Systems Integration

Dagstuhl Seminar 99111
March 14–19, 1999

Paolo Ciancarini
Stefan Conrad
Wilhelm Hasselbring

Preface

The integration of systems which have been developed and evolved independently is one of today's major challenges in computer science. In a large spectrum of application areas the necessity of integrating (pre-) existing software systems is present and demands for applicable solutions.

Problems of coupling and integrating heterogeneous database and information systems are for instance investigated in the database area. Building multidatabase systems or federated database systems incorporating legacy systems is a big challenge. Current work covers topics like schema integration, transaction models for federated database systems, consistency enforcement in heterogeneous systems, security models, and query processing.

On the other hand, systems integration is an important challenge for the area of software engineering as well. Current work deals with questions of adequate software architectures and design patterns, coordination languages and models, composition of software components, development of workflow systems, the proper use of middleware tools such as CORBA, and methodological approaches for the integration process.

The rapid development of Web-related methodologies and tools also stimulates new problems, with respect to the access to Web data, the design and maintenance of Web sites, and their integration with traditional applications.

This Seminar was initiated due to the fact that there was only a rather loose coupling of the work done in these scientific communities, although the work of these areas is obviously highly interrelated. Therefore, we saw the chance that both areas can profit a lot from a mutual exchange of problems and ideas.

The Seminar brought together scientists from these two areas. A main focus of the Seminar was set on integration on system level. However, the influence of other integration levels (e.g., integration on schema or model level) was considered as well. Cross-disciplinary working groups were established during the Seminar aiming at a more detailed investigation of common problems.

Paolo Ciancarini
Stefan Conrad
Wilhelm Hasselbring

Contents

1 Program

Monday, March 15, 1999

9:00 Welcome and Introduction to the Seminar

Wilhelm Hasselbring:
On the Process of Systems Integration

10:30 Coffee

10:45 Günter Preuner:
Integration of Object Life-Cycles in Object-Oriented Design

Mark Roantree:
The OASIS Architecture: Legacy System Migration and Federated Schema Construction

12:15 Lunch

13:30 Martin Schönhoff:
Global Version Management in a Federated Design Environment

Alfred Winter:
Experiences with Current Methods of Integrating Hospital Information Systems

15:30 Coffee & Cake

19:00 Working Group Discussion:
Applications & Processes

Tuesday, March 16, 1999

9:00 Ralf Kramer:
Metadata: A Key Issue for Systems Integration

Henrik Loeser:
iWebDB — An Integrated Web Content Management System

10:30 Coffee

10:45 Ralf-Detlef Kutsche:
Integration Platforms for Distributed Information Systems

Michael Goedicke:
Integration through ViewPoints

12:15 Lunch

14:00 Lukas Faulstich:
Integration of Semi-Structured Information Sources with Hyper View

Wolfgang Emmerich & Anthony Finkelstein:
XML and Related Technologies: Overview and Research Directions (I)

15:30 Coffee & Cake

16:00 Wolfgang Emmerich & Anthony Finkelstein:
XML and Related Technologies: Overview and Research Directions (II)

Paolo Ciancarini:
Integrating XML with Java for Active Documents

Karl Aberer:
XML Broker

Working Group Discussion:
XML & Canonical Data Models

Wednesday, March 17, 1999

9:00 Myra Spiliopoulou:
Data Mining for Service Integration

Ingo Schmitt:
Schema Integration for the Design of a Federated Database System

10:30 Coffee

10:45 Gregor Engels:
Schema Evolution in Federated Database Systems

Ulf Leser:
Query Mediation in Federated Information Systems

Felix Naumann:
Quality-driven Integration of Heterogeneous Information Systems

12:15 Lunch

14:00 Excursion (Uitstapje, Ausflug, ...)

Thursday, March 18, 1999

9:00 Bernd Krämer:
Enhancing CORBA Object Interfaces

Arne Koschel:
Configurable Event-triggered Services for COBRA-based, Heterogeneous, Distributed Information Systems

10:30 Coffee

10:45 Karl Aberer:
Extensible Databases as Middleware for Flexible Information Management

Stelvio Cimato:
A Methodology for the Specification of Java Components and Architectures

Jean-Marie Jacquet:
The Expressiveness of Coordination Models

12:15 Lunch

14:00 Robert Tolksdorf:
A Coordination Architecture for Workflows

Klaus-Peter Löhr:
Towards Accommodation of Heterogeneity in Software Architectures

15:30 Coffee & Cake

16:00 Working Group Discussion:
Software Architectures & Coordination

Friday, March 19, 1999

9:00 Results of the Working Groups:

Myra Spiliopoulou:
WG I: Applications & Processes

Mark Roantree, Karl Aberer:
WG II: XML and Canonical Data Models

Klaus-Peter Löhr:
WG III: Software Architectures & Coordination

10:30 Coffee

10:45 Final discussion & Closing

12:15 Lunch

2 Abstracts of the Talks

XML Broker

Karl Aberer

(joint work with Gerald Huck, Ingo Macherius und Peter Fankhauser)

The World Wide Web offers collections of information for almost every domain of interest. They can be used interactively via HTML based graphical user interfaces. Systematic research however requires the combination of information from multiple, independent sources. Our XML Broker offers tools for building such integrated information services.

