# Towards Comprehensive Modelling by Inter-Model Links Using an Integrating Repository

Matthias Riebisch, Stephan Bode, Qurat-Ul-Ann Farooq, Steffen Lehnert
Ilmenau University of Technology; Ilmenau, Germany
{matthias.riebisch|stephan.bode|qurat-ul-ann.farooq|steffen.lehnert}@tu-ilmenau.de

*Abstract*—**Model-based development techniques enable a high efficiency and the mastering of complexity. For many domains more than one model has to be used to express the relevant information. Many methods use different models without a tight coupling, with a high risk of inconsistencies. Other approaches are based on metamodel extension or unified metamodels, with a limited tool support as consequence. We present an approach for the interconnection between several models in a joint repository by means of dependency relationships. The interconnection is shown between UML models, BPMN models and feature models by examples for variable workflows in mobile systems. The presented approach is implemented by the EMF-based repository EMFTrace, with XML for model representation. Dependencies are determined either automatically by a rule set or manually by explicit references.**

***Keywords-inter-model dependencies; model interconnection; repository; dependency; traceability; model-based development***

## I. INTRODUCTION

Software developers have to notice that requirements to software systems have become more and more complex over the years. One of the major challenges consists in the mastering of this complexity. Additionally, there is strong need for efficiency of software system development and evolution. Model-based development provides information on a higher level of abstraction, which helps master the complexity. Tool support for analysis, comprehension, decision-making as well as for the generation and validation of software solutions operates on the models and helps to satisfy the needs.

There are several modelling languages e.g. UML and BPMN to represent different aspects of a system. However, it is important to provide an integrated representation of these aspects to enable a comprehensive coverage for the developer. To fulfil the above-mentioned needs several possibilities can be applied.. (1) One option could be the establishment of new modelling languages to represent the necessary aspects. However it would hardly be possible to make it widely accepted and supported. (2) A much more promising approach is the extension of existing modelling languages, which requires extensible metamodels. Unfortunately, this demands for changes in the tool infrastructures and can lead to less support by tool vendors. (3) To integrate the different aspects of a system, an alternative approach is to represent the relations between models as a separate model and integrate it into the modelling environment, thus keeping the tool support.

In this paper we present an approach to integrate several models. The contribution of our work includes the definition of explicit relations implemented as links, the definition of link types as well as the definition of a rule set for the identification of these links. In our approach we focus on the utilization of well-established and standardized modelling languages, and use the concept of inter-model links to maintain relations between these models. We implement the concept by storing links in a repository external to the model-specific tools. In the repository the relevant elements of all related models are linked thus treating inter-model links and intra-model links in the same way. The links are determined and established by rules in order to achieve a high precision and recall. The approach is evaluated by an application in several case studies.

## II. MODEL INTERCONNECTION IN GENERAL

For software and system development, models are used to represent parts of the real world in the environment of a system, both for a description of the problem as well as of the solution. The representation is limited to the relevant aspects to master the complexity of the real world. Therefore, models provide information about some entities of the real world and about their relevant properties. According to the different uses of the models, several aspects are of interest. These aspects are usually represented by different models. For software development for example, UML provides a structural and a behavioural view with several models for each of them.

If the representation is restricted to a use of well-established modelling languages such as UML, Entity Relationship Models (ERM), and BPMN, there are several advantages. The well-established modelling languages are usually the best suited ones for the representation of the most common issues. They are widely accepted, and usually skilled personnel for their usage and tools for their support are available.

For their utilization, the relevant models have to be interconnected, which can be performed by links between their model elements [13]. For categorization we can distinguish between two types of relations:

- links between corresponding elements, which represent the same entity of the real world, and
- links between related elements, which represent related entities of the real world.

The model elements that have to be related are of a broad variety. All models to be included and their relevant

model elements contribute to the set of model elements. The set of link types to be considered firstly consists of all types of relations, which are represented by the models. Secondly, dependencies have to be represented explicitly because they are not – or not completely – expressed by some models. Thirdly, traceability links have to be represented because they constitute dependencies that have been traversed by the developer during engineering activities, and that have to be explored to understand the intentions of a solution. Furthermore, some real world entities can be represented in more than one model, because the scope of some models overlap. For example, an actor is represented in use case models of the UML as well as in many other models.

