

Systematic Change Impact Determination in Complex Object Database Schemata

Simon Lock, Awais Rashid, Peter Sawyer, Gerald Kotonya

Computing Department, Lancaster University, Lancaster LA1 4YR, UK
{lock, marash, sawyer, gerald} @comp.lancs.ac.uk

Abstract

Impact analysis is seen as an important technique for controlling the evolution of software systems. Numerous approaches have been proposed which aim to assess the impact of changes on many types of development artefacts. These techniques, although suitable for specific applications, contain inherent weaknesses, making them unsuitable for analysis of object database schemata. In this paper, we propose a novel hybrid technique, applicable to object database schemata, which overcomes many of these deficiencies.

1. Introduction

Like most database applications, object databases are subject to evolution. Evolution, however, is critical in object oriented databases since it is the very characteristic of complex applications for which they provide inherent support. These applications not only require dynamic modifications to the data residing within the database but also dynamic modifications to the way the data has been modelled (i.e. both the objects residing within the database and the *schema* of the database are subject to change). Furthermore, there is a requirement to keep track of the change in case it needs to be reverted. Object database schemata designed to fulfil the above set of requirements can become very large and complex. The large amount of information and complex relationships between the various entities in these schemata combine to make the process of assessing the effect of change expensive, time consuming and error-prone. However, without proper assessment, it is impossible for developers and maintainers to fully appreciate the extent and complexity of proposed changes. For maintainers this makes cost estimation, resource allocation and change feasibility study impractical. For developers, a lack of adequate impact analysis can lead to difficulties in ensuring that all affected entities are updated for

each change to the conceptual structure of the database.

Impact analysis has been employed to determine the extent and complexity of proposed changes during the various stages of the software life cycle [1,5,8,13,15,6,9]. Although many of these techniques have been suggested for analysing the impact of changes to object oriented design and code level artefacts, inherent deficiencies in such methods render them unsuitable for performing change impact analysis in an object database schema. In this paper we propose a novel hybrid technique which combines traditional impact analysis approaches with experience based capabilities in order to support change impact analysis in complex object database schemata.

The next section describes a complex schema model which is used to apply our impact analysis technique. The following section provides an overview of the impact analysis process, and describes current approaches to, and problems involved in, performing such analysis. This is followed by a description of our proposed hybrid technique which is illustrated by a worked example based on the schema model described in section 2. Section 4 contrasts our work with related work in this area while section 5 summarises and concludes the paper.

2. A Complex Schema Model

We have chosen the SADES (Semi-Autonomous Database Evolution System) [17] schema model we proposed in [18] to apply our impact analysis technique. The schema model is quite complex since it aims at providing support for:

1. Class hierarchy evolution
2. Class versioning
3. Object versioning
4. Knowledge-base/rule-base evolution

The conceptual schema in SADES is a fully connected directed acyclic graph (DAG) depicting the class hierarchy in the system. SADES schema DAG uses *Version derivation graphs* [10]. Each node in the DAG is a *class version derivation graph*. Each node in the class

version derivation graph (CVDG) has the following structure:

- Reference(s) to predecessor(s)
- Reference(s) to successor(s)
- Reference to the versioned class object
- Descriptive information about the class version such as creation time, creator's identification, etc.
- Reference(s) to super-class(es) version(s)
- Reference(s) to sub-class(es) version(s)
- A set of reference(s) to *object version derivation graph(s)*

Each node of a CVDG keeps a set of reference(s) to some object version derivation graph(s) (OVDG). An OVDG is similar to a CVDG and keeps information about various versions of an instance rather than a class. Each OVDG node has the following structure [10]:

- Reference(s) to predecessor(s)
- Reference(s) to successor(s)
- Reference to the versioned instance
- Descriptive information about the instance version such as creation time, creator's identification, etc.

Since an OVDG is generated for each instance associated with a class version, a set of OVDGs results when a class version has more than one instance associated with it. As a result a CVDG node keeps a set of references to all these OVDGs. Figure 1 shows the structure of a CVDG node with references to two OVDGs.

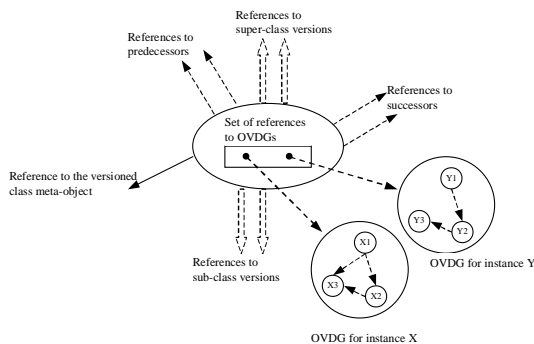


Figure 1: CVDG node with references to two OVDGs

The schema model has been implemented using the uniform dynamic relationships approach we proposed in [20]. [20] observes that three types of entities exist within an object oriented database. There are:

- Objects
- Meta-objects
- Meta-classes

Entities of each type reside within a specific *virtual space* within the database. Objects reside in the *object space* and are instances of classes. Classes together with defining scopes, class properties, class methods, etc. form the *meta-object space* [19]. Meta-objects are instances of meta-classes which constitute the *meta-class space*. Figure 2 identifies the various meta-classes in the system and the relationships in which their instances, the meta-objects, participate. The dashed block arrows indicate that all meta-classes inherit from the class *Root*. The line arrows, both solid and dashed, represent the various relationships meta-objects participate in. A solid arrow pointing from a meta-class back to itself indicates a recursive relationship. The dashed line arrows representing relationships among certain meta-classes and the class *Root* indicate that the relationship exists among instance(s) of the particular meta-class and instance(s) of a sub-class of the class *Root*. An instance of the class *OVDG Node*, for example, manages an object which is an instance of a sub-class of the class *Root* (since all classes inherit from the class *Root*). Similarly, a class *Property* or a method *Parameter* will normally have a sub-class of the class *Root* as its type.

The various relationships among instances of meta-classes shown in figure 2 can be dynamically modified to achieve dynamic schema modifications. It is possible to introduce new relationships. Schema changes are propagated to the instances using the *has-OVDG/is-head-of-OVDG-for* relationship that exists between a class version and its associated OVDGs.

From figures 1 and 2 it is obvious that a schema based on the SADES schema model or another model aiming at satisfying similar evolution requirements can become very large. The complex relationships between the various entities in such a schema will make the process of assessing the effect of change expensive, time consuming and error-prone. Effective impact analysis of changes in such large and complex schemata is, therefore, essential.

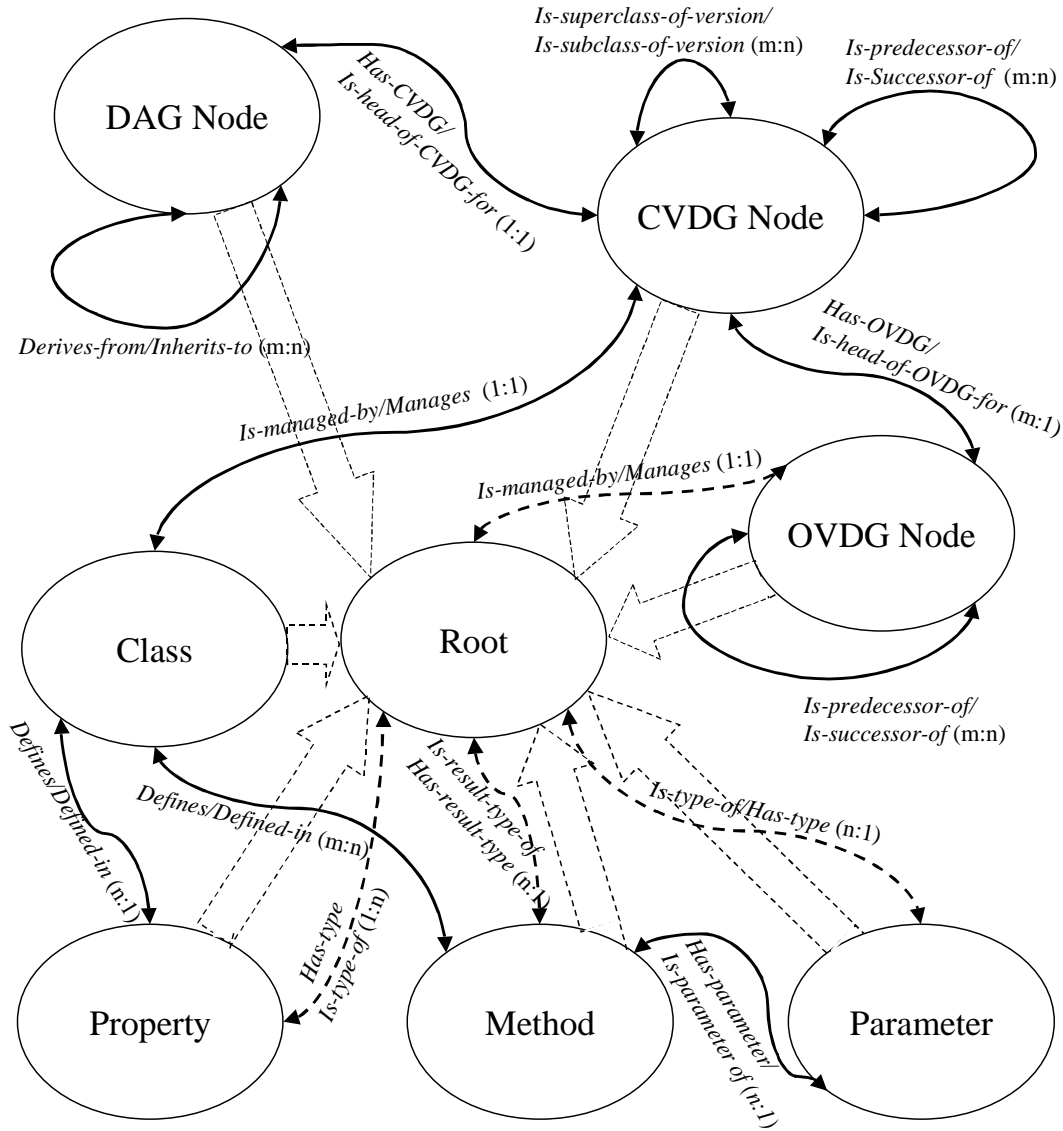


Figure 2: Meta-classes in SADES and the relationships their instances participate in