As an exemplary scenario the XML Broker was used to construct a service to help golf players choosing a course. It integrates three independent resources: a list of golf sites, a route planner and localized weather forecasts. A typical query would ask for all 18-hole golf sites located up to 150 km from Birlinghoven, where it will not rain tomorrow.

From a users' point of view the XML Broker behaves like a database with HTML-based interface. However, it does not provide any data by itself, but retrieves information from internet resources on demand. The information gained is structured in XML-documents and administered in a warehouse. It is accessible using a declarative, XML-specific query language. Queries can be used to form integrated views on the information modeled. Query results are XML-documents themselves and can be processed further by XML aware applications or displayed using a fifth-generation browser and stylesheets.

Extensible Databases as Middleware for Flexible Information Management

Karl Aberer

Information management today requires in first place the ability to flexibly restructure and combine information of different origin and nature for new uses. Extensible (object-relational and/or object-oriented) database technology is one of the (many) important middleware technologies applied for that purpose. A flexible data model, storage and query support and transaction services implemented in a scalable and efficient way are the main assets, which have to be traded for relatively high installation, administration and development effort and functional limitations. Based on the experiences gathered from a number of projects

on information management based on extensible database technology, we analyze in this talk some benefits and limitations of the use of extensible database technology for flexible information management.

Integrating XML with Java for Active Documents

Paolo Ciancarini

(joint work with F. Vitali)

While in terms of markup and support for hypertext features there is nothing missing in the XML proposals that may relate to the documentation of the software process, there is still one problem to be faced to obtain a fully usable environment: the visualization of the documents.

Since XML elements have no predefined meaning or rendering semantics, it is up to the visualization software to provide it. This happens by mapping the documents' element names to the program's element names through XSL stylesheets. As it is clear, the sophistication of the final rendering thus is independent of the markup, but heavily depends on the sophistication of the rendering program. This is an extremely sensible consideration, but has its weak points: we can write all sorts of documents in XML, but we can display only those that do not have extreme rendering needs. In the current use, we can map XML documents onto HTML and display them on standard WWW browsers, but specialized DTDs for strange notation (mathematics and chemistry come to mind) require a specialized browser to be displayed correctly.

XSL also provides a set of required visualization objects, that is, objects that are known to be available in all XSL implementations, but these only cover standard typographical aspects of the visualization of text, and do not provide support for non standard requirements.

The displet approach, on the other hand, provides a generic solution to this problem. The idea behind displets is to create a generic rendering browser that can load and activate little independent software modules tailored to create specific visualization objects. The rendering browser would then activate these modules depending on the content of the document, and deliver all kinds of required rendering needs. Displets (display applets) are such little software modules, designed to be fully interoperable and to be activated by a generic display module. We implement displets in Java because we have standard Java enabled browsers.

A Methodology for the Specification of Java Components and Architectures

Stelvio Cimato

Traditional software development methodologies are insufficient to deal with the increasing complexity and the evolving requirements of modern software systems. Component based software development and software architecture descriptions are design techniques which facilitate the development of new applications by assembling reusable software components. However, such techniques and frameworks often lack formal notations to support formal reasoning about the final products. In this talk, we show a formal framework to develop components and software architectures for Java applications based on the Larch approach. We define a behavioral interface specification language for Java classes and a methodology to support the design of Java components and software system built of such components.

XML and Related Technologies: Overview and Research Directions

Wolfgang Emmerich & Anthony Finkelstein

The talk has two parts. In the first part, we present a brief overview of the eXtensible Markup Language (XML) and related technologies. We give the motivation for the development of XML, discuss how XML document type definitions (DTDs) can be defined. We show how hyperlinks between two instances of XML DTDs can be defined using XLink and XPointer. We then indicate how mappings can be defined between different XML documents using the eXtensible Style Language (XSL). We present the Domain Object Model (DOM) and show how it can be used to provide an application programming interface to a structured XML document.

In the second part, we discuss how we use XML in our research. We firstly discuss an approach to the management of consistency between distributed XML documents. This work is motivated by the fact that an increasing number of DTDs are specified and that these are supported by tools, such as Office 2000, UML CASE tools and standard browsers. The approach specifies consistency relationships and uses XSL to visualize inconsistencies between documents. In a second project, we use XML to define application-level transport protocols. This work is motivated by the observation that existing object-oriented middleware is not well suited for transmitting bulk structured data between distributed

system components. We suggest an integration between XML and CORBA by specifying interoperability between the DOM and CORBA object models. Implementations of this specification implement compilers that translate XML DTDs into CORBA/IDL interface definitions and provide transparent use of the same complex data structure in XML enabled and CORBA-compliant components.

