process to execute and its indata

Analysis:

- define basic databases: EXPERIMENTDATABASE and PROGRAMDATABASE
- *Explorative Level* extensibility: Query language provides at least preliminary user interface
- *Functional Level* extensibility: Extend DBMS with functions for:
  - (i) invoking programs on indata images
  - (ii) notifying the user when program terminates
- schema development: Define base classes *Provider*, *Experiment*, *Subject*, *Condition*, *Repetition* and *Image* for data, and *PCModule* for programs.

(b) Whole processing chain execution: The privileged user designs a processing chain from raw data to some refined format, which is then executed and stored in the database.

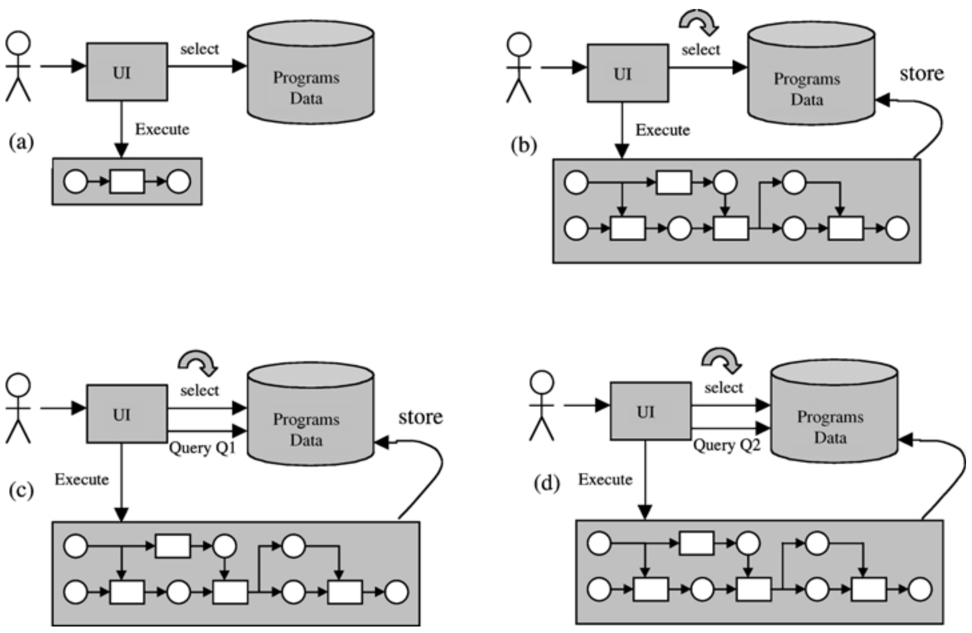

Figure 2. Four increasingly more challenging use cases.

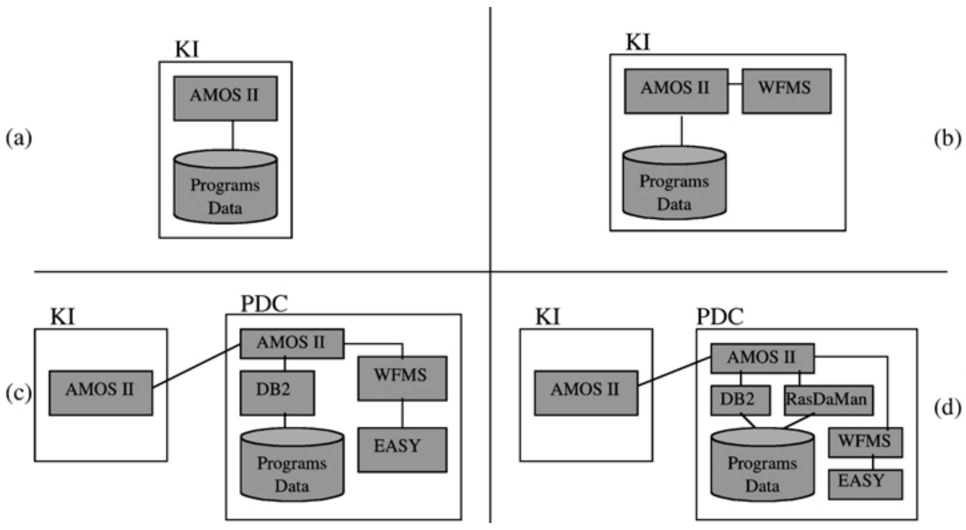

Figure 3. Architecture evolution of the BINS/Neurogenerator systems.

Analysis:

- *Functional Level* extensibility: complex processing creates requirement for workflow management, which is kept internal to the AMOS II DBMS by use of Event-Condition-Action (ECA) rules. A *computational proxy* (Cushing et al., 1994) is introduced as a new datatype with foreign functions to manipulate execution of its corresponding PCModule.
- a visual language is desirable to aid users while defining workflows.
- databases to be defined: WORKFLOWDATABASE for workflow data, TEMPORARYDATABASE for temporary image objects, HOMOGENEOUSDATABASE for end results
- schema development: Add classes *Analyst*, *Workflow*, *CompProxy* and *DataSource* for workflow management

(c) Same as (b), but processing at PDC node is monitored from KI node: Example query, issued by the user to the HOMOGENEOUSDATABASE: Q1 := “Find all images in which the subimage corresponding to the $[x_0 : x_1,\ y_0 : y_1,\ z_0 : z_1]$ interval has a voxel value greater than x”.