Metamodels describe available model elements, relations between them as well as constraints of the respective modelling language. For relations between elements of different models, the different metamodels have to be considered. A unified treatment of the relevant model elements is necessary to enable access to them. In our approach an additional, external repository is defined which covers all relevant model elements that have to be considered. Its metamodel represents the minimal set of definitions of the covered models. Figure 1 shows the external repository as the barrel below with its interconnection to other models managed by separate tools represented by boxes. Tool aspects are discussed later in section IV.

Regarding the implementation of relations between

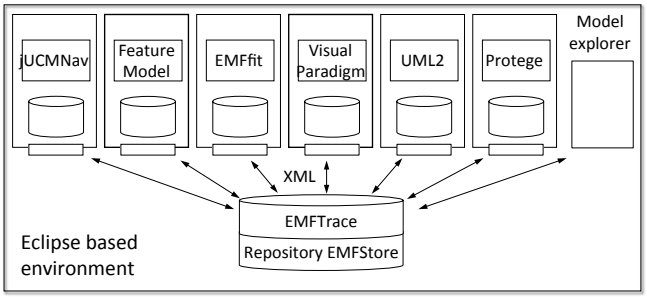


**Figure 1.** Model interconnection via external repository

model elements we have to distinguish between linking different elements within one model – which is usually performed by so-called intra-model relations available in the model's metamodel – and the so-called inter-model links between elements of different models. These relations can be established in two general ways.

Firstly, there is the way of establishing references between the elements of different models. The references can be formally defined in terms of syntax and semantics. They can be evaluated by tools and methods without ambiguity. Our approach follows this way.

Secondly, relations can be identified based on the names of identifiers of model elements. One option is the use of similarities between names as criteria, which can be determined by string comparison algorithms, as for example those of the field of information retrieval. The other option consists in the criterion of a relatedness of the concepts, which can be determined for example by an ontology. Such an approach is often used in the field of knowledge engineering. The claim of approaches of this type is, that every entity and relation can be modelled in a very flexible way. Many ontology-based approaches represent all relevant information by these means regardless of established modelling languages specialized for different aspects. However there is the drawback of limited rigour compared to usual modelling languages. For example there is no syntactic definition to distinguish between different types of model elements and relations. Due to ambiguity, these approaches enable a limited access for tools for analysis, validation, and utilization of the information.

Nevertheless, in our approach we make some use of ontologies, but limited to those aspects, which cannot be expressed by the existing modelling languages. We use ontologies similarly to glossary and thesaurus to represent relations with natural language expressions for example in requirements specifications, for which no formal syntax can be applied.

For cooperative work as required for practical projects, a versioning of models, model elements and links is required. For this purpose, a unified treatment of intra-model and inter-model links is necessary.

To fulfil the above-mentioned requirements, we have decided for an external repository, which covers all relevant model elements of the considered models, as well as their intra-model and the inter-model links. They are treated in the same way to provide a unified access. We are aware of the challenges related to this decision, such as cross-model consistency and synchronization of changes.

## III. MODEL INTERCONNECTION FOR WORKFLOW DOMAIN

For an effective discussion of the approach, concrete models and interconnection have to be considered instead of general ones. We present the approach for the development of highly flexible systems using workflow modelling for a decentralized application including mobile devices as processing platform.

### *A. Case study: workflow systems for mobile platform*

For illustration we present examples from a case study out of the workflow systems domain. The system supports service and maintenance personnel in the engineering industry by coordination and remote access using mobile devices. The system in its current state constitutes a prototype, which is developed in cooperation by industrial and academic partners. The development effort accounts for about 300 person months. The major challenges consist in scalability, flexibility, and platform independence. As platform technology, web services run on Apache ODE, XML based models including BPMN and BPEL, and specialized mobile devices, such as MoPad[1], are used. In the examples in this paper we show a cut-out for information acquisition via general-purpose and specialized mobile devices.