Integration of Semi-Structured Information Sources with HyperView

Lukas Faulstich

The HyperView approach to integration of semistructured information sources is based on the graph data model CGDM and on views defined by graph-transformation rules. It uses a layered architecture where each layer is a view of the preceding layer at a higher level of abstraction. These views are computed by a novel demand-driven rule activation strategy. At the top layer view computation is triggered by user requests for dynamic HTML pages. The returned pages present to the user combined and restructured data extracted from the underlying information sources. Thus, the HyperView system acts as a virtual Web Site. In this talk, the HyperView approach will be presented using a case study from the field of digital libraries.

Integration through ViewPoints

Michael Goedicke

The problem to integrate software systems has at least two dimensions. One dimension is that different components use different representation schemes not only to represent their internal information but also for the purpose to communicate it with other components. The other dimension is that the controlflow in different components follows different rules as well. In the case of newly created components where the designer has full control over the internal workings of those components the integration task can be accomplished by properly adjusting those internal workings. However, in contemporary software development components are often opaque and can only be influenced in a limited way. In order to overcome the border between such components the techniques to integrate different specification methods as developed within the ViewPoint framework can be applied as well. The idea of integration by viewpoint is to encapsulate a component

in viewpoint and describe the interaction between components as interaction between the encapsulating viewpoint. The inconsistencies which may arise during the course of a component interaction can then be handled using the same mechanism as in the ViewPoint case. We describe briefly the idea and outline a scheme based on graph rewriting techniques for implementing this integration idea.

On the Process of Systems Integration

Wilhelm Hasselbring

Traditionally, the integration of heterogeneous information systems proceeds in a *bottom-up* process. Information stored in existing legacy systems is analyzed with respect to potential overlaps. The overlapping areas of related information sources are subsequently integrated. The integration of these isolated information systems is usually realized by means of mediators, federated database systems or such-like system architectures. Typical goals for the integration of existing information systems are the development of global applications that access the data from multiple sources as well as consistency management of information that is stored in related systems. We discuss the process of integrating heterogeneous information systems from different viewpoints:

- The process in which heterogeneous information systems are traditionally integrated in a *bottom-up* way and some problems with this approach.
- The process in which heterogeneous information systems could ideally be integrated in a *top-down* way to achieve more usable and scalable systems.
- A combined *yo-yo* approach to exploit the benefits of both strategies and to serve as a migration path form the traditional bottom-up approach towards an ideal top-down approach.

Our emphasis is on a discussion of the *process* of integrating such heterogeneous information systems. Semantic interoperability is necessary in this context to ensure that exchange of information makes sense — that the provider and requester of information have a common understanding of the "meaning" of the requested services and data. Particularly, we emphasize on the role of domain-specific standards for managing semantic heterogeneity among dissimilar information sources in the context of the Domain-Specific Software Architecture engineering process which has been introduced to promote a clear distinction between domain and application requirements.

The Expressiveness of Coordination Models

Jean-Marie Jacquet

A number of different coordination models for specifying inter-process communication and synchronisation rely on a notion of shared dataspace. Many of these models are extensions of the Linda coordination model, which includes operations for adding, deleting and testing the presence/absence of data in a shared dataspace.

We compare the expressive power of three classes of coordination models based on shared dataspaces. The first class relies on Linda's communication primitives, while a second class relies on the more general notion of multi-set rewriting (e.g., like Bauhaus Linda or Gamma). Finally, we consider a third class of models featuring communication transactions that consist of sequences of Linda-like operations to be executed atomically (e.g., like in Shared Prolog or PoliS).

Schema Evolution in Federated Database Systems

Silvia Kolmschlag, Gregor Engels

(presented by Gregor Engels)

In the last years, the coupling of operational and management information systems has become more and more important. For example, data warehouses and electronic commerce systems offer a uniform interface to different company databases. The heterogeneity of databases should be hidden to the user. This leads to the concept of Federated Database Systems.

A Federated Database System (FDBS) couples autonomous heterogeneous database systems to allow uniform and transparent access to the federated data of the component database systems (CDBS) via a common global interface, the federated schema. Global applications are able to work on data of the coupled CDBS whereas local applications are still able to run. But a schema is not defined fixedly. Caused by changing requirements of applications and users, a schema has to react on the new requirements to make the FDBS flexible, extensible, and reusable. Therefore, schema evolution in an FDBS has to be supported.

Schema evolution in an FDBS requires to consider schema updates on the local schemata of the coupled CDBS on one hand and on the federated schema of the FDBS on the other. Therefore we distinguish between local and global schema evolution. Schema evolution on these schema levels have mutual impacts on each other. Because the federated schema of an FDBS is constructed from the local schemata of the CDBS by schema integration, a schema evolution on

one schema level can effect the respectively other schema level and also the data level. The whole integration of the system is influenced.

