On the Road to Behavior-Based Integration

Markus Stumptner, Michael Schrefl, Georg Grossmann

University of South Australia
Advanced Computing Research Centre
5095 Mawson Lakes SA, Adelaide, Australia
{mst|cismis}@cs.unisa.edu.au

Abstract

Integration of autonomous object-oriented systems requires the integration of object structure and object behaviour. Research in federated information systems has so far mainly addressed integration of object structure. The integration of object behaviour carries with it the promise to finally move integration tools beyond the state of the art at which they have (roughyl) existed for the better part of a decade, and we are embarked on a project that examines this problem in terms of object life cycles. The topic is also increasing in actuality since it also applies to the integration of business processes, which is typically required when companies merge or enter into consumer-producer relationships, and results can be expected to naturally extend to the arena of web services. Ultimately, the goal will be to define consistency criteria for behaviour integration and apply them in integration tools that guide the definition of global behavioural views upon autonomous object-oriented systems.

Keywords: object-oriented databases, database design, federated information systems, behaviour integration, business process integration, web services, metaclasses

1 Introduction

Over decades one of the central lessons in information system design and software engineering has been that the ad hoc development of information systems is generally a highly error prone and risky approach. The history of information systems is a series of stages in which models, procedures, guidelines, and criteria were developed to allow designers to judge the validity of the systems they have produced. These models at the same time permit to control the design process and, by identifying problems the moment that they arise, prevent them from becoming severe hindrances in later stages of the development process when large parts of the system may depend on what turns out to have been an incorrect decision. This is nowhere more visible, and more difficult, than when dealing with the integration of information systems.

In a number of differing guises and based on a set of differing, but nonetheless related, technologies, "Integration" is one of the driving themes in current database and applied computing research in general. A recent special issue of the *Communications of the ACM* [cac02] and several articles in subsequent issues dealt with integration topics. Whether at the level of classical database applications, web services, workflows, integration is a matter of significant concern.

It is well accepted that, like database design, system integration is an *engineering task* which cannot be fully automated. The methods and techniques developed to be used by a system designer during integration will comprise informal design guidelines as well as formal consistency criteria for checking whether the behaviour of objects in the federated schema is consistent with respect to the behaviour of objects in the export schemas (see Section 4.3). This will enable designers to identify and correct problems in the design of such systems before development progresses to the stage of detailed code development, generally resulting in high cost savings in development and later maintenance and extension.

However, a key aspect of existing research on the integration of federated information systems is that it has concentrated almost exclusively on the structural aspects. Past research in FIS systems (e.g. [BE96, Con97, GSSC95c, KS95, PS98a, Sch98, SN88, SN90]) has concentrated on this aspect almost exclusively. Integration of object behaviour has received some attention, but only at the level of single operations (or "activities" at the conceptual level) [CEH+97, VA97]. Integration of behaviours, both more complex and much more far-reaching in the benefits reaped, has so far been neglected. In no less than three international workshops on Engineering of Federated Information Systems [CEH+97, CHH+99, HvdHH+00], integration of behaviour kept being identified as an open problem and an important future research issue. It is a commonly recognized trend in object-oriented information systems development that the development of languages, notations, methodologies, or design policies tends to focus initially on dealing with issues of object structure, because structure is in general less complex and easier to handle than behavior. Nevertheless, rigorous guidelines for analysing and integrating behavior will assist the designer in approaching behavior integration in a structured rather than an ad-hoc manner. This results in a substantial increase in development productivity and significantly facilitates later extensions and modifications. The goal for the upcoming years clearly is to develop methods and techniques for the logical integration of object *behaviour* which may well extend to the integration of whole object-life cycles.

The paper is structured as follows. In Section 2 we first introduce the classical FIS approach as a reference model, and explain both the challenges and promises of behavior integration in terms of this architecture. In Section 3 we then examine existing work that already addresses or highlights some of the problems that have to be faced, and in Section 4 finally describe our current approach for addressing the tasks still at hand.

2 Federated Information Systems

Decentralised computing and the need for interoperability have led to the development of federated information systems systems. A federated information system (FIS) is an integration of autonomous heterogeneous software systems (often database systems, in which FIS are called federated database systems (FDBS) [AS90]), where both global applications accessing multiple systems, and local applications are supported [HvdHH+00]. The techniques for FDBS have been successfully applied to many application areas such as engineering design, healthcare and banking [HvdHH+00]. According to the 5-schema architecture [SPSG00], generally accepted as the basic structure of FIS [CEH+97, CHH+99, HvdHH+00], the contents of the local pre-existing systems are described by *local schemas* in possibly different native data models. The local schemas are translated into *component schemas* of the FIS using a common and semantically rich data model [GSSC95c, NS89]. The parts of the component schemas to be shared in federations are defined in *export schemas*. *Federated schemas* provide an integrated, common view on several (but not necessarily all) export schemas. End users access the FIS via *external schemas* derived from federated schemas.