Analysis:

- the shared supercomputing environment with vastly increased computational power and storage capacity needs to be remotely monitored and controlled through the EASY scheduling system.
- apply the mediator design pattern (here: multidatabase nodes KI and PDC).
- *Functional Level* extensibility: define type extensions using foreign functions to perform simple raster data queries.

- schema development: the database schema needs to be mirrored in a relational DB2 database for persistent storage. An ODBC translator provides mediated access to this data. Sub-image relations is modelled by the function `subimage` between *VoxelSet* and *Image*.

(d) Same as (c) but with larger database and more demanding meta-analysis queries: Example query, issued by the user to the HOMOGENEOUSDATABASE: Query Q2 := "For each voxel find the number of statistically significant clusters that contains this voxel". More queries of this type can be found in Section 4.1.

Analysis:

- add raster data package with secondary memory storage, special storage structures and access paths for better scaling properties, and data compression for faster data transfer; either by adding this to the previously defined (in c) Performance Level extension or by providing mediated access to a separate specialized DBMS
- schema development: define subclass to *VoxelSet*, *ConnComp*, to be able to represent an arbitrarily shaped region of the brain

3.4. Architectural consequences

A set of architectures resulting from the analyses outlined above is summarized in figure 3. An issue suggested by these architectures is whether three different DBMSs are really needed, or in general, how can we be sure that the evolutionary design process leads to a sound architecture? Like in biology, evolutionary software design is usually expressed as modification of existing structures, while more thorough redesign is rare.

One might consider two alternative designs, using for database management

- either a single relational DBMS (DB2)
- or a specialized raster data DBMS (RasDaMan) and a relational DBMS (DB2) for meta-data

The first of these alternatives is feasible, but content-based retrieval (e.g. query Q1) on this large amount of image data will be very slow in comparison. The second alternative, however, does not provide a common query language thus putting the demanding task of data integration on the application programmer. This is illustrated in Section 3.3. That is, the best solution depends largely on the size and complexity of the databases. The BINS project stops at level (c) of figure 3, while NeuroGenerator builds on results from the previous project and promises more effective solutions on several levels (Roland et al., 2001). In order to realize these solutions, NeuroGenerator has been set up to provide the expertise necessary for managing the resulting design complexity. The label EASY used in figure 3 denotes the supercomputer job scheduler interface at PDC.

BINS/NG databases may be partitioned into the following categories:

- EXPERIMENTDATABASE contains a catalog of experiments, subjects, raw image data, and image metadata.

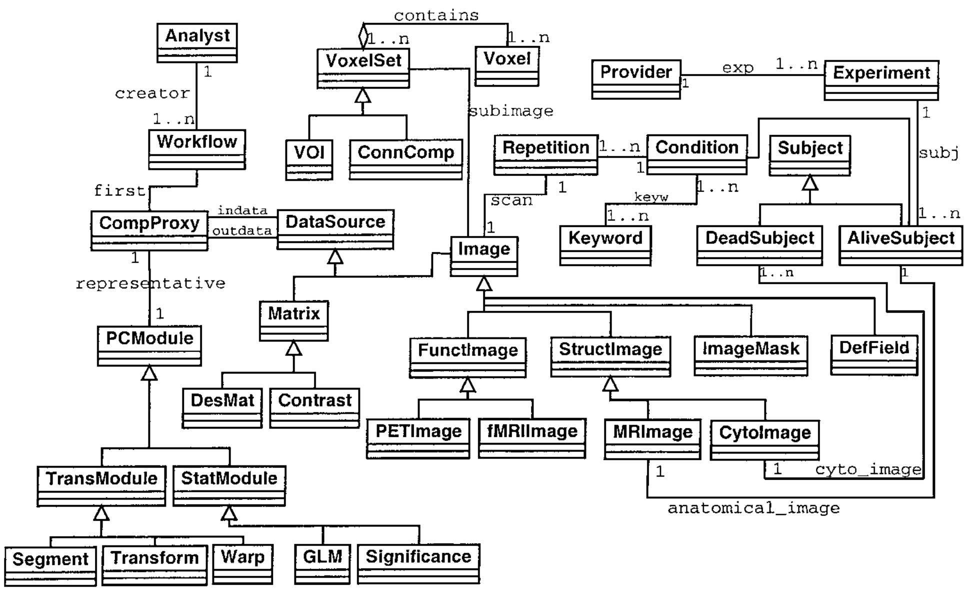

Figure 4. Database UML diagram.

- ProgramDatabase is a set of (mainly image) processing modules subject to software version control. Programs are stored at the PDC node in a DB2 database as BLOBs.
- WorkflowDatabase manages the workflow control data as shown schematically in figures 4 and 8. Intermediate data are stored in the file system. Loading of an image into the database is modelled as a workflow process.
- HomogeneousDatabase contains the processed database, consisting of processed images, their pedigrees and other metadata.
- TemporaryDatabase is used for view materialization of sub-workflows, optimization of subsequent analyses, and sample tests of process output quality. It is also in itself the target for meta-analysis.