The administrator of a system based on an FDBS needs support for developing an FDBS furtheron. In this work, schema evolution on the local as well as on the global schema level of an FDBS is investigated. For realising schema evolution we present two strategies that differ in the transparency of schema evolution for the other schema level and in the effect on the data. Therewith, we are able to support the FDBS administrator evolving an FDBS.

Configurable Event-triggered Services for CORBA-based, Heterogeneous, Distributed Information Systems

Arne Koschel

Todays distributed information systems are often collections of existing information sources and, as such, heterogeneous. Technical integration and access to heterogeneous sources is supported by CORBA for servers combined with Web technology for clients. Additional need however, is comfortable active functionality, e.g., for active user notification about information changes relevant to their work. Active functionality well known from Active DBMS-style ECA rules has proven useful for this purpose. Such active functionality, however is a gap in current CORBA functionality. To fill this gap for CORBA and to enhance the proven ADBMS-style active functionality for new heterogeneous, distributed worlds is the goal here. The talk thus contributes the concept, design, architecture, and implementation details of C2offein (Configurable Corba-based functionality for event-triggered information and notification), a service set contributing Active DBMS-style active functionality to CORBA-based systems. Moreover, since different enterprise applications need various kinds of active functionality, C2offein is widely configurable in its application specific definable architecture and overall functionality. The great benefit of such an approach is its flexibility and "smartness" compared to monolithic approaches like those traditionally used, e.g., in Active DBMS. In contrast to monolithic systems, C2offein's configurability offers the possibility of only investing effort and money for that active functionality which is really necessary for an individual application.

Metadata: A Key Issue for Systems Integration

Ralf Kramer

In distributed information systems, we are faced with two fundamental questions:

1. How do users find the information they are looking for?
2. How can we facilitate interoperability of services in such systems?

The basic claim of this talk is that metadata are the key issue to solve these problems and, hence, for systems integration. We use federated environmental information systems (EIS) as the application background as they are highly heterogeneous, both technically and semantically. Metadata can simply be defined as data about data. With respect to EIS, there is a whole variety of relevant metadata standards, including GELOS, CDS, ISO/TC 211, and the XML-based RDF. We explain their relationships. Based on the I3 reference architecture, we develop the general architecture for federated EIS in which metadata-based catalogue system provide information services. Semantic integration is identified as one of the main challenges whereas Web technologies greatly facilitate technical interoperability. Thesaurus federations provide an approach to solve problems imposed by heterogeneous vocabularies. The importance of taking into account both static and dynamic aspects when integrating systems is illustrated using the European Environmental Information Services Project (EEIS). A brief RDF example concludes the presentation.

Enhancing CORBA Object Interfaces

Bernd Krämer

Communication middleware such as CORBA implementations and DCOM support the integration of software components implemented in different programming languages to distributed applications. Standardized component interface descriptions, interaction protocols, and communication mechanisms allow these applications to interoperate transparently in heterogeneous operating environments. For certain types of applications, however, the information expressible in IDL interface specifications are not sufficient to support a reliable component integration. Important properties of security-, safety-critical or reactive components such as functionality, dynamic behavior, timing, or synchronization constraints cannot be documented formally, let alone checked automatically.

In this work we propose solutions for adding declarative specifications of such properties to component interfaces and automatically synthesizing code that instruments corresponding dynamic checks. Independently from the concrete syntax and semantics of such specification elements, we present a collection of design patterns that allow the designer to seamlessly integrate the synthesized code with the code frames generated by standard IDL compilers. We study these approaches along the concrete example of extending CORBA IDL with synchronization constraints and evaluate several implementations, solely based on standardized features of the CORBA standard.

Integration Plattforms for Distributed Information Systems

Ralf-Detlef Kutsche

"Systems Integration" as the general topic of the seminar has an extremely broad variety of concepts, methodologies and enabling technologies in its background, which split up into quite different, however related areas, according to the viewpoints, aspects and specific goals in a given context and requirements of an integration task.

This talk focuses on the particular aspects relevant in integrating distributed, heterogeneous information sources into large, federated information systems, in our terms also called: information infrastructures. Since one of the main tasks in software development in this area today is the reflection of the customer needs in integrating their well-established legacy solutions into larger contexts, the evolution of large information infrastructures must be considered as basic requirement, in our terms: "continuous software engineering".

Information infrastructures can be characterized by three (almost) orthogonal dimensions:

1. the technical dimension, covering the aspects of distribution over space via different networks, protocols, etc., and also the aspects of heterogeneity in hardware, operating systems, application software etc.;

2. the organisational dimension, dealing with complex responsibilities in enterprises like industrial companies, government, public administration, etc., where coping with the required autonomy in systems operation, administration, content provision, business process management is important;

3. the semantic dimension, where heterogeneity in data models, data structure and granularity as well as the important questions of semantic overlaps in

information resources, inducing redundancy, inconsistency etc. must be handled.