2.1 Federated Databases

The success of FIS technology depends on the availability of techniques for integrating export schemas into federated schemas. Without integration, global applications would need to access several export schemas directly and would need to resolve semantic conflicts, which occur frequently when local systems have been developed independently. This is not at all desirable for two reasons: (1) Global applications accessing the same export schemas need to resolve semantic conflicts redundantly. (2) Global applications become directly dependent on local changes to the export schemas.

This architecture naturally implies that one of the most important operations is the integration of several export Schemas into a FIS, where the schemas are expressed in a high-level, conceptual object-oriented model. This requires the integration of object structure and object behaviour. The immediate question is at which level behaviour integration should be examined; it should be high enough to allow a clear expression of basic object behaviour without being muddled by implementation details to the degree that a clear solution to the problem becomes impossible. The best level for this purpose would be at the level of object life cycles, which capture the development of objects over time by formalisms based on state charts [EKW92, RBPE91] or Petri nets [KS91a, KS96]. Work on this level, although so far limited to straightforward modelling instead of integration, has so far been undertaken specifically because it allows unambiguous manipulation of object semantics with acceptable effort and intuitive presentation to the user or developer, properties which are just as desirable for the problems to be handled in the current proposal.

Various projects on FIS have been carried out during the last decade (cf. [Con97, CEH+97, CHH+99] for an overview). Fundamental concepts for object-oriented database integration go back more than 15 years (e.g. [KS95, NS89, SN88, SN90]) and have found their realisation in various prototype systems (e.g. [GFFK94, KDN90, KFM+95]).

Although the different approaches to schema integration vary in details, the integration process typically comprises the following steps:

1. *Schema translation:* The local schemas represented in native data models are translated into component schemas represented in a canonical data model.
2. *Schema enrichment:* Most approaches use a semantically rich data model as the canonical model in order to facilitate the integration process. Then, if the native schemas are formulated in semantically poor data models, schema translation must be complemented by a schema enrichment process, to add semantics and knowledge not captured in the local schemas (e.g., generalisation or aggregation hierarchies) in interaction with the human user.
3. *Identification of correspondences* between elements of the component schemas[1] to be integrated. In this context a large set of semantic heterogeneities have to be considered (e.g, naming, scaling or type conflicts).
4. *Integration:* Found heterogeneities are resolved by renaming, rescaling or restructuring, and corresponding schema elements are integrated. Generalisation is a frequently applied technique to integrate corresponding object types in object-oriented integration [Con97].

Although many schema integration techniques make the above distinction between a *pre-integration phase* in which correspondences between elements in export schemas are detected and an *integration phase* in which the export schemas are integrated to a federated schema according to the detected correspondences, practical experience tells that these phases should be allowed to overlap and that users often ask to develop the federated schema incrementally (productive prototype approach). Integration methods need to take this proclivity into account.

It has taken a long time for theoretical FIS research on data integration (cf. [BE96, Con97, CEH+97, CHH+99, HvdHH+00]) to find its way into commercial practice. Key commercial players such as IBM offer now various middleware tools to build FIS (e.g., DB2 datajoiner). Mike Stonebraker observed in "Intelligent Enterprise" (April '99) that "data federation systems are coming" and "you can achieve scalability to the enterprise only by logical integration". Although not necessarily couched in standard FIS terminology, this capability is being pursued with a new sense of urgency given the increasing importance of B2B e-commerce applications that envisage the interoperation of multiple businesses or companies subject to their shifting business relationships, including mergers and acquisitions. Especially in the latter case the effort required for integrating different IT solutions on similar domains is often a critical problem that will highly profit from sound methodological support.

As identified by [CEH+97, CHH+99], the integration of object structure is well established, but the integration of object behaviour is an open issue [HvdHH+00]. Clearly, to carry over and extend well-known techniques for data integration to behaviour integration is both a challenging and useful topic of research, and the promise of using object lifecycles as the basis for behaviour integration was also pointed out in a dissertation on schema integration [Sch98]. In [Fra97], the author has addressed a related topic, view integration for object-oriented databases. The main difference to schema integration for federated databases is that in view integration the objects are virtual in the component schemas and real in the integrated schema, whereas in federated databases the situation is just the opposite [Sch87]. Overall, it is fair to assume that methods and techniques developed for behaviour integration in federated databases will drastically expand the scope and expressiveness of the state

[1] Without compromising the 5-schema-architecture, we do not distinguish between component schemas and export schemas. This distinction is irrelevant for the schema integration problem.

of the art in information system integration, which has so far been focused on mere structural integration. In particular, the incorporation of object life cycles into the integration process will constitute a major step forward in federated information systems, which have until now supported only the integration of data, but neglected the integration of the processes manipulating these data (cf. [CEH+97, CHH+99, HvdHH+00]).