The implications of the use-case analysis in Section 3.3 on the database schema is summarized in figure 4, which for clarity omits attributes on classes. The AMOS II data model is object-relational, thus allowing the UML model depicted in figure 4, but persistence of objects is achieved in DB2 and later on also in RasDaMan, so the actual long-term storage is in flat relational tables.

3.5. *Final architectural design*

Both BINS and NeuroGenerator are dual node systems, using the *PDC supercomputing center* as a backend storage and computing resource for image storage and processing. Raw data and modeling tools for these reside at PDC. A server at KI acts as the system's front-end, in control of its activities at PDC and also performing administrative tasks such as initiating the insertion of raw data into the database at PDC.

Privileged users can connect to the KI node to specify a new processing batch job at PDC, while underprivileged users only accesses the HOMOGENEOUSDATABASE, either from a local copy or over the Internet. Raw data submitters use a client/server based software to package the raw data, which is subsequently sent to PDC and inserted by running an interactive raw data loading program. The submitter uses this software to ensure that all the raw data files will be readable at KI, as well as to categorize the study and provide labels for the different files. The packaged files is either sent to a server at PDC, or saved to CDs/Tape. One *Communication Interface* provides access for the privileged users to create and initiate a new processing batch job, and another Communication Interface talks to PDC. *Process Feedback and Control* can be requested from the KI node and is sent to PDC via the Communication Interface through a high-bandwidth *Fibre Link*.

In BINS, references to files containing image raster data is stored as a user-defined external data type in *AMOS II* and manipulated with a set of user-defined procedural extensions to the ORQL. In NeuroGenerator, image raster data are stored in the specialized raster database system, RasDaMan (Baumann et al., 1997a, 1997b). RasDaMan manipulates data by means of a special array-oriented query language *RasQL*. An interface between AMOS II and RasDaMan will be developed in order to wrap the required RasQL functionality seamlessly and efficiently into its ORQL. Analogous interfaces already exist for SQL 92 and ODBC. The mediator will furthermore contain meta-data and views to mediate between data stored in DB2, RasDaMan, AMOS II, and files.

Figure 5 illustrates the mediator architecture of NeuroGenerator where distributed mediators define views of DB2 and RasDaMan data sources. A server mediator database running

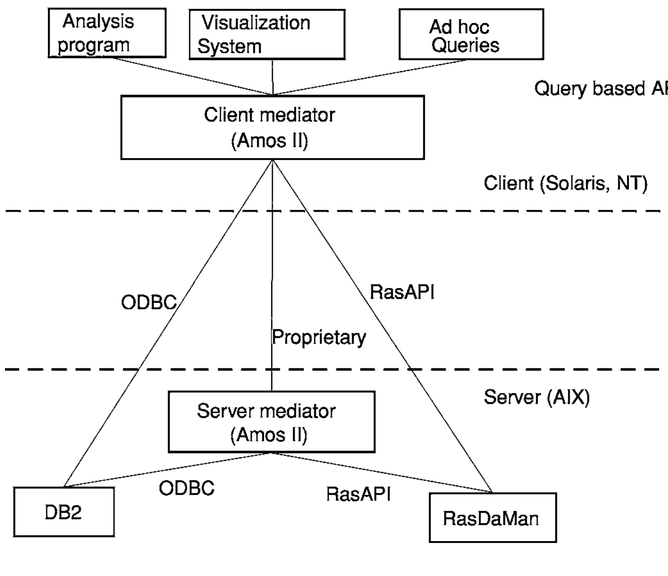

Figure 5. Distributed mediation of DB2 and RasDaMan databases.

at PDC provides views of data used by all applications and by the processing steps, while client mediators at, e.g., KI provide personalized views of integrated neuroscience data. Users and applications submit ORQL queries to a client mediator, which, in turn, translates the queries to optimized subqueries sent to the underlying DBMSs and the server mediator, using their client-server interfaces. The server mediator at PDC will include definitions of workflows for the processing steps expressed though active database rules, as explained below. The mediators have knowledge how to translate ORQL queries to a mediator into execution plans involving (sub-)queries to other mediators and to the data sources (Josifovski and Risch, 2001). Direct communication between the mediators and the DB2 DBMS is managed by the AMOS II ODBC translator while special translators are needed for querying RasDaMan. The AMOS II mediators internally communicate using a TCP/IP socket based protocol.

Various kinds of applications will run on the clients and they communicate with the client mediators through a query based API. In the figure this is exemplified with applications that analyze and visualize mediated data. Scientists can furthermore formulate general neuroscience database queries through an ad hoc query formulation system.

PDC furthermore stores all raw data and processing tools to be applied on the data as files in the *Hierarchical Storage Management* (HSM) system. These data are also transparently accessed via the server mediator engine residing in the PDC computer system.

4. Two illustrative use cases: Mediated access to distributed data and modelling of processing chains and workflows

The process described above based on evolutionary use-case models is one way of finding domains where existing knowledge can be exploited. Two examples of domain knowledge that were identified above are raster data and workflow management, to be discussed below.