3. Impact Analysis

Many different approaches to the problem of impact analysis have been proposed. The majority of these techniques have emerged from system development and maintenance disciplines. Due to the nature of these sources, there emerges a concentration of impact analysis on software lifecycle artefacts, and especially design and code level components [1,5,7,11]. Many of these approaches may also exhibit a degree of applicability for assessing the impact of change on database schemata. In addition to this, the recent concentration on object orientation in software development has resulted in many techniques which support impact analysis under such a paradigm [1,8]. This is of

considerable relevance to the assessment of impact in object oriented databases due to the transferability of techniques based on such paradigms. These techniques are, however, not specific to object databases and, as will be discussed later in section 3.2.1, cannot by themselves provide effective impact analysis in object database schemata.

3.1. Impact Analysis Process

The impact analysis process described by Moreton [14] is a generalised model which is relevant for all elements which constitute software systems. The main stages of the impact analysis process are illustrated in Figure 3.

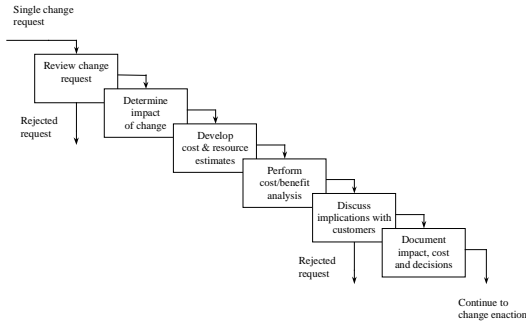


Figure 3: Impact analysis process

1. *Review change request:* This activity takes as input an initial evaluation of the change request. This information is then used to check the validity of the proposed change.
2. *Determine impact of change:* This involves the analysis of the target system to try to determine the impact of the change.
3. *Develop cost and resource estimates:* Estimation methods such as algorithmic cost modelling or static analysis are used to calculate the potential cost of the change.
4. *Perform cost-benefit analysis:* The estimated costs of the change are compared against the expected benefits to determine the cost effectiveness of the change.
5. *Discuss implications with customers:* The decision to accept or reject the changes must be made based on the relative cost and benefits. This is normally done in consultation with the customer organisation.
6. *Document impact, cost and decisions:* The potential effect and probable cost of the change on all artefacts as calculated in the previous stages must be recorded (even if a decision has been made to reject the change [12]).

Accurate impact determination is crucial for effective impact analysis. Because the information produced is often large and complex, care must be taken to ensure that the information generated is both relevant and adequate. A careful balance must be maintained between producing analysis results with too much information and not supplying enough to allow proper appreciation of change impact.

3.2. Impact Determination

The aim of impact determination is to predict the probable effect of a proposed change upon the artefacts which make up a system. The focus of our work is the assessment of the effect of changes on database schemata. Our approach uses traceability information extracted from

various sources to determine impact propagation paths between entities.

Traceability links indicate any association between two artefacts which has the potential to conduct an impact. In the case of object databases, all explicit and implicit relationships between the various entities residing in the database constitute traceability relationships.

Once all identifiable traceability links have been collected, they are composed into a single structure. Such a structure may then be used as a basis for direct investigation, or for the production of a visualisation of the effect of the proposed change [9]. Figure 4 shows the impact determination process for database schemata.



Figure 4 Impact determination Process

Traceability Extraction denotes the extraction of individual traceability relationships from the database schema model.

Traceability Analysis involves the processing of the raw traceability information extracted previously to produce a representation of the possible impact of a change on the database.

3.2.1 Current practices and their limitations

Existing impact analysis approaches can be classified into:

- Pre-recorded traceability approaches
- Dependency analysis techniques
- Knowledge based approaches

Pre-recorded traceability approaches are dependent upon the availability of accumulated details of relationships between the various components which make up a system. The traceability relationships required to perform impact analysis are available before analysis commences, albeit implicitly embedded in the recorded component relationships. Despite the attractive simplicity of pre-recorded traceability approaches, they can be difficult to employ due to the generally vague nature of the pre-recorded relationships [3]. This is due to the informal and loose definition of much of this observed traceability data which makes some important relationships hard to distinguish.

Dependency analysis techniques attempt to assess the impact of proposed changes by identifying the relationships which are present in semi-formal or structured representations of system component interaction. Such representations can include source code modules, design documents, finite state models, formal

specifications, and indeed object oriented database models [2]. Dependency analysis based methods extract traceability by analysing and identifying relationship links in the pre-existing system models. Working with dependency relationships can often be more efficient than with pre-recorded traceability relationships. This is because dependency relationships have more detailed information associated with them and are also more directional than traceability links. Despite providing detailed analysis for formalised information such as source code and semi-formal models, such techniques have little support for less concrete representations [3]. Furthermore, the utility of dependency analysis techniques to object oriented database impact analysis is limited since entity interaction forms only part of the overall database model.