2.2 Business processes

Although not usually included in the a major *application area* of this integration of object-life cycles is the integration of business processes. The traditional techniques of structured analysis and design are being increasingly replaced by object-oriented modelling approaches in the development of business information systems. Following an object-oriented approach, data about business cases is represented by the structure of objects and processing of business cases is represented by the behaviour of objects, whereby the processing of business cases over time is reflected by object life cycles [BPS97]. Typical scenarios which generate the need to integrate pre-existing, autonomous business processes are the merger of or the establishing of a consumer-producer relationship between two companies. As depicted in [SPSG00], any number of business entities can interact in B2B transactions using B2B framework mechanisms, which means these frameworks must provide the capabilities described for Federated Information Systems above. The conclusion the authors draw is that semantic conversions are needed, with integration of legacy systems and enterprise applications being important to the success of such systems, and that existing frameworks "do not address these issues sufficiently." In the December 2000 issue of ACM SIGMOD Record, Hasselbring et al. [HvdHH+00] point out that the success of new component technologies within industrial settings is dependent on usable solutions for behavior integration. The next section describes our approach to address this urgent topic, which, according to Hasselbring et al. "has evaded researchers for a long time", by developing a sound methodology and valuable tool support for behavior integration [HvdHH+00].

3 Languages for Process Descriptions

A fundamental decision concerns the selection of the canonical data model. The canonical model must be rich enough to model the semantics expressed in the local schemas, as well as any additional semantics obtained from the enrichment process in order to facilitate the detection of correspondences between elements of the component schemas [GSSC95c, GSSC95b]. Availability as implementation model and wide acceptance are other important criteria [Sch98]. Object models, mainly ODMG, are generally used as canonical models [CEH+97, HvdHH+00], but ODMG is not semantically rich enough (cf. [CHH+99]) to capture object life cycles, as opposed to design notations that are, e.g., based on state charts (e.g. [EKW92, RBPE91, RJB99]) or Petri nets (e.g. [KS91a, KS96]). Our own research has used Object/Behaviour Diagrams (OBD) as starting-point in the project for several reasons:

1. Basic OBD constructs have a very close correspondence to object-oriented language concepts (cf. Table 3 in [KS91a]), which enhances comprehensibility and facilitates translation to an implementation model
2. later extensions, such as refinement hierarchies [Sch90], local referential integrity [KS92], labelled behaviour diagrams (LBD) [SS02], and the distinction between types and classes [LOS97], provide important richness which can be utilised during the integration process,
3. commercial tools for modelling business processes based on Petri nets are available (e.g., [OSS97]).

In addition, we have demonstrated before [SS00a] that specialisation in OBD (and its inverse, generalisation, which is used for integration) allows to distinguish clearly between extension (i.e., adding new features) and refinement (i.e., considering some features in more detail), whereas such a clear distinction is not always possible in UML [VA97]. Most importantly, given that a formal definition of UML is still in a state of flux, we have been successful in the past in solving issues of model semantics in OBD and transferring the results to UML [SS00b]. After using OBDs as initial research vehicle, the investigation of how our results can be carried over to UML is a natural next step.

Object behaviour diagrams were originally presented as a graphical design notation for the design of object-oriented databases [KS91a] and later extended for the modelling of business processes [BPS97]. Object diagrams are based on semantic data model concepts, and behaviour diagrams rely on Petri nets (but deviate from them in specific concepts significantly, see [KS91a, KS96]). A behaviour diagram of an object-type consists of activities and states, corresponding to transitions and places of Petri nets, and instances (more exactly, object identifiers) correspond to tokens. Activities take time; objects reside in an activity state during an activity execution.

3.1 Integration concepts

Initial ideas on behaviour integration have been presented in earlier work [PS98c, PS98b, PCS01].

- In [PS98c], the integration approach was outlined on a very specific subcase (disjoint object types are integrated) and for a specific consistency criterion (observation consistency).
- [PS98b, HvdHH+00] considered view integration from the observation consistency perspective.
- [PC99] deals with the integration of different views on object behavior.
- In [PS02] a 3 level integration architecture was examined which permits the separate modeling of business process (background logic), workflow, and webflow (presentation). As a result, a certain degree of modeling independence between the different levels can be achieved. Integration between different processes could occur at either level. Integration at a higher level will imply integration at the lower levels.
- In [PS02], formal correctness criteria for e-business-processes are described, and an algorithm that determines a behavior-consistent composition of web services is introduced.

3.2 From databases to business processes

Even though UML is the de facto standard in object modeling, in the last years, a number of new conceptual languages for describing application behaviour have been

developed, mostly with a view towards modeling business processes and web services. So far no standard language for modeling business processes and their behavior has emerged.

We give a quick overview of the different languages or language families and their properties, and contrast them with other existing languages. One feature that virtually all these languages with the exception of BPEL4WS share is the absence of support for compensation, an important ingredient of any transaction model that is to be suitable for a web service environment.

Petri Nets: In [vdAvH02] 'high-level coloured Petri nets' are used to model workflows. The approach provides a graphical and expressive notation and supports a life-cycle model. However, there is no special consideration for modeling exceptions and compensations.