In the first example, we indicate how high-level query language extensions are first implemented by foreign functions and later by mediated query transformation to a specialized raster data DBMS. To the user the two cases look the same, thus algorithms developed using the first version of raster data management need not be changed.

The workflow management subsystem was developed within AMOS II, using a simple active-rule-based design, cf. Ailamaki et al. (1998), Dayal et al. (1990). It could also have been implemented using a separate workflow engine wrapped by a mediator. Our implementation allows the state of a processing chain to be directly queried with all data stored in AMOS II. For an external workflow system, such functionality would require the workflow system to allow exportation of state data to the wrapping mediator.

4.1. Mediated access to distributed image data and metadata

A number of queries can be formulated to the databases described earlier. We choose to focus on a set of queries, described below, which has been decided to be of interest to the brain research community users. We then proceed to analyze demands on the IMAE in terms of modelling capabilities of the DBMS and make some initial observations on DBMS performance requirements from this set of queries. The resulting database schema can be found in figure 4.

Q1: To the HOMOGENEOUSDATABASE: "Find all images in which the subimage corresponding to the $[x_0 : x_1,\ y_0 : y_1,\ z_0 : z_1]$ interval has a voxel value greater than x and where all subjects were right-handed".
Q2: To the HOMOGENEOUSDATABASE: "Find the number of statistically significant clusters, for each voxel, that contains this voxel"
Q3: To the HOMOGENEOUSDATABASE: "Find all intersecting clusters from n images larger than V mm^3"
Q4: To a TEMPORARYDATABASE of, to a standard format, normalized functional images (before GLM processing module in figure 7): "Find all regions/voxels most correlated with region R/voxel v"
Q5: To the HOMOGENEOUSDATABASE: "Find the most frequent descriptive keyword associated with an activation in region R/voxel v"

All of these queries can more or less easily be phrased in AMOSQL in the generic form:

```
select function1(im, vs, v, x, k, s)
from Image im, VoxelSet vs, Voxel v, Keyword k, Subject s
where function2(im, vs, v, x, k, s)
```

Here, `function1` and `function2` are user-defined, possibly foreign functions to AMOSQL and x denotes a scalar value. The class *VoxelSet* is either a simple 3D bounding-box (Volume Of Interest, *VOI,* in figure 4) or an object representation of a connected region in space (*ConnComp* in figure 4). The *VoxelSet* class is now used to model the concepts subimage, cluster and region mentioned in Q1-Q5 and functions on this class will have separate realizations in each subclass. In the BINS system (figure 2(c)), images are stored in DB2 as BLOBs and accessed through the AMOS mediator. Q1 becomes ($\mathrm{I} := [\mathrm{x}_0 : \mathrm{x}_1,\ \mathrm{y}_0 : \mathrm{y}_1,\ \mathrm{z}_0 : \mathrm{z}_1]$):

```
select subimage (im, I)
from Image im
where voxel_activation(subimage(im, I), x) and right_handed
(subjects (im))
```

The query is phrased in AMOSQL, extended with the user-defined *foreign functions*:

- `voxel_activation (Image im, VoxelSet I, integer v) -> boolean`
- `subimage (Image im, VoxelSet I) -> Image`

performing activation checking and subimage extraction, respectively. The function `subjects (Image)` is a derived AMOSQL function that returns all subjects given a statistical image in the HOMOGENEOUSDATABASE by accessing the WORKFLOWDATABASE, described in Section 4.2, and finding the raw data images and the handedness of their subjects. The *Image* datatype is a *derived type* (Josifovski and Risch 1999), which mediates accesses to the actual DB2 BLOB database via an ODBC translator.

Similarly, in query Q2 we will need the ConnComp class and two foreign functions:

- `find_components(Image im) -> bag of ConnComp`
- `member_of(VoxelSet vs, Voxel v) -> boolean`

The first function computes all connected components for each image and the second determines whether a given voxel is a member of a *VoxelSet*. Queries Q3 and Q4 introduce intersection and correlational functions into the rapidly growing number of foreign functions needed.

However elegant as illustration of DBMS extensibility, the above solution is inefficient. Spatial indexes and disk access minimization techniques are useful features of a raster data manager, and the AMOS II system can be extended with this functionality. However, raster data management is a well-known problem domain with existing solutions that should be utilized. In the NeuroGenerator system, a RasDaMan subsystem manages all raster data. This subsystem is hidden from the application programmer, just as access to DB2 BLOBs is hidden in the BINS system. The AMOS II query processor analyzes each query, breaking it down into subqueries each of which can be efficiently answered by one of the data sources (i.e., DB2, RasDaMan, files, or AMOS II itself). The AMOSQL query Q1 can now be parsed and broken down into two subqueries, Q11 and Q12, where the former concerns handedness of the subjects and is sent to DB2. The latter is sent to RasDaMan (cf. figure 5) and looks like this:

```
    select im[x0:x1, y0:y1, z0:z1]
    from Image im
    where some_cell(im[x0:x1, y0:y1, z0:z1]) > x
```

In RasDaMan, a *tile-based* storage method (Baumann et al., 1997a) guarantees that only predefined subimages, *tiles*, which intersect the interval of interest are fetched. Moreover, RasDaMan uses compression of raster data, both when stored and when transferred from server to client. RasDaMan stores each image as several BLOBs in DB2, and serves in the architecture as a more efficient access path to raster data. A join of results from Q11 and Q12 can now be performed in the mediator. This last step is important, since without the mediator, this join would have to be performed by the application programmer (Fredriksson, 1999; Fredriksson et al., 1999). Cost-based optimization techniques and query transformations (re-writes) can be applied as a *performance level* DBMS extension to the AMOS II mediator. Typically, it will be much more costly to query raster data than metadata.