[1] http://www.tonfunk.de/en/produktlinien.html

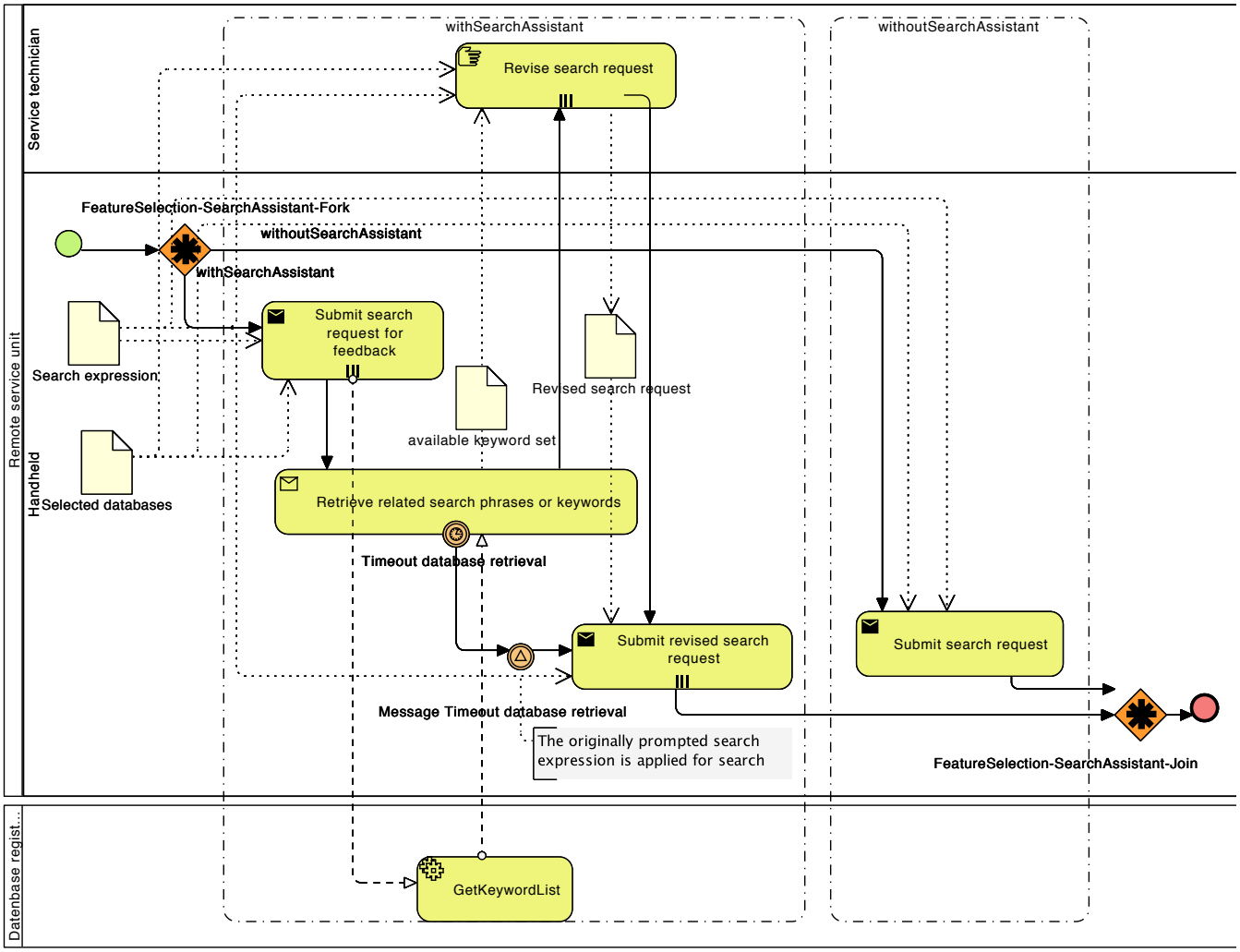


**Figure 2.** Business process diagram Information Acquisition with two variants modeled as groups

### *B. Requirements for extended modelling*

For the case study, BPMN workflow models have to be extended in multiple ways. Firstly, structural information has to be represented, such as the organizational structure of a service enterprise and of customer companies, system architectures of the technical systems to be maintained, and the information structure of the service knowledge. Instead of extending the workflow models it was decided, that this information is to be represented by related structural models using UML, for example by class and component diagrams. Secondly, variability of the workflow models has to be expressed, according to different variants of the system. Again, the decision was not to extend the workflow models but to establish references to feature models instead. Thirdly, terms and there relations are represented by an ontology in order to bridge between natural language documents, requirements specifications, and models.