OBDs: A Petri net based formalism with additions for modeling software execution.

UML Activity Diagrams: UML is a widely used modeling language with many different notation types. Activity Diagrams (AD) are the notation suggested for the description of business processes. Their main drawback is the limited expressiveness for parallelism (limited to hierarchically nested composite states) and the informally defined semantics.

UML EDOC : With the introduction of the Meta Object Facility (MOF) the OMG created the possibility to describe metalevel models in UML. In 2002 the UML Profile for Enterprise Distributed Object Computing (EDOC) was released which is a MOF compliant framework. It has an expressive graphical notation but the semantics for its life-cycles are limited and ambiguous.

XPDL: The Workflow Management Coalition (WfMC) has developed an XML-based Process Definition Language (XPDL) for supporting the interchange of process descriptions between different workflow systems. It is rich in modelling concepts but there is no graphical representation.

Web service languages: Most of the released web flow languages are based on XML and are built on the top of the Web Service Description Language (WSDL) by the W3C. WSDL describes only static interfaces does not have a graphical representation. The Business Process Execution Language for Web Services (BPEL4WS) has a graphical notation and combines the concepts of Microsoft's XLANG and IBM's WSFL. It allows the description of life cycles and provides a message correlation model. I actually

ebXML: ebXML offers a standard for describing business processes and for choreography of business transactions in business collaborations. It focuses on the exchange of business messages, collaboration agreements and collaboration profiles. These aspects are not mentioned in other related work [Tec01]. ebXML also uses UML for business process specification schemas. However, only the class diagrams are used and are not intended for the direct creation of ebXML business process specifications, i.e., there is no graphical notation except for the class diagrams. [Tea01].

4 A Metaclass Framework for Integration

Based on the underlying issues identified in the last chapters, we propose a meta-class based approach to behavior integration. Metamodelling is a generally accepted powerful tool for introducing semantic relationships into an object oriented model in fashion. Examples include [KS95], where explicit metaclasses were identified as a highly suitable tool for tailoring data models for specific applications by introducing particular application dependent semantic relationships. More recently, metamodelling has become a favored path to exploring UML extensions through stereotypes and constraints. These can now be described through the OMG MetaObject Facility (MOF) since it has been integrated into UML.

Moreover, a metaclass-oriented approach fits the nature of the object lifecycle view that we identified initially as our preferred angle for high level behavior descriptions.

4.1 Integration Levels

The approach views the integration process as affecting four modelling levels, progressing from the basic modeling infrastructure down to the individual instances.

1. Meta-Metaclass Level: At this level, a framework consisting of meta-metaclasses provides tools which are used to create metaclasses with particular properties, i.e., metaclasses that can be used to define groups of classes with similar behavior. The intent in this case would be to use this level to define the generic underlying behavior description and merging infrastructure: the definition of operators that would take multiple classes and integrate their behavior.

2. Metaclass Level: The metaclasses in this level instantiate the structure and behaviour of the meta-metaclasses. Existing behavior can be specialized and extended. At this level, the integration framework will be specified more precisely for a particular application domain, e.g. the insurance industry, libraries, or car manufacturing.

3. Class Level: On this level the classes instantiate the behaviour of the classes in level 2, i.e., a class defines an individual business process (or one of its constituents). Special data structures which are needed by the different applications of a domain are implemented in the classes.

4. Instance Level: On the last level the instance process takes place. The instances are created and combined with data, e.g. a RAA Car Insurance policy, a University library, or a Holden car.

The different system definition levels are mirrored by the sequence of integration stages and the different groups of persons involved in each stage.

Stage 1: Framework development - the definition of the Meta-Metaclasses and the operations they provide is, effectively, a research outcome. They build a framework for the level 2 and support it with objects and operators.

Stage 2: This is the highest level of pre-integration. It is the job of a system administrator to define metaclasses within the meta-metaclass framework. The behaviour and structure of these metaclasses are made for a special domain but should not be specialised on an application.

Stage 3: At the Class Level, the application developer is able to build his classes as instances of the meta-classes from level 3. The classes are extended for the special requirements of a particular application and so can only be used by this application.

Stage 4: The end user of the application creates instances of the classes built in level 3 and works with them in the application. At this level, no knowledge of the relations and behaviour of the objects is required; all that need be done is to provide the data that are stored in the object attributes.

4.2 Prerequisites

The idea is to choose typical standard business processes (such as order processing) that cross the boundaries of organisations or organisational sub-units and to select similar business processes in different organisation (or organisational sub-units) that could be unified.