The need for faster query processing will grow with the size of the database. The simple approach to raster data storage in BINS allows testing of important system aspects at an early stage, making it comparatively easy to evolve from the BINS to the NeuroGenerator approach without changing application, control or communication code. In query Q2 we will find that e.g. an index on each voxel concerning membership of *VoxelSet*s will speed up query execution. This could imply storing, for each voxel, a list of pointers into statistically

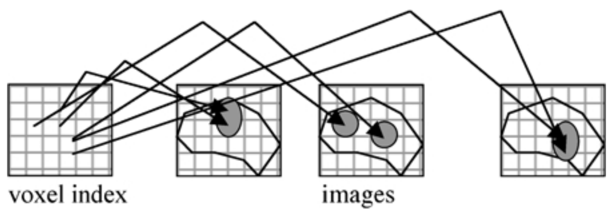

Figure 6. Voxel index of statistically significant clusters.

significant cluster in images (figure 6), or storing for each image every cluster in the object representation of *ConnComp*s.

In query Q3 this approach is no longer feasible, since we get a combinatorial increase in the number of intersections of *n* cluster images. Q3 and Q4 might be best handled in the workflow setting, described in the next section, and subject to batch processing on a supercomputer.

4.2. Modelling processing chains and workflows in a supercomputing environment

The following processing example deals with analysis of a PET experiment and serves here as a representative for an open-ended collection of processing chains that will be defined by future users of BINS and NeuroGenerator. In the example analysis we deal with a hypothetical experiment consisting of 10 subjects scanned during 4 conditions, repeated 8 times. This results in 10 anatomical MR images (one for each subject) and $10 \times 4 \times 8 = 320$ functional PET images and one standard brain (a "median" brain chosen from a population of brains).

The analysis is done by first mapping each anatomical MR image into the MR image of the standard brain, thus creating *deformation fields* describing each particular transformation. The deformation field transformation is subsequently applied to all PET images in the *warping* stage of the process. Before warping can occur, a *segmentation* of anatomical MR images has to be performed, to strip away the skull from the MR image, generating binary *mask images*. All transformed PET images, along with a logical description of the experiment in the form of a *design matrix X* and, in this case, three different *contrast vectors* $\boldsymbol{c}_{\mathrm{k}}$, $\mathrm{k} = 1, 2, 3$ (describing hypotheses about intensity-differences between functional images, obtained under different conditions) are fed into an implementation of the general linear model, *GLM*. The output of the GLM module are images of Student t distributed voxels, assessing the hypothesis associated with each contrast. *Statistically significant* clusters of voxels where the hypothesis was rejected is determined using the last module in the processing chain. These cluster images are inserted into the HOMOGENEOUSDATABASE.

This sequence of operations, one of the most popular workflows in this context, may be specified by use of the *Visual Workflow Editor* and results in the workflow structure shown in figure 7 (numbers in curly braces denote cardinality). Our design of the *Workflow Management System* (WFMS) is similar to those of Dayal et al. (1990) and Ailamaki et al. (1998) in the sense that the WFMS is internal to the DBMS and implemented by use of database schemas and active rules.

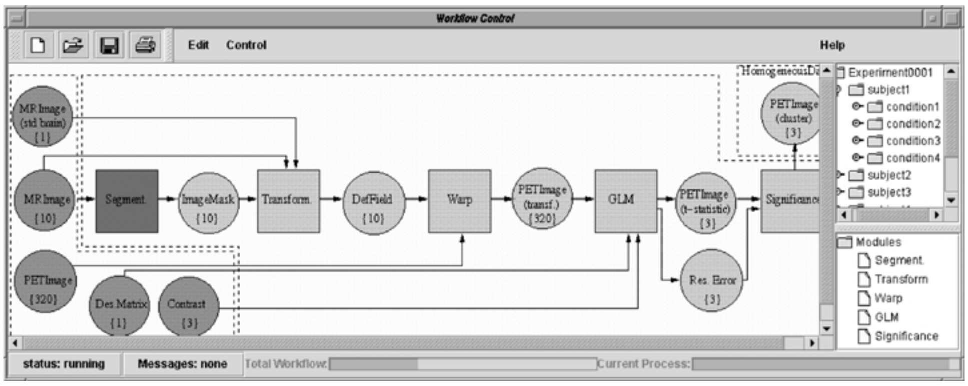

Figure 7. A user-defined workflow instance.