Along the classical separation of SWE into the layers of analysis, design, implementation and running system, we mainly address the tasks of analysis and design. Our methodological approach is based on four principles, namely:

- uniform modelling of legacy and new components, using appropriate combinations of modelling and specification languages, trying to establish a maximum of uniformity in syntax and semantics;
- using metainformation on several abstraction levels in a domain oriented (by modeling metainformation explicitely and using metadata standards from the respective application domains) and, additionally, generic way (as using general metadata approaches, formats and representation techniques);
- using the reference model of Open Distributed Processing (RM- ODP) as a conceptual framework, particularly the viewpoint driven separation of concerns (N.B. The classification given above explicitely refines to the ODP viewpoints: enterprise v. (2), information v. (2, 3), computational v. (1, 3), engineering v. (1) and technology v. (1));
- evolution based on the integration platforms philosophy, to be explained in more detail now.

The term of an integration platform can be considered as a general notion of handling particular integration tasks in an organized, potentially transparent way. In a more technological and engineering view, taking systems interoperation as one of the predominant tasks in systems integration, we shall consider object- oriented interoperation platforms like CORBA or DCOM. In order to deal with persistence as most relevant aspect of information systems, standards like the SQL language standard, like the ODMG object database standard or the underlying object model and structure of persistence services can be considered as platforms, supporting information integration. The same applies to document standards, to metadata standards or even to the metalevel (metamodels) of existing modeling languages like the Unified Modeling Language UML - they all provide platforms to allow for easier exchange, understanding and processing of relevant "entities" in an integration scenario. Finally, very important, the software process itself takes considerable advantage from following strict guidelines in using patterns, particularly in design, as by the well-known design pattern catalogues, but also in the analysis process, or with respect to global component architectures.

Future work in more detailed description of patterns, to be used in integration platforms, will be one key in succeeding with our concern of the evolution of large information infrastructures.

Query Mediation in Federated Information Systems

Ulf Leser

We propose a method for the translation of queries in a scenario of independently existing and evolving component schemas. The translation algorithm is based on query correspondence assertions, which are set-equations between two queries against different schemas.

We develop our method for the purpose of building flexible and highly evolvable mediator-based information systems. We describe physical data sources through a relational export schema (assuming the existence of a wrapper), and aim at querying the federation through a relational, global schema. The task of a mediator is to translate a global query in semantically equivalent sets of queries against the export schemas, which in our cases uses semantic descriptions of the export schemas with respect to the global schema in the form of QCAs.

Since QCAs are essentially rules, our method offers a high level of "declarativity". Evolution of sources (or mediators) can often be encountered by simply changing rules, thus avoiding changes in affected schemas.

iWebDB — An Integrated Web Content Management System

Henrik Loeser

Since its emergence, the Web, i.e., the number of users, of Web sites, and the amount of accessible data resp. the number of documents is constantly increasing. While in the first time, information providers, e.g., institutions, and enterprises, have offered only "statical" documents, they, today, have migrated their Web sites to so-called Web Information Systems offering latest information and periodically updated documents. If different persons or groups maintain these Web documents, manipulations must be coordinated in order to keep document contents and hyperlinks consistent. Moreover, document validity must be managed, and generating statical documents triggered by changes to production data must be provided in order to keep the offered information up-to-date. In addition, a seamless integration of file-system-based data must be supported to the users to allow the employment of user-preferred editing tools. In my talk, I present iWebDB, a Web Content resp. Document Management System (WCMS/WDMS) integrating its functionality into an object-relational DBS. Thus, HTML documents are treated as first class DB citizens, and problems, such as inconsistent local hyperlinks, outdated documents, and obsolete index data, can be avoided.

Towards Accommodation of Heterogeneity in Software Architectures

Klaus-Peter Löhr

As there is no universal agreement on what constitutes a *software component*, the least common denominator may be the possibility of modelling components as state machines. Talking about states and state transitions allows us to abstract from the actual nature of the component's interaction with its environment. It is therefore suggested to use interface descriptions as known from object-oriented languages as the syntactical basis for component specification. As this is *not* meant to prescribe a specific interaction style (such as "imperative code can invoke operations exported by a component"), the term *abstract interface* is suggested. An abstract interface specifying two events *in* and *out* might hide, e.g., a Unix filter program. This goes beyond CORBA IDL language mapping because the program does not follow the invocation style of interaction.

Using abstract interfaces implies that different interaction styles can be used on both sides of an interface. The *glue code* to be provided when assembling a system from components may thus be chosen from the full range of languages, from imperative to more declarative languages to architectural description languages. This would allow for heterogeneous architectures (which use different styles in different subsystems), alleviate integration of legacy systems and support component-based development.

If accommodating heterogeneity across abstract interfaces would require manual construction of all kinds of wrappers, not much would be gained. Therefore, the goal of this research is to define mappings to and from an abstract interface style for several interaction styles and to investigate the construction of tools for automatic wrapper generation.