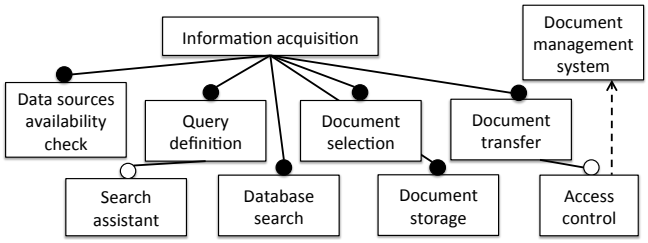


**Figure 3.** Feature model

### *C. Introducing Variability into BPMN Models*

The extension of BPMN models by variability is used here to give another example of model interconnection – between BPMN process diagrams and feature models. The extension of BPMN by variability is not discussed in detail because it is out of the scope of this paper. To represent variability for different variants of the system, a lightweight approach for workflow variability is applied. Variable BPMN modules are defined by a model element *group*. A variant out of a set of mutually exchangeable variants is defined as a *group* with equivalent interfaces – in terms of both control flow and message flow – within the set. In Figure 2 there are two groups for the variants *withSearchAssistant* and *withoutSearchAssistant* (framed

**Table 1**. Dependencies between BPMN process diagram and UML class diagram (partly)

| BPMN process diagram element | LinkType | UML class diagram element |
|---|---|---|
| MessageFlow | <use> | Class [by Message.ItemDefinition.reference] Property |
| ServiceTask | <isDefinedBy> | Operation [Message corresponds to an Operation, which corresponds to a ServiceTask] |
| SendTask and ReceiveTask | <use> | Class [Message.ItemDefinition.refrence] |
| DataObject | <instanceOf> | Class [by ItemAwareElement.ItemDefinition.Reference] |
| DataObject(isCollection=true) | <instanceOf> | Class {where the class or any of it parent classes have an association with the attribite isComposite=true} |
| DataStore | <isDefinedby> | Class |
| Process, Activities and Events | <defines> | Class [by Property (is a Container of data for the flow elements(processes, activities and events) and it is not visible in a process)] |
| DataInput | <instanceOf> | Class/Property |
| DataOutput | <instanceOf> | Class/Property |
| Property | <instanceOf> | Class |
| CallActivity | <uses> | Operation defined in a class |
| Process | <refines> | Operation |
| DataAssociation | <defines> <transforms> | Class/Property |

with a dashed-and-dotted line). A model element *gateway* constitutes the so-called hot spot that controls the instantiation of one variant out of the set. The gateway is designated by a link to the feature *Search assistant* to show that it controls the instantiation instead of the control flow. Figure 3 shows the corresponding part of the feature model, with *Search assistant* as a variable feature.

In addition to the behavioural aspects, structural aspects have to be modelled to represent data structure, system architecture and system environment. Figure 4 shows a cutout from a UML class diagram as an example.

### D. Model interconnection concept of the case study

For the case study, links between elements of BPMN process models, UML class diagrams, feature models and OWL ontologies are considered. A cut-out is presented in this paper. Further UML models such as component diagram and deployment diagram are omitted due to space reasons. Between this set of model elements there are relations of several types, both inter-model and intra-model links. Table 1 shows some of the inter-model links between BPMN process model elements and UML class diagram elements with their link types. There are several further link types such as part-of, abstracts, realizes, instantiates, interprets, substitutes, binds, simulates, verifies, tests, asserts, requires, imports, includes, invokes, calls, replaces, and conforms, which are of interest for different validation and utilization activities. For example, the type of a relation is used to determine the change impact between model elements.

For the support of various development activities, the links are staffed with attributes. For traceability links, design decisions can be represented by attributes that refer to goal, design alternatives and choice. For dependencies, a design rule or principle is referenced by another attribute. For a support of reverse engineering activities, links have a confidence factor.