A user-defined workflow instance, such as the one in figure 7, is converted by the workflow editor to a schema which is stored and instantiated in the WORKFLOWDATABASE, figure 8. The functions `indata (CompProxy cp)` and `outdata (CompProxy cp)` represent the edges of the graph in figure 7. A computational proxy (Cushing et al., 1994), is a database representation of a computational processing step in a scientific workflow. When all indata to a `CompProxy` is present, the active rule described below fires and starts the corresponding process at PDC. Each of the three processes can be executed only when its corresponding indata is present, as captured by the following AMOSQL *Event-Condition-Action* (ECA) rule:

```
create rule execute_rule () as
from DataSource ds, CompProxy cp
on updated (iscreated (ds))
when ds = indata (cp) and ready (cp) and not (isexecuted
(cp)) do invoke(cp);
```

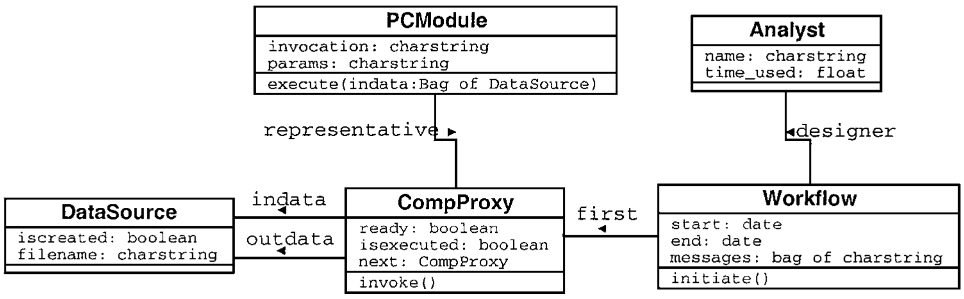

Figure 8. Detail from database schema corresponding to WORKFLOWDATABASE (refer to figure 4 for subclasses to PCModule and DataSource).

On the *event* of updating the Boolean property `iscreated` of a `DataSource`, the rule checks the *condition* that all indata is present (using the function `ready (CompProxy cp))` and that `CompProxy` has not already been executed. If these conditions are fulfilled, the *action* to invoke the process corresponding to this `CompProxy` is performed.

The function `invoke (CompProxy cp)` should now launch the actual process at the PDC node. In the AMOS II system, this can be performed in one of two ways:

- send a string of AMOSQL code through the TCP/IP connection to the AMOS II system at the PDC node that starts up the relevant process, or
- connect the KI AMOS II mediator to the PDC mediator. The KI mediator can now directly provide a database function that initiates the process at the PDC node.

Either way, an AMOSQL function is needed to initiate processing at PDC. This is a *foreign function*, phrased in Java, which talks to the EASY supercomputer scheduling system and negotiates a time-slot for this process. Later, when outdata from the process is created, the Boolean property `iscreated` of the corresponding outdata proxy at the KI node is set to true, again in one of the two ways described above. Figure 8 depicts the base-classes with attributes and functions needed for database representation of workflow. The larger schema in figure 4 shows all sub-classes to PCModule and DataSource present in figure 7.

5. Related work

Architectural concepts for scientific data management and analysis environments have been proposed and explored by a number of researchers since the 1970's, e.g. (Arnborg et al., 1979; Stonebraker, 1994; Ioannidis et al., 1996). A related research effort, usually directed towards development of less data-intensive and domain-specific computing environments than IMAEs, are the distributed *Problem Solving Environments* (*PSEs*) (Gallopoulos et al., 1994; Topcuoglu et al., 1997; Walker et al., 2000). Key aspects of a PSE are (Walker et al., 2000):

- *Intelligence*, to ensure the PSE is easy to use and computationally efficient,
- *Collaborative tools*, to allow effective collaborative work on complex problems; and
- *Visualization*, to support data analysis and navigation, as well as provide visual support for runtime monitoring and steering of applications.

Several other related concepts have been proposed, in particular *Experiment Management Environments* (Ioannidis et al., 1996), *Data-Centered Frameworks for Scientific Research* (Jones et al., 1999), and *Domain-Specific Environments for Computational Science* (Cuny et al., 1997). Ioannidis et al. (1996), Jones et al. (1999) are among the few PSE publications which emphasize applications that are simultaneously data-and compute-intensive, and which propose a data-centered PSE architecture.

We believe that our project is the first to apply mediator technology in order to transform the development of a complex data-centered research framework into an evolutionary process which can integrate powerful legacy software into a conceptually unified object-oriented database-centered system.

6. Discussion and conclusion

We have shown how design and development of a certain class of distributed, heterogeneous (different OS's, different processor architectures, different DBMS technology etc) scientific information systems can be both conceptually and operationally simplified by the use of distributed and composable database mediators. The old "time-sharing" model for supercomputer access can be abandoned and be replaced by a concept where the supercomputer becomes an invisible *backend processor* and *backend storage device*. The user does not need to deal with idiosyncratic concepts local to this particular kind of environment. Extensibility at the end-user level may be comparatively easily achieved by incremental introduction of additional interfaces to the database subsystems.