The third class of impact analysis techniques, knowledge based approaches, extracts traceability information regarding system components by analysing the impact effects of previous change enactments. In order to achieve such evaluation, data concerning changes made to a system and their impact effects must be recorded. Each individual change and its consequences are recorded as a single change 'case' or change 'scenario'. When a significant number of such change cases have been identified it is then possible to identify traceability relationships within the system by analysing these cases. One limitation of knowledge based approaches is that during the learning stage, little traceability analysis is possible. This is because analysis using this approach requires a record of the causes and paths of database evolution. Thus, in order to utilise such a technique we must have available a relatively mature object database schema.

The above discussion illustrates the fact that each class of impact analysis technique is limited in scope. This means that no individual method is guaranteed to identify every single traceability link within a system. Furthermore, many techniques are over conservative. That is, each method will identify traceability links which do not necessarily imply an impact propagation

path. Much emphasis has been placed on assessing the impact of algorithmic design and code level artefacts. The applicability of these methods to object oriented databases is further limited by the abstract nature of schema models. As a result of this, the development of impact analysis techniques specific to object databases has been largely ignored.

3.2.2 Our Hybrid Approach

Our proposed solution is an integrated approach that combines the strengths of individual techniques. Such a technique can overcome the shortcomings of the individual methods, the whole being superior to the sum of the parts. We believe such an approach would provide a basis for better impact analysis of changes to database schemata by ensuring completeness through improved traceability coverage.

In addition to this, a degree of contrast can be identified between the potential impacts identified. This results from the fact that impacts identified by more than one analysis technique are more probable than those identified by a single technique.

Finally, the use of a composite approach provides a powerful and flexible technique which can help counteract for the abstractness of database entities.

The first step in impact determination is to extract traceability information from a number of different models of the database. These models of the database are as follows:

- Schema model - Used as a basis for performing pre-recorded analysis
- Object interaction model - Used as a basis for performing dependency analysis
- Past experience knowledge base - Used as a basis for performing knowledge based analysis

Our approach extracts the various implicit and explicit relationships which exist in these three models. The extracted traceability information is then processed to produce an impact propagation structure. A diagram showing the process used to generate this structure is illustrated in Figure 5.

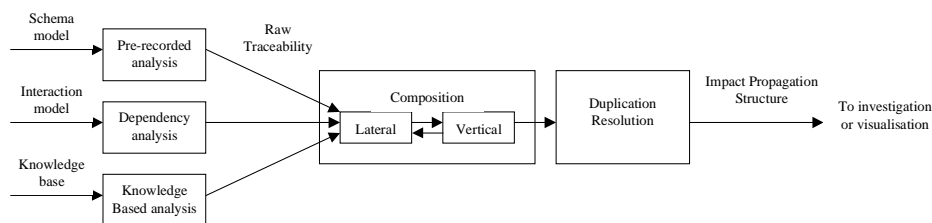


Figure 5: Propagation structure generation

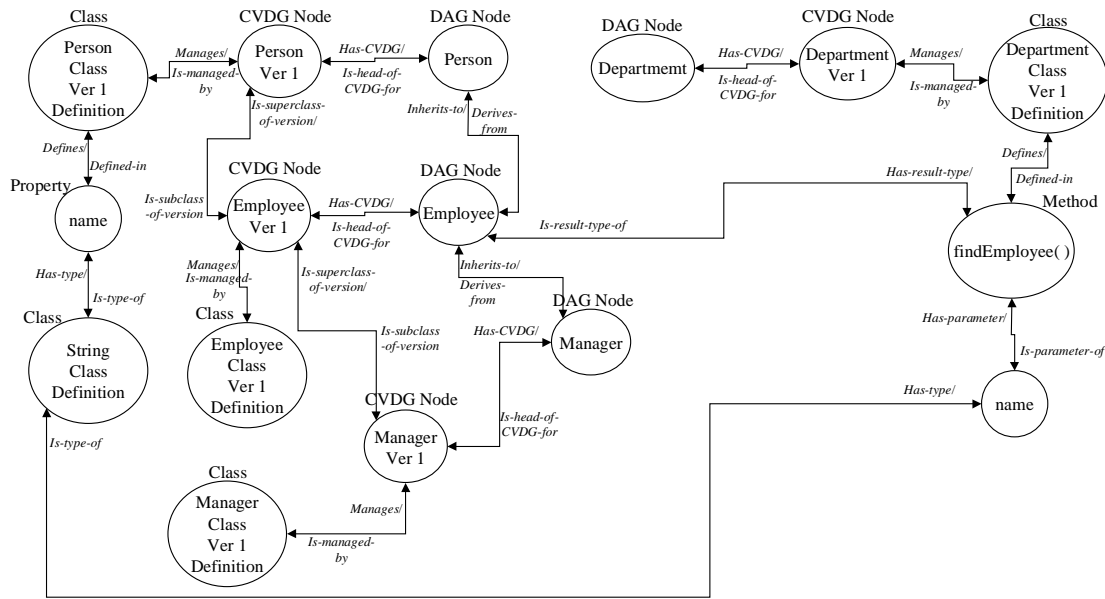


Figure 6: A SADES Schema model for example database

The generation of the impact propagation structure involves three distinct stages, the first two of which must be interleaved and performed repetitively. The three stages involved will now be described with the aid of the following example.