We assume here that the component schemas to be integrated are already provided in the high level description format (e.g., in OBD), but are unenriched. In the same way as aggregation hierarchies are added during schema enrichment for data integration [GSSC95c, NS89], refinement hierarchies of activities and states [KS96, Sch90] can be added during schema enrichment for behaviour integration, yielding abstract activities and abstract states. (These can be utilised later to compare schemas at a higher level of abstraction.) We will define a set of relevant abstraction operators. (Note: abstract activities are just semantic abstractions and cannot be invoked such as abstract supertypes may not be instantiated.) In addition, it seems worthwhile to identify strategies to enrich unlabelled behaviour diagrams with extensions allowing the verification of the important formal properties of label preservation, unique label distribution, and common label distribution [SS02]. These properties enable an easy check for behaviour consistent specialisations [SS02] and we expect them also to simplify the testing of the inverse operation, generalisation, for consistency.

Since the component schemas have been developed independently, they may show semantic heterogeneities. Comprehensive taxonomies of these heterogeneities have been established for data integration for various data models [GSSC95b, KS91b, KCGS93]. A similar taxonomy for object behaviour is needed for behaviour integration. Such a taxonomy will be established by analysing to which dual behaviour heterogeneity each known data heterogeneity gives rise (e.g., in a structural conflict, an attribute "address" may correspond to two attributes "city" and "street"; similarly, an activity "move" might correspond to two activities "changeCity" and "changeStreet") by investigating our selected case studies, and by systematic consideration of the possible usages of OBD's modelling elements.

Elements of two component schemas may be correspond to each other according to various semantic relationships that require different integration strategies. Such correspondence relationships have been identified for data elements in [KS95, SN88] (e.g., identical, category-related, role-related, history-related) and elsewhere (see: [BE96, Con97]). Again, a corresponding taxonomy is needed for behavioural elements and will need to be developed as described above for semantic heterogeneities.

4.3 Performing Integration

Integration by generalisation is the approach commonly used in object-oriented integration methods [Con97]. More specifically, *discriminated generalisation* [GSSC95a] (which corresponds to *generalisation with object colouring* [KS95, SN88], suggests itself as the main integration operator to integrate comparable local object classes into a generalised, global object class.

Typical subcases of the integration problem that suggest themselves are: (a) for two classes that represent different real world objects/business processes (e.g. hotel reservations and car reservations; both being reservations); a situation which is typical for a company merger, (b) for local classes that represent different roles of possibly the same real world objects (e.g., an order in administrations and in manufacturing), (c) for two classes that represent the same real world objects but at different points in time (e.g., if there is a typical producer consumer relationship between two companies), and (d) for the mixed case.

The standard correctness criteria used in earlier work on object lifecycles, *observation consistency* (OC) and *invocation consistency* (IC), apply here as well. Informally, OC guarantees that any life cycle occurrence of an instance of a local object type is observable as correct processing according to the behaviour diagram of the global object type. And, IC guarantees that any start or completion of an activity on an instance of global object type can be translated into the start or completion of an activity on the corresponding local instance such that the local effect, when perceived at the global level, corresponds to the behaviour diagram of the global object type. Formal definitions with respect to LBDs and specialisation are given in [SS02]. Necessary extensions and modifications include (a) definition of IC and OC for activities with parameters, i.e., activities affecting several objects, (b) object specialisation (roles) [GSR96], where a root object is specialised into two objects as opposed to their current definition with type specialisation in mind [GS98] where extensions of two subtypes are usually disjoint, and (c) handling of inconsistency between corresponding local objects due to asynchrony caused by delayed updates in one of the local autonomous systems.

A highly desirable goal is to find necessary and sufficient conditions for OC and IC that can be checked (as far as possible) locally for a behaviour element (state or activity) when it is introduced in the generalised, global object type. This enables the incremental development of the behaviour diagram of a global object type. It would then be possible to define a set of integration operators which, once given a semantic relationship between comparable behaviour elements, integrates (if possible) these elements into the global object type according to the chosen consistency criterion (IC, or OC, or both). It is these operators that would be

Concurrency plays a role in this process as well. Due to delayed updates, corresponding local objects may be inconsistent. This is reflected in data integration by obsolete data values. In behaviour integration, the local objects will then reside in normally incompatible states (e.g., in states "toPay" and "paid"). We will identify conditions under which asynchronisms can be recognised and shielded at the global object type, if desired by the user (discriminated generalisation can always provide the full picture, if necessary). We call such inconsistencies "weak" as they will disappear once delayed activities have been performed in contrast to strong inconsistencies which are caused by performing conflicting activities (such as alternatives) in the local systems.

Although our initial studies and earlier work relies on OBDs due to their elegant semantics, we expect that (as in earlier work) the results will *mutatis mutandis* carry over to UML. The integration of state charts was already studied by Frank and Eder [FE99]. However, this work did

not examine the option of a "loose coupling" between different systems, and concurrency aspects such as the weak inconsistencies above, which might well arise in the case of, say, a web service based systems, were not covered, nor was the topic of database evolution.

The latter is a another as yet unexplored area that has to be addressed in the long term; when developing behavior-based integration approaches, ultimately one aspect of behavior that will have to be addressed is long-term behavior at the federated database level, if the local systems themselves evolve, or a new local system is added to the whole.