In software engineering, issues of software reusability and interoperability (Coulange, 1998; Szyperski, 1998) and interoperability between distributed objects (Orfali et al., 1996) have received considerable attention in past years. Research in mediator technology has mainly focused on integrating heterogeneous data sources (Wiederhold, 1992) but has also been envisioned as a tool for modularisation of software systems (Wiederhold, 1995b). We claim above that distributed mediator technology allows an evolutionary approach to developing complex information systems, such as Image Meta-Analysis Environments, based on re-use of existing DBMS and other software components. In comparison with conventional component-based development, a major advantage of our approach is the availability across heterogeneous subsystems of a common extensible object-relational query language (ORQL), allowing fast development of database schemas and basic processing functions of both prototypes and final subsystems. This language could become common to all project members, also in interdisciplinary projects where language commonality is likely to be particularly important. This property promotes evolutionary development by offering a possibility to evaluate proposed concepts at an early stage without requiring extensive user interface, or other software development.

In addition to providing an intelligent query-language-based search tool to users who need to find specific computational methods or experiment data sets, the architecture is making available as further intelligent features a visual active-rule-based workflow definition facility and processing pedigree databases which allow users to survey, recreate and modify arbitrary historical processing chains.

Using, as we propose, experimental software, or any piece of software still under development, as essential building blocks of a complex system raises the issue whether the resulting system will be stable enough to be acceptable to users. Necessary, if not sufficient, for this will be the early introduction and utilization of a stable logging, checkpointing and recovery subsystem, allowing system crashes to occur without causing irreversible destruction of valuable databases.

Acknowledgments

Members of the BINS and Neurogenerator project teams include: Per Roland KI/HBR, Gert Svensson KTH/PDC, Tony Lindeberg KTH/NADA/CVAP, Tore Risch UU/DIS/UDBL, Per Svensson KTH/NADA/TCS, Peter Baumann Active Knowledge, Andreas Dehmel Forwiss, Hjörleifur Halldorsson CVAP, Jesper Fredriksson TCS, Johan Sandström UDBL, Lars Forsberg PDC, and Calle Undeman HBR.

Many concepts discussed above originate from discussions with other project team members. Their contributions are gratefully acknowledged.

Notes

1. BINS (Brain Image Neuroinformatics System) is supported by the Swedish Foundation for Strategic Research, SSF.
2. NeuroGenerator (A Database Generator for the Neuroimaging Community) is supported by the European Commission under the LIFE programme (Quality of Life and management of Living Resources).
3. www.neurogenerator.org
4. More information about Amos II can be obtained at *http://www.dis.uu.se/∼udbl*
5. Parcellation of cortex with respect to cellular and laminar properties of neurons.

References

Ailamaki, A., Ioannidis, Y.E., and Livny, M. (1998). Scientific Workflow Management by Database Management. In *Proc. 10th Int. Conf. Scient. and Statist. Database Management*, Capri, Italy (pp. 190–199). IEEE Computer Society Press.

Arnborg, S., Elvers, E., and Svensson, P. (1979). Design Specification for Datalab, a System for Data Analysis based on the Relational Model of Data. Swedish Defence Research Agency (FOA) Report C 20326-D8, Stockholm, Sweden.

Baumann, P., Furtado, P., Ritsch, R., and Widmann, N. (1997a). The RasDaMan Approach to Multidimensional Database Management. In *Proc. ACM Symp. Appl. Comp.*, San Jose, California (pp. 166–173). ACM Press.

Baumann, P., Furtado, P., Ritsch, R., and Widmann, N. (1997b). Geo/Environmental and Medical Data Management in the RasDaMan System. In *Proc. 23rd Conf. Very Large Databases*, Athens, Greece (pp. 548–552). Morgan Kaufmann.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *Pattern-Oriented Software Architecture, A System of Patterns*. West Sussex, England: Wiley.

Carey, M.J. and deWitt, D.J. (1996). Of Objects and Databases. A Decade of Turmoil. In *Proc. 22nd Conf. Very Large Databases*, Bombay, India. Morgan Kaufmann.

Coplien, J.O. (1999). Reevaluating the Architectural Metaphor: Toward Piecemeal Growth. *IEEE Software*, 16(5), 40–44.

Coulange, B. (1998). *Software Reuse*. London, England: Springer.

Cuny, J.E., Dunn, R.A., Hackstadt, S.T., Harrop, C.W., Herey, H.H., Malony, A.D., and Toomey, D.R. (1997). Building Domain-Specific Environments for Computational Science: A Case Study in Seismic Tomography. *Int. J. Supercomputing Appl.*, 11(3), 179–196.

Cushing, J.B., Maier, D., Rao, M., Abel, D., Feller, D., and deVaney, D.M. (1994). Computational Proxies: Modeling Scientific Applications in Object Databases. In *Proc. 7th Int. Conf. Scientific and Statistical Database Management*, Charlottesville, Virginia (pp. 196–206). IEEE Computer Science Press.

Dayal, U., Hsu, M., Ladin, R. (1990). Organizing Long-Running Activities with Triggers and Transactions. In *Proc. SIGMOD Conference*, Atlantic City, New Jersey (pp. 204–214). ACM Press.

Fredriksson, J. (1999). Design of an Internet Accessible Visual Human Brain Database System. In *Proc. Int. Conf. Multimedia Computing and Systems*, Florence, Italy (vol. 1, pp. 469–474). IEEE Computer Science Press.

Fredriksson, J., Roland, P., and Svensson, P. (1999). Rationale and Design of the European Computerized Human Brain Database. In *Proc. 11th Int. Conf. Scientific and Statistical Database Management*, Cleveland, Ohio (pp. 148–157). IEEE Computer Society Press.