3.3. An Example

To help illustrate our hybrid approach, we now introduce a very simple example database upon which we will perform manual impact analysis for a proposed change. It is important to note that due to the complexity of most systems, impact analysis is only usually viable with the aid of automated support tools.

The example database has been designed to hold information about the business departments of a hypothetical company. The SADES schema model for our example database which is utilised in pre-recorded traceability analysis is illustrated in figure 6. To ensure the simplicity of the model, the versioning features of SADES have not been utilised and all classes are represented with single versions.

The interaction relationship information for this database which is utilised by dependency analysis is illustrated in figure 7.

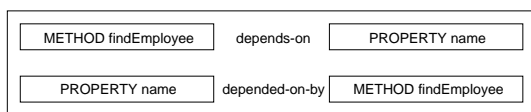


Figure 7: Interaction relationships

To illustrate the use of past experience knowledge as a basis for impact analysis, the effect of a previous change to the schema has been recorded. This change involved the proposal and inclusion of the *findEmployee()* method now present in the *Department (ver 1)* class. The effects of this change on the database are illustrated in figure 8.

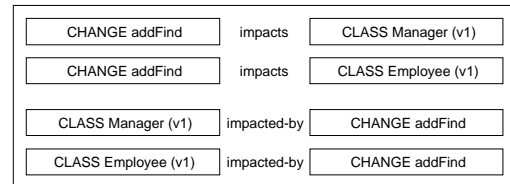


Figure 8: Knowledge base of past experience

With these three models in place we are able to use our hybrid technique to analyse the impact effect of proposed changes.

In our example application, it may become clear that the class representing the manager is no longer required in the database. The direct impact of this decision will be to make 'DAGN Manager' obsolete. Our aim is to analyse the total impact of this proposed change, using our hybrid approach.

3.3.1 Vertical composition

In order to identify the total potential impact of a change on a particular schema, the raw traceability data must be repeatedly analysed to

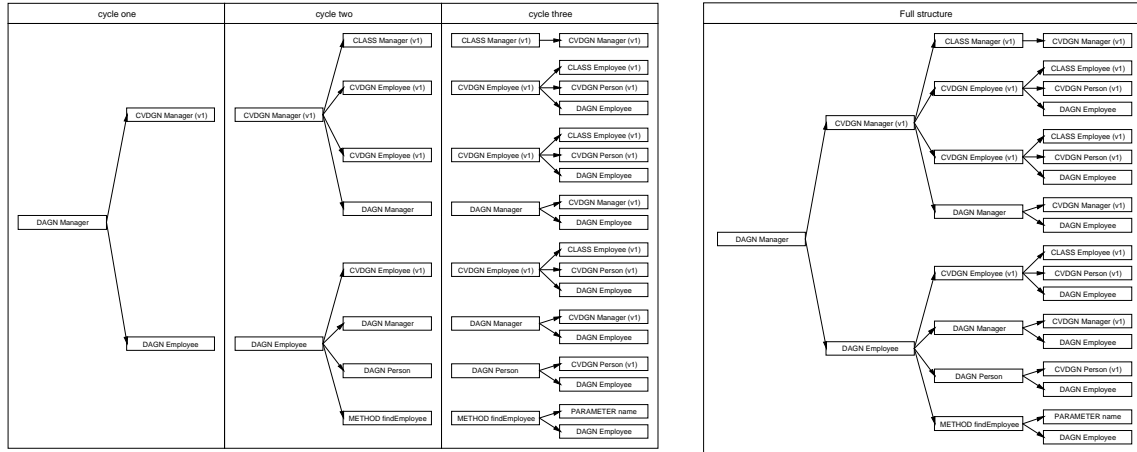


Figure 9: Vertical composition

identify every possible impact of that change. Each analysis cycle takes a set of impact sources (entities which must be altered to accommodate the change proposed) as input and produces a set of impact targets (entities which could be impacted as a result of the alterations made to the sources) as outputs. Repeated analysis is required because the targets of the impacts can propagate the change impact to other entities and must therefore be fed into the next analysis cycle as impact sources [21]. This is illustrated in Figure 9 which shows the vertical composition involved in the pre-recorded analysis of our example database. (N.B. for simplicity's sake, all impact analysis diagrams are limited to a depth of 3 cycles).