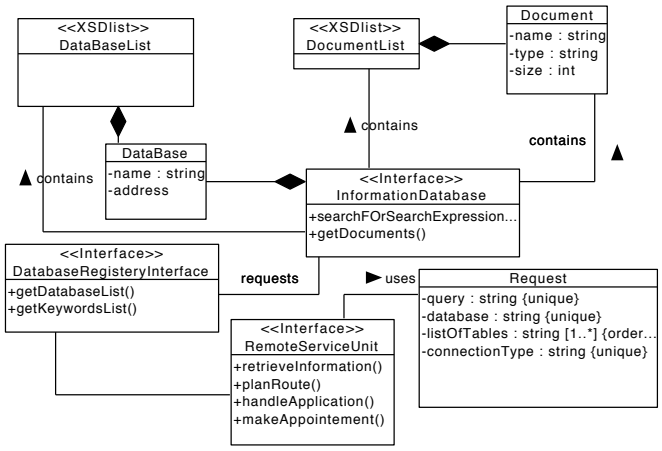


**Figure 4.** Class diagram with structural models (partly)

links, rules can be established. Rules for intra-model links can be defined based on the respective metamodel. Rules for inter-model links are defined for corresponding elements from different models (see Table 1 for examples) and by definitions of design methods, for example between an message flow in a BPMN process diagram and a UML class representing a data structure defined for this message. In the next section an example for the XML implementation of a rule regarding a relation to an ontology term is given (Figure 5).

## IV. Implementation of the Concept for the Case Study

### A. Tool Infrastructure

To implement our concept of model interconnection via inter-model links we developed a tool called EMFTrace, which enables traceability links between model elements [11]. EMFTrace is based on the EMFStore[2]

[2] https://code.google.com/p/unicase/wiki/EMFStoreNavigation

```
<TraceRule RuleID="TraceRule61" Description="Find similarities between UML-Components and OWL-Classes">
    <Elements Type="Class" Alias="e1"/>
    <Elements Type="Component" Alias="e2"/>
    <Conditions Type="And">
        <BaseConditions Type="NotNull" Source="e2::umlID"/>
        <LogicConditions Type="Or">
            <BaseConditions Type="Contains" Source="e1::IRI" Target="e2::name"/>
            <BaseConditions Type="Contains" Source="e1::abbreviatedIRI" Target="e2::name"/>
        </LogicConditions>
    </Conditions>
    <Actions ActionType="CreateLink" LinkType="Overlap" LinkSource="e1" LinkTarget="e2"/>
</TraceRule>
```

**Figure 5.** XML Rule for links between corresponding elements UML component and OWL class [11]

model repository [10]. It supports storing and versioning of Eclipse Modeling Framework (EMF)[3]-based models as well as their import and export based on XML technology. With EMFStore Ecore models[4] can easily be integrated in the repository via code generation.

EMFTrace provides an access layer to EMFStore for the flexible interconnection with external CASE tools. Figure 1 in section II shows these interconnections by arrows. For the interfaces between EMFTrace and the other CASE tools XML technology with XSL transformations is used. EMFTrace constitutes a platform for several CASE tools: (a) the Eclipse UML2Tools for UML[5], (b) Visual Paradigm[6] for UML and BPMN, (c) the Eclipse-based jUCMNav[7] for modelling requirements with the User Requirements Notation (URN) [8], (d) Protégé[8] for ontologies modelled with OWL, and (e) EMFfit an own tool implementation supporting architectural analysis according to [7]. Additional CASE tools, such as a toolbox with knowledge for system development or validation tools for traceability link evaluation, can easily be integrated. EMFTrace just has to be extended by additional EMF-based metamodels in the EMFStore repository and by additional XSL transformations.

For the actual dependency identification EMFTrace uses a rule-based traceability approach. Therefore, we integrated additional metamodels into EMFStore to maintain and store traceability links and rules directly in the repository. The traceability metamodel provides explicit support for modelling a hierarchy of dependency types. The types are stored in specialized catalogues inside EMFStore and hence implicitly are subject of model versioning. The metamodel for traceability rules offers a similar concept of rule catalogues to manage a set of rules efficiently and is based on similar approaches as proposed in [18] and [19]. However, we combined our rule-based concept with information retrieval techniques to improve dependency detection. For applying the rules to find model relations and represent them as traceability links, EMFTrace contains a rule engine.

[3] http://www.eclipse.org/modeling/emf/
[4] http://www.eclipse.org/modeling/emf/?project=emf
[5] http://www.eclipse.org/modeling/mdt/?project=uml2
[6] http://www.visual-paradigm.com/
[7] http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/
[8] http://protege.stanford.edu/

### B. *Model interconnection implementation*

For each of the model types BPMN, UML, OWL and feature model a metamodel is implemented with Ecore to be compatible with the Eclipse Modeling Framework and with EMFStore. In this way editor code for elementary model manipulation can be generated. Using EMFStore the models can be stored and versioned, and an XML representation can be derived as well. To integrate concrete model instances of the different types, e.g. a BPMN model from Visual Paradigm, the XML representation of the external tools has to be transformed into EMFStore's XML format using XSLT templates. The import and export procedure using XSLT has been encapsulated in EMFTrace functionality.