4.4 Tool support

It is widely accepted that database integration is an *engineering task* which cannot be fully automated. Schema enrichment as well as detection of semantic heterogeneities and correspondences between schema elements (or verification of automatically detected correspondences) requires human intervention. Similarly, different reasonable options to integrate comparable elements exist and it is up to the human user to choose the option most appropriate for the intended application (e.g. the choice between OC and IC).

To support the system integrator as far as possible, an obvious avenue is to investigate how semantic knowledge added during schema enrichment can be exploited to detect possible correspondences between schema elements. It is quite reasonable to expect that semantic abstractions of behavioural elements can guide this detection process in the same way that semantic abstractions of structural elements have been utilised in the past [GSSC95c]. Another concern is the exploration of design guidelines (similar to those established for conceptual database design) that assist the integrator in choosing between multiple meaningful integration choices, especially when semantic heterogeneities need to be resolved.

Since in practice detection of correspondences and integration are carried out iteratively and incrementally (especially if one follows the productive prototype approach and starts with a small federated database at first), a schema integration tool must support the definition of partially integrated schemas after some correspondences have been verified. In such cases the tool must keep track about integrated activities or states at which OC and IC have not yet been checked.

The current state of our tool support only addresses issues of representation or, more accurate, notation, and not yet integration. As part of the preparation for bridging the gap between our OBD based approaches and the UML world, we have developed a metamodel for an adapted version of UML activity diagrams that provides improved parallelism and other advantages, making the translation of results from OBDs easier [SH03]. Two masters theses deal with the translations between BPEL4WS models and activity diagrams [Rin03, Rei04]. Transformation rules specified in an OCL subset are used to define a mapping between MOF metamodels to define the BPEL4WS-UML mapping [Rei04]. The next suite of tools will build on these to provide analysis and integration capability.

5 Conclusion

Database and application integration remain as much of an urgent topic as they were ten years ago; in fact the problem area has arguably grown in importance. In particular, the rise of loosely coupled, distributed, dynamically arranged applications based on the Web Service paradigm means that the past focus on structural aspects grows less and less adequate. Tools that are intended to provide semantic support for integration tasks will need to deal with high level representations of behavior and object life cycles such as expressed through OBDs, UML activity diagrams, or web service coordination languages. A metaclass architecture is designed to permit tailoring of the scope of integration operators to the desired set of domains, with basic underlying operators defined at the highest level and available for all domains. Consistency rules between the behaviour diagrams of the object types to be merged will aid in the correct selection of operators, and tool support can provide consistency checking functions that greatly facilitate the job of the human user who makes the ultimate integration decisions.

References

[AS90] J. Larson A. Sheth. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[BE96] Omran A. Bukhres and Ahmed Elmagarmid. *Object-Oriented Multidatabase Systems: A Solution for Advanced Applications*. Prentice Hall, 1996.

[BPS97] P. Bichler, G. Preuner, and M. Schrefl. Workflow Transparency. In J.A. Pastor A. Olivé, editor, *Advanced Information Systems Engineering, Proc. of the 9th International Conference (CAiSE '97), Barcelona, Spain*, Lecture Notes in Computer Science, LNCS 1250, pages 423–436. Springer Verlag, June 1997.

[cac02] Special issue on enterprise application integration. *Communications of the ACM*, 45(10), October 2002.

[CEH+97] Stefan Conrad, Barry Eaglestone, Wilhelm Hasselbring, Mark Roantree, Felix Saltor, Martin Schonhoff, Markus Strassler, and Mark W. W. Vermeer. Research issues in federated database systems: Report of EFDBS '97 workshop. *SIGMOD Record*, 26(4):54–56, 1997.

[CHH+99] Stefan Conrad, Wilhelm Hasselbring, Uwe Hohenstein, Ralf-Detlef Kutsche, Mark Roantree, Gunter Saake, and Fèlix Saltor. Engineering federated information systems: Report of EFIS '99 workshop. *SIGMOD Record*, 28(3):9–11, 1999.

[Con97] Stefan Conrad. *Föderierte Datenbanksysteme. Konzepte der Datenintegration*. Springer Verlag, 1997.

[EKW92] D.W. Embley, B.D. Kurtz, and S.N. Woodfield. *Object-Oriented Systems Analysis: A Model-Driven Approach*. Prentice Hall, 1992.

[FE99] Heinz Frank and Johann Eder. Towards an automatic integration of statecharts. In *Proc. ER'99*, pages 430–444, 1999.

[Fra97] H. Frank. *View Integration für objektorientierte Datenbanken*. Reihe "Dissertationen zu Datenbanken und Informationssystemen", DISDBIS 32. Infix-Verlag, 1997. PhD Thesis.

[GFFK94] G. Gardarin, B. Finance, P. Fankhauser, and W. Klas. *IRO-DB : A Distributed System Federating Object and Relational Databases*, chapter 1. Prentice Hall, 1994.