3.3.2 Lateral composition

When performing impact determination, our approach uses three complementary techniques for performing traceability extraction. The results of each of these individual techniques must be combined to produce a hybrid approach. This process is known as lateral composition and is achieved by summing all the impacts detected by the individual methods after each cycle of analysis. Lateral composition is illustrated in figure 10 which shows the composition of pre-recorded traceability, dependency analysis and knowledge based analysis of our example database.

3.3.3 Duplication resolution

It is possible for duplicate propagation paths to be identified between two entities. This results from the same impact being predicted by two different extraction mechanisms or multiple detection from the same mechanism. During

duplication resolution these are converted into single links. This is done to remove redundancy and thus minimise the size of the produced propagation structure. In our example the two *CVDGN Employee (v1)* impacts of *CVDGN Manager (v1)* in cycle two of the pre-recorded traceability analysis method require resolution.

3.3.4 Impact propagation structure

The impact propagation structure produced as a result of traceability analysis constitutes the end product of the impact determination process and shows all possible direct and indirect propagation paths for the particular change being analysed. It is important to note that this structure is not an absolute prediction of impact, but more a guide, indicating potential impacts. A database entity may be associated with any number of nodes within this structure. However, every node is unique because the impact which it represents is defined by the total path to that node.

For most non trivial databases, the size of the impact propagation structure produced can be quite significant. A number of mechanisms can be employed to help minimise the extent of this structure. These include filtering mechanisms such as:

- Focusing on a single view of the schema and excluding all propagations which do not affect that view
- The utilisation of a structural cut-off level to limit the depth of the impact structure
- The removal of recursive propagation paths from the structure

In addition to mechanisms which aim to reduce the physical size of the structure, it is also possible to ease the process of construction and

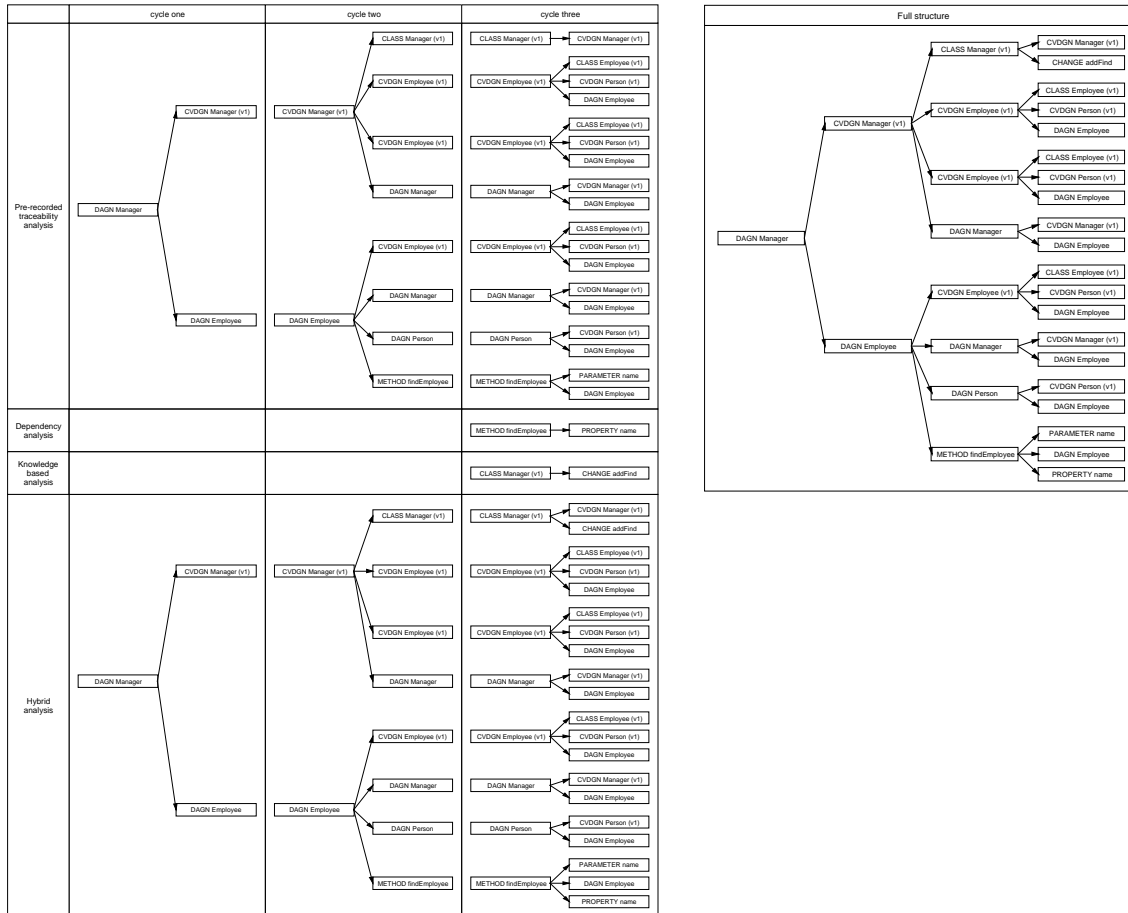


Figure 10: Lateral composition

investigation. This can be done by providing tool support for the impact analysis process. Such tool support can include facilities for collection and analysis of traceability, as well as the visualisation of the produced impact propagation structures. Without such tool support, impact analysis of most non trivial systems is not viable. The predicted impact effect of the proposed change on our example database is presented in figure 11. The illustrated structure has had all duplicate links resolved, all recursive paths removed and a cut-off level of three enforced. The impact propagation structure shown in figure 11 shows the total possible consequences of implementing the proposed change upon the database. This propagation structure is intended as a guide and may be used for:

- Aiding the general appreciation of change
- Producing graphical visualisations of change
- Assisting cost/benefit analysis
- Guiding implementation of change
- Directing post implementation testing
- Experimentation without implementation

4. Related work

Recently, there has been considerable interest in the pre-recorded traceability of development artefacts. Pinheiro and Goguen [16] have developed a tool for maintaining system traceability and classifying proposed changes to such systems for the purposes of performing impact analysis at the system requirement level. Han [6] also proposes a tool which can be used to predict the impact effect of proposed changes, but which is not limited to artefacts from a particular stage of development. Madhavji [12] has developed an abstract model to represent development artefacts which may be used for recording relationships and classifying changes respectively.

Much of the current work being done on dependency analysis concentrates on code level development artefacts. Such work includes a maintenance support system developed by Kuhn [7], Ada source code modification evaluation, test identification, and complexity/maintainability assessment tools by Loyall, Mathisen and

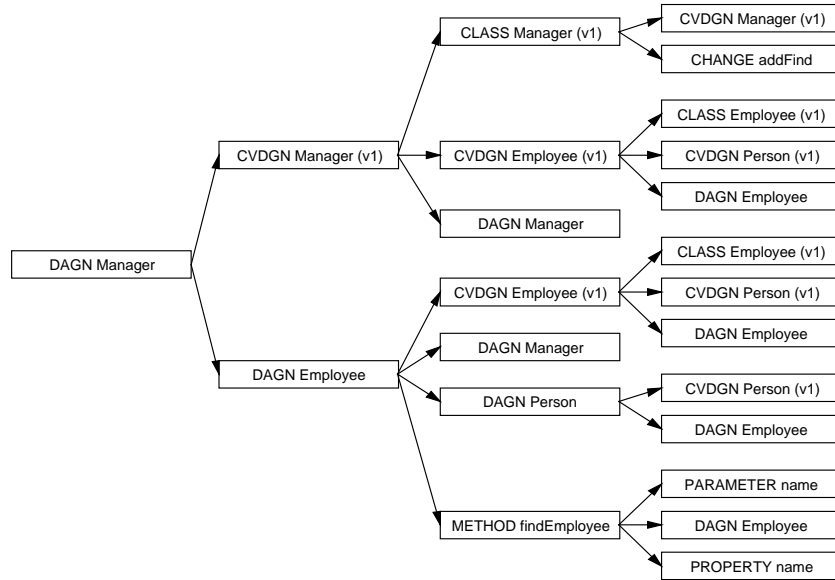


Figure 11: The total predicted impact effect of the proposed change

Satterthwaite [11] and automated C code analysis by Goradia [5]. The work done by Li and Offutt [8] offers an object oriented perspective on impact analysis of development artefact from the design stage onwards which is particularly relevant to our current area of investigation. Other related work has been done by Ajila [1] which includes tool support for the analysis of HOOD design models.

Limited research has been undertaken using knowledge based impact analysis, yet that which has been done shows considerable promise. Moriconi and Winkley [15] have proposed a rule based inference technique using component classification and component context elements. Additionally, Dhar and Jarke [4] describe a knowledge based dependency learning and prediction mechanism for the maintenance, acquisition, storage and retrieval of knowledge regarding system objects.

Despite the existence of these numerous approaches, directly applying such techniques to object databases is problematic. Not only do they all suffer from limited scope and difficulties with handling abstract information but, as we have seen earlier, each class is associate with type specific deficiencies.

Lock and Kotonya [9] have developed a hybrid impact determination tool and process which combines elements of the above techniques into one consistent approach. This process focuses on the prediction of change on requirements level artefacts, but is extensible to cover later stages of development.

Even such a hybrid approach cannot be directly applied to performing impact analysis of database schema without the significant alteration, customisation and optimisation for database specific issues.

5. Conclusions

In this paper we have presented an impact analysis approach for object oriented databases. The novelty of the work is in the hybrid nature of the impact analysis technique and its focus on object oriented databases. We have presented a classification for the current approaches to impact analysis and have highlighted the deficiencies of each. Our proposed hybrid impact analysis technique overcomes the limitations of these single paradigm methods by combining multiple analysis techniques. As a result it permits more controlled evolution of object database schemata. Such solid analysis is essential to support the ever increasing size and complexity of modern, feature rich object databases. We have also provided a brief description of the SADES database schema model and have used an example schema based on the model to describe the application of our approach. Future work includes automation of the process using a tool for analysing changes to object database schemata and generating visualisations of the possible impacts.