Quality-driven Integration of Heterogeneous Information Systems

Felix Naumann

(joint work with Ulf Leser and Johann Christoph Freytag)

Integrated access to information that is spread over multiple, distributed, and heterogeneous sources is an important problem. While much work has been done on query processing and choosing plans under cost criteria, very little is known about the important problem of incorporating the information quality aspect into query planning.

The talk described a framework for multidatabase query processing that fully includes the quality of information in many facets, such as completeness, timeliness, accuracy, etc. We seamlessly include information quality into a multidatabase query processor based on a view-rewriting mechanism. We model information quality at different levels: First, we perform quality-driven source selection and continue only with the best sources. Second, we compute query-dependent information quality of the view definitions that describe the content of sources. Finally we determine the overall quality of plan alternatives by aggregating these information quality scores to find a set of high-quality query-answering plans.

Integration of Object Life-Cycles in Object-Oriented Design

Günter Preuner

The design of object-oriented database schemas comprises the definition of structure and behavior of object types. For representing the behavior of objects, we use life-cycle diagrams that show for each object type which sequences of activities can be invoked on its instances.

In large systems, the database schema cannot be designed by a single person but will require the cooperation with potential future users of the database. These users know the application domain from everyday work with paper forms and thus know to specify those parts of structure and behavior of object types that are relevant for their work. Thus, different users know different parts of the life-cycle diagrams in detail, roughly, or even not at all.

This talk presents the core of a method to integrate views of life-cycle diagrams. In this method, first correspondences between object types are identified and object types are defined in the integrated schema. The life-cycle diagrams of these object types are defined based on the life-cycle diagrams in the views.

The OASIS Architecture: Legacy System Migration and Federated Schema Construction

Mark Roantree

Healthcare systems have often been used to demonstrate multidatabase prototypes as they contain a wide range of information systems for specific tasks within the organisation. Recently a number of emerging technologies have enabled healthcare institutions to build their own layers of interoperability. One example is the use of wrapper technology to extract a universal relation from MUMPS based legacy systems. For this reason, some healthcare IT departments are discovering migration solutions whereas before they were content simply to transfer information between systems. In this paper, we present an architecture which attempts to offer the basis for either approach.

Schema Integration for the Design of a Federated Database System

Ingo Schmitt

Integrating heterogeneous schemata with overlapping class extensions and types is an essential task in federated database design. In this scenario conflicting inheritance hierarchies have to be merged. Inheritance hierarchies often occur explicitly in object-oriented databases as well as implicitly in relational databases. In the presented approach, the problem of integrating different inheritance hierarchies is transformed to the theory of formal concept analysis. Hence, mechanisms of that theory can be used to derive a concept lattice which can then be regarded as the integrated schema.

After having investigated the power and complexity of concept analysis algorithms a better algorithm was presented tailored to the problem of schema integration. The new algorithm has polynomial complexity and helps to optimize the resulting hierarchy with respect to certain quality criteria, e.g. number of classes and null values. An example demonstrates the practicability of the approach to integrate heterogeneous schemata.

Global Version Management in a Federated Design Environment

Martin Schönhoff

Integrated engineering environments, based on federated database technology, are a means to control the integrity of and dependencies between product data created in many different engineering applications. Continuing the engineers' tradition of keeping different versions of drawings and documents, most engineering applications support the management of different versions of a product and its parts. Consequently, federations in engineering environments should provide version management on their global layer in order to support homogeneous global access to versions from different local systems and to provide system-wide consistency of versioned data.

The talk introduces a simplified federated turbine design environment to present the problems which are specific to global version management in a federated system. It then investigates and evaluates these problems as well as concepts to their solutions in the context of the example environment. Finally, a generalisation of selected solutions for a wider range of application domains is discussed.

Data Mining for Service Integration

Myra Spiliopoulou

Providers of web-based services are interested in monitoring the usage of their services in combination with those of other providers. The providers of services invoked together frequently may decide to cooperate and replace these services by a jointly developed "integrated" service that offers additional features.

We propose data mining to discover services appropriate as a basis for company cooperation. In particular, we model the activities of a user application as a sequence of service invocations recorded in a log. We then can apply our pattern discovery miner WUM to find groups of sequences that satisfy semantical, structural and/or statistical constraints. Those constraints are defined by the mining expert to reflect what is "interesting" enough for a company to launch a cooperation upon, according to the company's strategic policies.

WUM discovers interesting patterns within a log of records over a time period. An equally important aspect is the evolution of those patterns over time. Are two services used together constantly, or are there bursts of joint usage? How is each service used outside the time intervals of high activity? To help answering such questions, we are currently specifying the important features that describe service interactions in form of time series.

A Coordination Architecture for Workflows

Robert Tolksdorf

The combination of workflow systems and the Internet is hoped to be beneficial for both domains of technology. We propose an architecture based on emerging Web technology that allows for a flexible coordination of workflows over the Internet.