All relevant parts of the metamodels of the different modelling languages are covered by the Ecore model in EMFStore and EMFTrace. For UML and BPMN, all model elements of the valid standard are covered. However, the XML export of different CASE tools contains additional information that is kept separately instead of imported to the repository.

Using the features of EMFTrace the different models can be integrated into one model repository to establish dependency links by applying rules. Some dependency links for the diagrams of the example are shown in Table 1. Further examples are

- Gateway *FeatureSelection-SearchAssistant-Fork* (BPMN process model) <realizes> Feature *Search assistant* (Feature model)
- Component *RemoteServiceHandheld* (UML component model) <realizes> Pool *Remote service unit* (BPMN process model)
- Term *Role-based access control* (Ontology) <defines> Use Case *Assign rights* (UML model) <realizes> Feature *Access control* (Feature Model)

### C. *Link determination and discovery*

Import interfaces of EMFTrace provide the creation of intra-model links within the repository. For the establishment of inter-model links, rules are applied. There are rules for the identification of corresponding model elements as well as rules for the discovery of inter-model dependencies.

Following the general design proposed in [18] and [19], our rules for dependency discovery and link determi-

nation comprise three parts as listed below, whereas each part is responsible for a certain task:

*Element definition*: What models are effected by the rule?
*Query:* How are models related?
*Result definition:* What should be done with related models?

The rules can be edited in the repository or created with an XML editor. Figure 5 shows an example rule for the corresponding elements UML component and OWL class, which results in a traceability link of the type *overlap*.

### D. Link exploitation: examples for utilization

The links constitute an important part of the information represented by the models. Due to our concept of an identical treatment of intra-model and inter-model links, their exploitation does not demand any changes of methods and algorithms. Moreover, the extended set of links enables more ways, a greater depth, and a more comprehensive scope of their exploitation. Here some examples:

**Variability.** For a family of products with variability regarding workflows, the development of a new product demands an instantiation of the workflow model. This can be performed automatically by a configuration if the variable groups according to the selected features. For our case study, the definition of a product without the feature *Search assistant* (Figure 3) would lead to a configuration of the process model (Figure 2) with the group *withoutSearchAssistant.* The configured workflow model can be transformed to a BPEL model and can be executed without manual effort.

**Workflow execution.** A linkage between workflow models and structural models by references enables a more sophisticated workflow control. For example, a condition regarding the organizational unit, in which a service engineer currently works, is used to control the selection of the queried databases for information acquisition (part of the case study, but not covered by Figure 2.

**Evaluation of design quality.** Traceability links can be used to calculate metrics for design quality criteria, such as modularity, separation of concerns and complexity [3].

**Deployment.** Architectural models representing the deployment view for example using UML deployment diagrams can be used to control the execution of BPEL fragments. References between a BPMN service task in a process diagram and a computing device in a UML deployment diagram can be used to control the configuration of a workflow system.

**Analysis and validation.** Relations between requirements specifications, requirements models such as UML use case diagrams, test specifications and similar can be evaluated to check coverage and consistency.

## V. Related Work

The approaches for model integration and multi view modelling [1] are strongly related to our work. Jossic et al. [9] presented a transformation-based approach to establish links between models. They defined an equivalence function to establish the links between models by matching the similarities of the models. In our approach we are also using rules to discover the traceability links, however, our rules not only rely on the notion of similarity but also consider structural information of the models and many different links types.

Eramo et al. [5] also used the concepts of multi view integration, however, they store the correspondences between several models as relations within models. This leads to model pollution and is not a feasible option for us.

Repositories typically support model-based development by providing a shared place for automated model management and versioning. As a basis for our approach we considered seven model repository projects. The main decisive criteria for their evaluation were maturity, supported features as synchronisation of changes, extensibility, usability, and available documentation. The CDO Model repository[9] and EMFStore were evaluated to be the most suitable repositories for our approach to link EMF-based models. Finally, EMFStore was chosen as the model repository providing the basis for EMFTrace.

Since we want to link models expressed with modelling languages as BPMN, UML, or OWL we need corresponding metamodels. These metamodels need to be represented as Ecore models to cooperate with the EMF-based repository. The Eclipse Model Development Tools (MDT) provide such an implementation of UML as an Ecore model with their UML2Tools project. For BPMN such an Ecore model is available as well.