[GS98] G. Gottlob and M. Schrefl. The Evolving Algebra Semantics of Class and Role Hierarchies. In B. Thalheim and L. Libkin, editors, *Semantics in Databases*, Lecture Notes in Computer Science, LNCS Vol. 1358, pages 92–113. Springer Verlag, 1998.

[GSR96] G. Gottlob, M. Schrefl, and B. Röck. Extending Object-Oriented Systems with Roles. *ACM Transactions on Information Systems*, 1996.

[GSSC95a] Manuel García-Solaco, Fèlix Saltor, and Malú Castellanos. A semantic-discriminated approach to integration of federated databases. In *Proc. COOPIS*, pages 19–31, 1995.

[GSSC95b] Manuel García-Solaco, Fèlix Saltor, and Malú Castellanos. Semantic heterogeneity in multidatabase systems. In Ahmed Elmagarmid and Omran Bukhres, editors, *Object-Oriented Multidatabase Systems*, pages 129–202. Prentice Hall, 1995.

[GSSC95c] Manuel García-Solaco, Fèlix Saltor, and Malú Castellanos. A structure based schema integration methodology. In Philip S. Yu and Arbee L. P. Chen, editors, *Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan*, pages 505–512. IEEE Computer Society, 1995.

[HvdHH+00] W. Hasselbring, W.-J. van den Heuvel, G. J. Houben, R.-D. Kutsche, B. Rieger, M. Roantree, and K. Subieta. Research and practice in federated information systems. *ACM SIGMOD Record*, 29(4):16–19, 2000.

[KCGS93] Won Kim, Injun Choi, Sunit Gala, and Mark Scheevel. On resolving schematic heterogeneity in multidatabase systems. *Distrib. Parallel Databases*, 1(3):251–279, 1993.

[KDN90] Manfred Kaul, Klaus Drosten, and Erich J. Neuhold. Viewsystem: Integrating heterogeneous information bases by object-oriented views. In *Proceedings of the Sixth International Conference on Data Engineering, February 5-9, 1990, Los Angeles, California, USA*, pages 2–10. IEEE Computer Society, 1990.

[KFM+95] W. Klas, P. Fankhauser, P. Muth, T. Rakow, and E. Neuhold. Database integration using the open objectoriented database system vodak. In Ahmed Elmagarmid and Omran Bukhres, editors, *Object-Oriented Multidatabase Systems*. Prentice Hall, 1995.

[KS91a] G. Kappel and M. Schrefl. Object/Behavior diagrams. In *Proc. 7th IEEE Int. Conf. on Data Engineering*, pages 530–539, 1991.

[KS91b] Won Kim and Jungyun Seo. Classifying schematic and data heterogeneity in multidatabase systems. *Computer*, 24(12):12–18, 1991.

[KS92] G. Kappel and M. Schrefl. Local Referential Integrity. In G. Pernul and A M. Tjoa, editors, *Proceedings of the 11th International Conference on Entity-Relationship Approach (ER'92)*, LNCS 645. Springer-Verlag, 1992.

[KS95] W. Klas and M. Schrefl. *Metaclasses and their Applications: Data Model Tailoring and Database Integration*. LNCS 943. Springer-Verlag, Berlin, Heidelberg, 1995.

[KS96] G. Kappel and M. Schrefl. *Objektorientierte Informationssysteme: Konzepte, Darstellungsmittel, Methoden*. Springer-Verlag, 1996. (in German).

[LOS97] P. Lang, W. Obermair, and M. Schrefl. Modeling Business Rules with Situation/Activation Diagrams. In P. Larson A. Gray, editor, *Proceedings of the 13th International Conference on Data Engineering (ICDE '97), Birmingham, U.K., April 7-11, 1997*, pages 455–464, 1997.

[NS89] E. J. Neuhold and M. Schrefl. Towards databases for knowledge representation. In J. W. Schmidt and C. Thanos, editors, *Foundations of Knowledge Base Management: Contributions from Logic, Databases, and Artificial Intelligence*, pages 241–257. Springer, Berlin, Heidelberg, 1989.

[OSS97] Andreas Oberweis, Roland Schaetzle, and Wolffried Stucky. INCOME/WF - a Petri net based approach to workflow management. *Wirtschaftsinformatik*, pages 557–580, 1997.

[PC99] G. Preuner and S. Conrad. View Integration of Object Life-cycles in Object-oriented Design. In J. Akoka, M. Bouzeghoub, I. Comyn-Wattiau, and E. Métais, editors, *Proceedings of the 18th International Conference on Conceptual Modeling (ER '99), Paris, France, November 15–18, 1999*, Lecture Notes in Computer Science, LNCS Vol. 1728, pages 413–429. Springer, 1999.

[PCS01] G. Preuner, S. Conrad, and M. Schrefl. View integration of behavior in object-oriented databases. *Data and Knowledge Engineering*, 36(2):153–183, February 2001.