Fredriksson, J. and Svensson, P. (2001). Evolutionary Design and Development of Image Meta-Analysis Environments Based on Object-Relational Database Mediator Technology. In *Proc. 13th Int. Conf. Scientific and Statistical Database Management*, Fairfax, Virginia (pp. 190–199). IEEE Computer Society Press.

Gallopoulos, E., Houstis, E., and Rice, J.R. (1994). Computer as Thinker/Doer: Problem-Solving Environments for Computational Science. *IEEE Comput. Sci. and Eng.*, 1(2), 11–23.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns—Elements of Reusable Object-Oriented Software*. Reading, MA USA: Addison-Wesley.

Ioannidis, Y., Livny, M., Gupta, S., and Ponnekanti, N. (1996). ZOO: A Desktop Experiment Management Environment. In *Proc. 22nd International VLDB Conference*, Bombay, India (pp. 274–285). Morgan Kaufmann.

Jacobson, I. (1994). *Object-Oriented Software Engineering—A Use Case Driven Approach*. Addison-Wesley.

Jones, D.R., Gracio, D.K., Taylor, H., Keller, T.L., and Schuchardt, K.L. (1999). Extensible Computational Chemistry Environment (ECCE) Data-Centered Framework for Scientific Research. In M.E. Fayad and R.E. Johnson (Eds.), *Domain-Specific Application Frameworks: Frameworks Experience by Industry*, Ch. 24. Wiley.

Josifovski, V., Katchaounov, T., and Risch, T. (1999). Optimizing Queries in Distributed and Composable Mediators. In *Proc. 4th IFCIS Int. Conf. Cooperative Information Systems*, Edinburgh, Scotland (pp. 291–302). IEEE Computer Society Press.

Josifovski, V. and Risch, T. (1999). Integrating Heterogeneous Overlapping Databases Through Object-Oriented Transformations. In *Proc. 25rd Conf. Very Large Databases*, Edinburgh, Scotland (pp. 435–446). Morgan Kaufmann.

Josifovski, V. and Risch, T. (2001). Query Decomposition for a Distributed Object-Oriented Mediator System. *Distributed and Parallel Databases J.* (to be published).

Lévy, N., Losavio, F., and Matteo, A. (1998). Comparing Architectural Styles: Broker specializes Mediator. In *Proc. 3rd Int. Software Architecture Workshop*, Orlando, Florida (pp. 93–96). ACM Press.

Orfali, R., Harkey, D., and Edwards, J. (1996). *The Essential Distributed Objects Survival Guide*. Wiley.

Orsborn, K. and Risch, T. (1996). Next Generation of O-O Database Techniques in Finite Element Analysis. In B.H.V. Topping (Ed.), *Advances in Computational Structures Technology* (pp. 121–136). Civil-Comp Press.

Oukbir, K. (2001). A Database Query Language for Uncertain Spatial Data. Lic. Thesis TRITA-NA-0121, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden, 2001.

Risch, T. and Josifovski, V. (2001). Distributed Data Integration by Object-Oriented Mediator Servers. *Concurrency: Practice and Experience*, 13(11), 933–953.

Roland, P., Svensson, G., Lindeberg, T., Risch, T., Baumann, P., Dehmel, A., Fredriksson, J., Halldorsson, H., Young, J., and Zilles, K. (2001). A Database Generator for Human Brain Imaging. *Trends in Neurosciences*, 24(10), 562–564.

Stonebraker, M. (1994). SEQUOIA 2000—A Reflection on the First Three Years. In *Proc. 7th Int. Conf. Scientific and Statistical Database Management*, Charlottesville, Virginia (pp. 108–116). IEEE Computer Science Press.

Stonebraker, M. and Moore, D. (1995). *Object-Relational DBMSs: The Next Great Wave*. San Mateo, CA USA: Morgan Kaufmann.

Szyperski, C. (1998). *Component Software—Beyond Object-Oriented Programming*. Addison Wesley, New York, NY USA: ACM Press.

Topcuoglu, H., Hariri, S., Furmanski, W., Valente, J., Ra, I., Kim, D., Kim, Y., Bing, X., and Ye, B. (1997). The Software Architecture of a Virtual Distributed Computing Environment. In *Proc. 6th IEEE Int. Symp. High Performance Distributed Computing*, Portland, Oregon (pp. 40–49). IEEE Computer Society Press.

Walker, D.W., Rana, O.F., Li, M., Shields, M.S., and Huang, Y. (2000). The Software Architecture of a Distributed Problem-Solving Environment. *Concurrency: Practice and Experience*, 12(15), 1455–1480.

Wiederhold, G. (1992). Mediators in the Architectures of Future Information Systems. *IEEE Computer*, 25(3), 38–49.

Wiederhold, G. (1995a). Mediation in Information Systems. *ACM Computing Surveys*, 27(2), 265–267.

Wiederhold, G. (1995b). Modelling and System Maintenance. In *14th Int. Conf. Object-Oriented and Entity-Relationship Modelling*, Gold Coast, Australia (pp. 1–20). Lecture Notes in Computer Science No. 1021, Springer.