We understand each step in a workflow as the rewriting of a document into another one by some activity. In our architecture, the documents are marked up following some application dependent XML-DTD.

We introduce a workflow engine that is able to process the description of an activity for a step. The activity can be an automatic one which simply processes the document, an external one that invokes an external application, a user step which is not computer supported at all, or a meta-step that transforms some activity description.

In our architecture, an XSL processor is the component of choice to transform an XML-based document into another one. According to the kind of step described, different XSL-rules are supplied to the engine.

The overall workflow is defined as a graph in which activity in nodes can be performed when the documents required on the incoming arcs are available to be transformed into other documents. In our architecture, they are stored in a distributed document space that can be accessed using Linda-like primitives.

In order to be executed by the XSL engines, the workflow graph has to be compiled in a series of steps of the named kinds plus coordination steps. In our architecture, we take the Workflow Process Description Language as defined by the Workflow Management Coalition as the linguistic basis for the expression of the workflow graph. We implement a respective XML-DTD to represent the workflow as an XML document. The named transformation of the graph into steps thus is subject to processing by an XSL engine guided by a set of meta-rules. These rules define how the coordination constructs embedded in the graph are compiled into primitive coordination steps.

Our architecture combines standard Web technology with publicly available implementations with workflow by concentrating on coordination aspects. It provides a high degree of flexibility by decomposition of the workflow graph and a highly decoupled architecture. Subworkflows can be easily moved, delegated and distributed. The use of explicit representation of workflows and coordination rules allows for changes in the graph as well as in its interpretation.

Our vision is to provide an open, easily accessible, wide area workflow system in which people exchange the descriptions of the processes that they use in a highly distributed and flexible manner.

Experiences with Current Methods of Integrating Hospital Information Systems

Alfred Winter

A Hospital information system (HIS) is that subsystem of a hospital, which is comprising all information processing actions and the concerned human or mechanical actors in their role as information processors. HIS must encompass information as well as medical knowledge, computer-supported as well as non-computer-supported information processing, all divisions, all buildings, and all categories of staff of the hospital. Even for the computer-supported part of the HIS no vendor can offer a software product which covers it all. Instead of developing appropriate software for their own it is the main task of the hospital information system management to combine or "glue" together commercially produced softwareproducts.

Thus HIS usually consist of a considerable amount of autonomous application systems with (partly) redundant data bases (A(n)-architecture) [3].

As a means for integration communication servers [1] are used to provide asynchronous message exchange. Messages should conform to communication standards as HL7 [2]. Nevertheless there are some disadvantages: even standardised communication interfaces are expensive, complex transactions (e.g. appointment scheduling) are hardly to realise, fast and synchronous communication (retrievals in databases) is hardly possible and replication of data is sometimes impossible (e.g. with radiographic pictures).

[1] Gräber S. (1996). Communication Services for a Distributed Hospital Information System. Methods of Information in Medicine. 35(3), 230-341.

[2] Ohe K., Kaihara S. (1996). Implementation of HL7 to client-server hospital information system (HIS) in the University of Tokyo Hospital. J Med Syst. 20(4), 197-205.

[3] Winter A. (Hrsg.) (1996). Rahmenkonzept für die Weiterentwicklung des Klinikuminformationssystems des Universitätsklinikums Leipzig. Universitätsklinikum der Universität Leipzig: Leipzig.

3 Working Groups: Issues and Results

Working Group I: Applications & Processes

Myra Spiliopoulou

Working Group 1 addressed issues under the general title "Applications & Processes" in the context of systems' integration. A deep terminological discussion on the notions of process, application etc was decided to be of limited usefulness, while defining a precise notion of the term "integration" is indispensable. Two orthogonal categorizations were agreed upon. The first covers the phases of analysis, design and implementation/run-time. The second distinguishes between high-level interworking, interoperation and low-level interconnection. The level of integration agreed upon was that of *interoperation*.

The notion of integration is goal-oriented. Depending on the goal, communication is a means for integration. At any case, integration may not focus solely at structural issues; semantic problems must also be resolved.

Integration of specifications is the first step to systems' integration. Beyond data, behaviour must be integrated. There is a variety of models for describing them. Mapping those models to a universal canonical model is not always appropriate. In this context, XML is not a solution to the integration problem.

The final issue concerned the cost of integrating a large system, whereby the term "large" is ambiguous. It is recognized that the cost of integration covers expenses for equipment, installation and maintenance, training of the users and time. Ad hoc solutions are less costly than the establishment of a federated DBMS. A reliable budget estimation for a FDBMS is very difficult, for the integration of software it is almost impossible.