A contrary approach to ours is the Unicase project[10] which is also based on EMFStore. Unicase relies on a general unified metamodel comprising several artefacts from different development phases such as requirements, use cases, UML models, or project planning schedules. In this way it inherently provides traceability support. However, it does not support the complete standardized UML metamodel and no well-established modelling tools. Instead, the approach forces the developer to rely on the one Unicase tool, which seems not to be applicable in practice. Unicase does not fit to the needs of our project because it does not support BPMN.

Most of the approaches on the interconnection of BPMN with other models in the literature deal with the dependencies of process models on other process models. Most of these dependencies are about order of activities in a process model, parallelism, alternatives and time dependencies. Grossman [5] and Sell [17] do not deal with the dependencies of process models on other requirement, structural design and data models. For the representation of dependencies, metamodel based and ontology based approaches are used [17].

Approaches for variability in workflow models such as BPMN are presented in several works, for example the ones reviewed in [15].

Furthermore, some approaches in the literature emphasize the need of considering relations of processes with

[9] http://wiki.eclipse.org/CDO
[10] http://unicase.org/

other software artefacts such as state machines to represent object life cycle and other organizational models [2][16]. However, the specific type of dependencies between several models and how they can be extracted is not focused.

The most prominent approaches on dependency discovery stem from requirements traceability and model-driven development [21]. They use traceability links to express dependencies with auxiliary information for tracking the development activities. The concept traceability links enriched with semantic information such as link types and attributes is most suitable to explicitly express model dependencies with additional information needed for further analyses.

Since model-driven approaches express dependencies between the models subject to transformations (as for example in [4] for BPMN and BPEL), they can be fully automated. However, they are limited to these transformations and depend on the expressiveness of the transformation languages.

Requirements traceability approaches utilize either rules or information retrieval techniques to create links. Information retrieval approaches as proposed in [12] or [14] base their links on similarities between identifiers of model elements and therefore are easy to adopt to new models and data. They provide good recall, however, they lack an appropriate precision. Moreover, the absence of additional dependency information limits their usability for dependency detection among models.

In contrast, rule-based approaches seem to be more promising for inter-model linkage. They require more work to be adapted to new models or data since new rules have to be defined, but their precision is well above results achieved through information retrieval. Therefore they result in more reliable links. Rule-based approaches as proposed by [18] and [19] are able to provide auxiliary information on dependencies, such as the type of the dependency specified by the rule. Well-defined dependency types are inevitable to identify semantically rich dependencies and classify them properly into several groups as for example proposed in[19]. However, neither of the approaches is concerned with tracing BPMN and related models.

## VI. Conclusion

In this paper we have shown, how separate models can be used to represent the different relevant aspects of the real world. As bridges between the models we have presented cross-model links, which are dependency relations by nature. Traceability links are covered by this concept because they are of the same type. The cross-model links are stored in an external, joint repository with all related models side-by-side. We have shown this concept for models of the workflow system domain, for the modelling languages BPMN, UML, feature model and OWL. For the determination and the establishment of the links we have applied rules both for the identification of corresponding elements in different models and for the identification of depending elements in different models. The major benefit of the rule-based determination compared to other approaches consists in high precision and recall values for the established links.

## VII. Future Work

The implementation of the concept is an ongoing work. Further modelling languages have to be included, such as for requirements and test specification. Furthermore, additional instruments for a more comprehensive evaluation of model quality by rules shall be developed. A classification of link types constitutes another issue relevant for the evaluation and utilization of the represented information. Based on them, other means for utilization are subject of ongoing research, for example for design decision support and for test case generation. Furthermore, the management of the modelled information in a distributed way leads to other challenges to modelling languages and tools: expressiveness, synchronization of changes, and analyses for cross-model consistency.

## VIII. Acknowledgement

The research presented in this paper was partly funded by the Federal State Thuringia and the European Regional Development Fund ERDF through the Thüringer Aufbaubank with project no. 2007 FE 9041.

## IX. References

[1] M. Barbero, MD. Del Fabro, and J. Bézivin, Traceability and Provenance Issues in Global Model Management. In: 3rd ECMDA-Traceability Workshop. 2007.

[2] M. Bhuiyan, M.M. Zahidul Islam, G. Koliadis, A. Krishna, and A. Ghose, Managing Business Process Risk Using Rich Organizational Models, Proc. Computer Software and Applications Conference, 2007. (COMPSAC-2007), pp. 509-520.