References

- [1] Ajila, S., “**Software Maintenance: An Approach to Impact Analysis of Objects Change**”, *Software-Practice and Experience*, 25(10) October 1995, pp1155-1181
- [2] Bohner, S. A., Arnold, R. S., Preface to “**Software Change Impact Analysis**”, Bohner, S. A., Arnold, R. S., Los Alamitos IEEE Computer Society, 1996, pp. ix-xii
- [3] Bohner, S. A., Arnold, R. S., “**An Introduction to Software Change Impact Analysis**”, *Software Change Impact Analysis*, Bohner, S. A., Arnold, R. S., Los Alamitos, IEEE Computer Society 1996 pp. 1-26
- [4] Dhar, V., Jarke, M., “**Dependency Directed Reasoning and Learning in Systems Maintenance Support**”, *IEEE Transactions on Software Engineering Vol. 14 No. 2*, Basili, V. R., Los Alamitos, IEEE Computer Society, February 1988, pp. 211-227
- [5] Goradia, T., “**Dynamic Impact Analysis: A cost-effective Technique to Enforce Error-propagation**”, *Proceedings of the 1993 International Symposium on Software Testing and Analysis*, pp171-181, 1993
- [6] Han, J., “**Supporting Impact Analysis and Change, Propagation in Software Engineering Environments**”, *Proceedings of the Eighth IEEE International Workshop on Software Technology and Engineering Practice*, pp172-182, 1997
- [7] Kuhn, D. R., “**A source code analyzer for maintenance**”, *Proceedings of the Conference on Software Maintenance*, Martin, R. J. (chair), Los Alamitos, C, IEEE Computer society, September 1987, pp. 176-180
- [8] Li, L., Offutt, A. J., “**Algorithmic Analysis of the Impact of Changes to Object-Oriented Software**”, *Proceedings of the International Conference on Software Maintenance*, pp171-184, November 1996
- [9] Lock, S., Kotonya, G., “**Requirement Level Change Management and Impact Analysis**”, technical report no. CSEG/21/98, Lancaster University Computing Department, October 1998
- [10] Loomis, M. E. S., “**Object Versioning**”, *Journal of Object Oriented Programming*, Jan. 1992, pp. 40-43
- [11] Loyall, J. P., Mathisen, S. A., Satterthwaite, C. P., “**Impact Analysis and Change Management for Avionics Software**”, *Proceedings of the National Aerospace and Electronics Conference*, New York, IEEE press, 1997, pp. 740-747
- [12] Madhavji, N. H., “**Environment Evolution: The Prism Model of Changes**”, *IEEE Transactions on Software Engineering*, 18(5), pp 380-392, May 1992
- [13] McCrickard, D. S., Abowd, G. D., “**Assessing the Impact of Changes at the Architectural level: A case study on Graphical Debuggers**”, *Proceedings of the International Conference on Software Maintenance*, pp59-67, November 1996
- [14] Moreton, R., “**A Process Model for Software Maintenance Software**”, *Change Impact Analysis*, pp 29-33, IEEE Computer Society, 1996
- [15] Moriconi, M., Winkley, T. C., “**Approximate reasoning about the effects of program changes**”, *IEEE Transactions on Software Engineering*, 16(9), pp980-992, September 1990
- [16] Pinheiro, F. A. C., Goguen, J. A., “**An Object Oriented Tool for Tracing Requirements**”, *IEEE Software*, Davis, A. M., New York, IEEE Computer Society, March 1996, pp. 53-61
- [17] Rashid, A., “**SADES - A Semi-Autonomous Database Evolution System**”, *Proceedings of the 8th International Workshop of Doctoral Students in Object Oriented Systems*, Jul. 1998, *ECOOP '98 Workshop Reader, Lecture Notes in Computer Science 1543*, pp. 24-25
- [18] Rashid, A., & Sawyer, P., “**Facilitating Virtual Representation of CAD Data through a Learning Based Approach to Conceptual Database Evolution Employing Direct Instance Sharing**”, *Proceedings of the 9th International Conference on Database and Expert Systems Applications*, Aug. 1998, *Lecture Notes in Computer Science 1460*, pp. 384-393
- [19] Rashid, A., & Sawyer, P., “**Evaluation for Evolution: How Well Commercial Systems Do**”, *proceedings of the First Workshop on Object Oriented Databases, held in conjunction with the 13th European Conference on Object Oriented Programming*, June 1999, Lisbon, Portugal
- [20] Rashid, A., & Sawyer, P., “**Dynamic Relationships in Object Oriented Databases: A Uniform Approach**”, *Accepted for the 10th International Conference on Database and Expert Systems Applications to be held in Florence, Italy*, Aug-Sept. 1999
- [21] Thompson, J., “**What you need to manage requirements**”, *IEEE Software*, pp 115-118, March 1994
- [22] Watkins, W., “**Why and how of Requirements tracing**”, *IEEE Software*, Card, D., New York, IEEE Computer Society, July 1994, pp. 104-106