Working Group II:
XML and Canonical Data Models

Mark Roantree, Karl Aberer

(presented by Karl Aberer)

The focus of the workgroup was to discuss whether or not XML could function as a canonical data model for systems integration, or if it could in some way, support an existing canonical model. The workgroup identified a number of different themes under which to discuss the model:

- XML's relationship with Java/C++
- XML's relationship with CORBA
- Query Languages for XML
- XML's role in integrating data and services
- Attaching semantics to XML
- Can XML be a CDM, or is it just a model for data transfer?

XML's relationship with Java/C++ Java can make use of the Domain Object Model (DOM) to interact with XML. The main point to emerge was that it makes no sense to combine XML with Java, as this defeats the purpose of a language like XML which should be visible as an information transportation medium.

paragraphXML's relationship with CORBA

This topic sparked quite a lot of debate and the Boeing Project which employs Orbix and XML to integrate a large number of disparate systems provided a good reference point.

For large scale projects such as the Boeing Project it was generally agreed that CORBA could act as the main communicating technology, but it was felt that some high level technologies such as XML also has a role to play (where it made no sense to use CORBA). One of the problems pointed out by the group was the requirement to create Document Type Definitions (DTDs) for every time of object to be transported. Although it was felt that DTDs could be specified for entire domains, it was agreed that there would be many cases where DTDs would need to be constructed for isolated objects.

The concept of XML replacing IDL was also raised but quickly dropped as being non-viable. CORBA has a specific role for defining behaviour and managing the integration of behaviour, an area where XML was lacking.

The failure of SGML was also raised, and questions were asked as to whether or not XML could avoid the mistakes (complexity) of SGML. Finally, XML was highlighted as a possible replacement for CGI.

Query Languages for XML Details of upto four communities defining different query languages were provided, with XQL and XMLQL being the most prevalent current query language proposals. Microsoft is supporting XQL, XWL update facilities can be expected from the providers.

What emerged from this topic is that there are query languages currently available for XML (a good thing) but no query language has emerged as a standard (bad thing). In fact, different query languages currently offer different types of output as a result of XML queries (documents, elements sets, restructured documents).

XML's role in integrating data and services The focus of this theme was on DTDs. The question of strategy was raised: do we merge two DTDs or can we simply map the same document to different DTDs? The idea of the query language being used to solve the integration issue (possibly through the use of constraints) was also raised.

Can a generic DTD like XMI be used to exchange schemata? In general, it was felt that using XML would make it easier to exchange schemata but that the same issue of semantics still exists: it has just moved to another platform. In fact, it was felt that DTDs were not as rich as database schemas.

Attaching semantics to XML This topic focused on the idea of extending XML to include semantics or some form of rules. The group felt that this was a bad idea, as it wasn't possible to achieve properly. We were also reminded that SGML's reason for failure was its complexity. If XML is to succeed, it should use its simplicity as a key strength.

Can XML be a CDM, or is it just a model for data transfer? The (almost) unanimous feeling is that XML cannot be a CDM, but should be used where it is strongest: the transfer of data.

Working Group III: Software Architectures & Coordination

Klaus-Peter Löhr

As this was the last of the three workshops, there was a strong feeling among the participants that we should walk away with some general insight into the overall theme of the seminar. So after surveying several topics that had been suggested for discussion, we decided to focus on *patterns of integration*.

The unevitable quest for a precise definition of terms (pattern, integration) was quickly resolved for "pattern" — to mean a recurring structure in space and time that is revealed by abstraction of some kind. Understanding "integration" turned out to be harder. There were widely varying opinions on what constitutes an "integrated system" or a "system resulting from integration", ranging from loosely coupled systems to supersystems where the individual components are no longer recognizable.

An agreement was reached that there are three — largely orthogonal — impediments to integration: *heterogeneity*, *distribution* and *autonomy*. Different degrees of integration will exhibit different degrees of masking heterogeneity, hiding distribution and accommodating autonomy. So a more or less integrated system would be placed somewhere in a three-dimensional space, and the distance from the origin would be a measure of integration.

Typical approaches to coping with heterogeneity, distribution and autonomy were identified to follow four important patterns:

(Heterogeneity:)

1. *Homogenization*: Introduce a new, well-chosen "canonical" entity in addition to the given heterogeneous entities. Define mappings between these and the new entity. Support those mappings by providing wrappers, mediators, adaptors and similar devices.
2. *Direct accommodation*: Define pairwise mappings between the participating entities. Support those mappings by providing bridges for all pairs where accommmodation is indeed required.

(Distribution:)

3. *Remote invocation via proxies*: This well-known approach is the basis for several middleware systems and can achieve a high degree of invocation transparency.

(We would have loved to come up with a pattern for choosing appropriate middleware in a systematic way. Unfortunately, we didn't; so the issue is left as a topic for further research :-)

(Autonomy:)

(This pattern was not developed in the working group; it was suggested in the following plenary discussion.)

4. *Higher authority*: Existing authorities agree to give up part of their autonomy in order to allow for some coordination exercised by a higher authority.

While each of these observations is not radically new, the members of the working group liked the "integrated view" on systems integration that was achieved.

4 List of Participants