[3] R. Brcina, S. Bode, M. Riebisch: Optimisation Process for Maintaining Evolvability during Software Evolution. In: Proc. 16th IEEE Conf. Engineering of Computer Based Systems (ECBS2009), IEEE CPS, 2009, pp. 196-205.

[4] G. Doux, F.Jouault, and J. Bézivin, Transforming BPMN process models to BPEL process definitions with ATL. In GraBaTs 2009 : 5th International Workshop on Graph- Based Tools, 2009.

[5] R. Eramo, A. Pierantonio, J.R. Romero, and A. Vallecillo, Change Management in Multi-Viewpoint System Using ASP, Proc. of the 5th Int. Workshop on ODP for Enterprise Computing (EDOC 2008), pp. 433 - 440.

[6] G. Grossmann, M. Schrefl, and M. Stumptner, Modelling inter-process dependencies with high-level business process modelling languages. Proc. 5th Asia-Pacific Conf. Conceptual Modelling (APCCM '08), Australian Comp. Soc., 2008, pp. 89-102.

[7] C. Hofmeister, R. Nord, and D. Soni, Applied Software Architecture. Boston, MA, USA: Addison-Wesley, 2000.

[8] ITU-T, Recommendation ITU-T Z.151 User requirements notation (URN) – Language definition, ITU-T, Nov 2008.

[9] A. Jossic, M. Didonet JP. Del Fabro, Lerat, J. Bézivin, and F. Jouault, Model Integration with Model Weaving: a Case Study in System Architecture In: ICSEM'07 - International Conference on Systems Engineering and Modeling. Israel, March 2007

[10] M. Koegel, and J. Helming, EMFStore: a model repository for EMF models, in Proc. Int. Conf. on Software Engineering (ICSE'10). ACM, 2010, pp. 307–308.

[11] S. Lehnert, Software architectural design and implementation of a repository for cross-model traceability (in German), Diploma thesis, Ilmenau Univ. of Technology, 2010.

[12] A. D. Lucia, F. Fasano, R. Oliveto, and G. Tortora, Recovering traceability links in software artifact management systems using information retrieval methods, ACM Trans. Softw. Eng. Methodol., vol. 16, no. 4, Article 13, Sept. 2007.

[13] B. Mahr, Position Statement, Models in Software and Systems Development in ECEASST, 30, 2010. http://journal.ub.tu-berlin.de/index.php/eceasst/article/view/437

[14] A. Marcus and J. I. Maletic, Recovering documentation- to-source-code traceability links using latent semantic in- dexing, in Proc. Int. Conf. on Software Engineering (ICSE'03), IEEE, 2003, pp. 125–135.

[15] I. Montero Perez, A Methodology Fragment For Developing Families of Business Information Systems. Improving The Design of Business Families for Service Oriented Architectures, PhD, Univ. of Seville, 2009.

[16] K. Ryndina, J.M. Küster, and H. Gall, Consistency of Business Process Models and Object Life Cycles, Univ. Eindhoven, 2006.

[17] C. Sell, M. Winkler, T. Springer, and A. Schill, Two Dependency Modeling Approaches for Business Process Adaptation, in: D. Karagiannis and Z. Jin, Eds., Knowledge Science, Engineering and Management, Springer, 2009, pp. 418-429.

[18] G.Spanoudakis, A. d'AvilaGarces, and A. Zisman,Revising rules to capture requirements traceability relations: A machine learning approach, in Proc. Int. Conf. in Software Engineering and Knowledge Engineering (SEKE 2003). Knowledge Systems Institute, Skokie, 2003, pp. 570–577.

[19] G. Spanoudakis, A. Zisman, E. Perez-Minana, and P. Krause, Rule-based generation of requirements trace- ability relations, JSS, vol. 72, no. 2, pp. 105–127, 2004.

[20] G. Spanoudakis and A. Zisman, Software traceability: A roadmap, in Handbook of Software Engineering and Knowledge Engineering, C. S. K., Ed. River Edge, NJ: World Scientific Publishing, 2005, vol. III, pp. 395–428.

[21] S. Winkler and J. von Pilgrim, A survey of traceability in requirements engineering and model-driven development, Softw. and Syst. Model., vol. 9, no. 4, pp. 529–565, 2010.