[PS98a] Christine Parent and Stefano Spaccapietra. Issues and approaches of database integration. *Communications of the ACM*, 41(5es):166–178, 1998.

[PS98b] G. Preuner and M. Schrefl. Observation Consistent Integration of Business Processes. In Chris McDonald, editor, *Database Systems 98, Proceedings*

of the 9th Australasian Database Conference (ADC '98), Perth, Australia, February 2–3, 1998, Australian Computer Science Communications, Vol. 20, No. 2, pages 201–212. Springer Verlag, 1998.

[PS98c] G. Preuner and M. Schrefl. Observation Consistent Integration of Views of Object Life-Cycles. In Suzanne M. Embury et al., editor, *Advances in Databases, Proceedings of the 16th British National Conference on Databases (BNCOD 16), Cardiff, Wales, U.K., July 1998*, Lecture Notes in Computer Science, LNCS Vol. 1405, pages 32–48. Springer Verlag, 1998.

[PS02] Günter Preuner and Michael Schrefl. Behavior-consistent composition of business processes from internal and external services. In *Proceedings of the International Workshop on Conceptual Modeling Approaches for e-Business (eCOMO 2002)*, Lecture Notes in Computer Science. Springer, October 2002.

[RBPE91] J. Rumbaugh, M. Blaha, W. Premerlani, and F. Eddy. *Object-Oriented Modeling and Design*. Prentice Hall, 1991.

[Rei04] Thomas Reiter. Metamodel-based BPEL4WS-UML translation. Master's thesis, 2004. In progress.

[Rin03] David Rinner. Transformation of UML to WSDL/BPEL4WS. Master's thesis, 2003.

[RJB99] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley Publishing Company, 1999.

[Sch87] M. Schrefl. A Comparative Analysis of View Integration Methodologies. In R.R. Wagner, R. Traunmüller, and H.C. Mayr, editors, *Informationsbedarfsermittlung und -analyse für den Entwurf von Informationssystemen, Proc. Fachtagung EMISA, Linz, Österreich, 2.–3. Juli 1987*, pages 119–136. Springer Verlag, 1987.

[Sch90] M. Schrefl. Behavior modeling by stepwise refining behavior diagrams. In *Proc. ER'90*, pages 113–128, 1990.

[Sch98] I. Schmitt. *Schema Integration for the Design of Federated Databases*. Dissertationen zu Datenbanken und Informationssystemen, Vol. 43. infix-Verlag, Sankt Augustin, 1998.

[SH03] Michael Schrefl Stefan Haberl, Markus Stumptner. A metamodel for business collaborations based on uml activity diagrams. Technical report, University of South Australia, Advanced Computing Research Centre, 2003.

[SN88] Michael Schrefl and Erich J. Neuhold. Object class definition by generalization using upward inheritance. In *Proceedings of the Fourth International Conference on Data Engineering, February 1-5, 1988, Los Angeles, California, USA*, pages 4–13. IEEE Computer Society, 1988.

[SN90] M. Schrefl and E.J. Neuhold. A Knowledge-Based Approach to overcome Structural Differences in Object-Oriented Database Integration. In R.A. Meersman, Z. Shi, and C.-H. Kung, editors, *Artificial Intelligence in Databases and Information Systems (DS-3): Proc. of the IFIP TC2/TC8/WG 2.6/WG 8.1 Working Conference, PR China, July 4–8, 1988*, pages 265–304. Elsevier Science B.V. (North-Holland), 1990.

[SPSG00] Simon S. Y. Shim, Vishnu S. Pendyala, Meera Sundaram, and Jerry Z. Gao. Business-to-business E-commerce frameworks. *IEEE Computer*, 33(10):40–47, October 2000.

[SS00a] M. Schrefl and M. Stumptner. On the design of behavior consistent specializations of object life cycles in OBD and UML. In Mike Papazoglou, Stefano Spaccapietra, and Zahir Tari, editors, *Advances in Object-Oriented Data Modelling*. MIT Press, 2000.

[SS00b] M. Stumptner and M. Schrefl. Behavior consistent inheritance in UML. In *Proc. Intl. Entity-Relationship Conference (ER 2000)*, Salt Lake City, October 2000.

[SS02] M. Schrefl and M. Stumptner. Behavior consistent specialization of object life cycles. *ACM Transactions on Software Engineering and Methodology*, 11(1):92–148, 2002.

[Tea01] Business Process Team. ebxml business process specification schema. Technical report, ebXML.org, May 2001.

[Tec01] Technical Architecture Team. ebXML Technical Architecture specification. Technical report, ebXML.org, Febuary 2001.

[VA97] Mark W. W. Vermeer and Peter M. G. Apers. Behaviour specification in database interoperation. In *Conference on Advanced Information Systems Engineering*, pages 61–74, 1997.

[vdAvH02] Wil van der Aalst and Kees van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, January 2002.