Navigation Design Support Using Reusable Navigation Templates

Peter Barna, Geert-Jan Houben, Philippe Thiran, Ad Aerts, and Flavius Frasincar
Technische Universiteit Eindhoven
PO Box 513, NL-5600 MB Eindhoven, The Netherlands
{p.barna, g.j.houben, ph.thiran, a.t.m.aerts, f.frasincar}@tue.nl

Abstract

Reuse is a fundamental concept of software design. It has many aspects and can be applied at various levels of abstraction. In this paper we focus on the reuse of high level (design model) specifications of software components for the Web. Concretely, we discuss the reuse of (parts of) navigation models that can be deployed in different application domains based on different data sources: navigation templates. At the same time, such components should be specified in a way that allows easy deployment. In this paper we propose a solution to this apparent contradiction using a component specific conceptual model. By applying a mapping from this model to a concrete domain model, an automatic deployment of the navigation templates can be performed. The process of navigation template design and deployment (including the process of defining the mapping) is explained and demonstrated on two examples using the Hera design methodology.

1 Introduction and Related Work

One of the major concepts of software design is reuse of software artifacts applied at different levels of abstraction - from reuse of system requirements to reuse of software code, and at different levels of granularity - from reuse of software packages or whole applications through use of design patterns [4] to reuse of classes (concepts) organized in hierarchies. The benefits of reuse are obvious, few major include saving software development effort (avoiding redundant design), facilitating the maintenance of software systems, and making the design traceable and transparent.

Due to the specific nature of the Web extensions of traditional software design methods have been proposed for development of web applications. Practically all software design methods for the Web consider reuse, mostly at finer levels of granularity (libraries of generic classes/concepts). However, flawless and practical reuse of navigation structure specification (often referred to as ”Web Patterns” or ”Web Design Patterns”) seems to be still a challenging problem. Good overview of common web patterns is presented in [10]. The description can serve as handy guidelines for web designers. Existing software libraries offer a wide variety of useful generic primitives that can be (re)used during building of web applications, but they rarely contain larger navigation patterns. Since navigation models are usually closely coupled with concrete domains specified by Conceptual Models (CM), to achieve the domain portability is not an easy task.

Concrete existing methods for web design benefit from the reuse concept in various ways. WebML [3] specifies the navigation structure by means of different (predefined) types of units. The method allows easy and convincing composition of different units, but the (data) units are also associated with data they present or process. Object-oriented approaches like OO-H [5], UWE [7], or OOWS [9] show solid approach supporting object-oriented reuse techniques like class abstractions. The problem of domain portability of navigation models in object-oriented environment using the OOHDM method is discussed in [12]. Web design frameworks introduced there represent abstract navigation models or their parts that are isolated from concrete domains and can be instantiated to a concrete domains. The deployment process consists of deriving a concrete OOHDM model from a OOHDM-frame.

In this paper we propose a practical approach for design and deployment of reusable and (domain) portable navigation patterns called Navigation Templates (NT). The approach uses a mapping from a Template Conceptual Model (TCM) describing the structure of data used within a NT, to a concrete domain conceptual model (CM). It allows not only to relatively easily deploy a NT to a concrete domain, but it also facilitates the specification of possible data manipulations in NT. The mapping is a data (schema) integration model, and we can benefit from existing knowledge in this field of research. The process of deployment of such NT to a concrete domain can be automated by using a NT specification and an appropriate mapping to a concrete domain. We explain the basic concepts of the NT design and deployment on two examples using the Hera methodology.

Despite using a concrete method the proposed approach of mapping NT to concrete domains can be used for other methods.

Section 2 explains the requirements on NT and context of their usage. The core of the paper is Section 3 explaining the approach in details using the Hera methodology and two examples. Possible mapping (data integration) problems and solutions are also discussed here. The current work on software tools supporting the design and deployment of NT for Hera is briefly explained in Section 4 and the text is concluded by Section 5.

2 Navigation Templates Overview

Navigation Template (NT) is a specification of a part of navigation structure that can be reused for a class of (similar) applications. A NT defines a navigation structure and its basic business logic (functionality). A NT is in general a (non-trivial) building block of a web application that can be reused for many similar applications and even used multiple times in a single application in different context. An example can be a user selection device (virtual shopping basket) that is used in a single web application for online shops once as a classical shopping basket for customers, while that same NT is also used for the selection of product categories of the customers interest (facilitating product search).

A NT is a software component specification that:

- define navigation structure and its basic business logic (functionality),
- is domain data independent, and
- can be deployed as a software executable.

Every NT contains a simple conceptual data model, the Template Conceptual Model (TCM). This is a minimal model that describes the structure of information presented and processed by the NT. When a concrete NT is being deployed, a mapping from TCM to a concrete domain is defined. The structure of a concrete domain data is specified in a concrete Conceptual Model (CM).

Figure 1 sketches how NT can be deployed within a navigation model. Thick arrows represent hyperlinks (possibly) carrying parameters (the internal structure of NT do not reflect any real structure and is sketched only for illustration purposes). Thin arrows show the deployment process with transformation of a NT specification to a concrete (part of) navigation model based on CM. This transformation is automatic, but uses TCM to CM mapping. The situation in Figure 1 requires two mappings, since the same shopping basket NT is used for different data concepts (though within the same domain). Because of the NT specification and

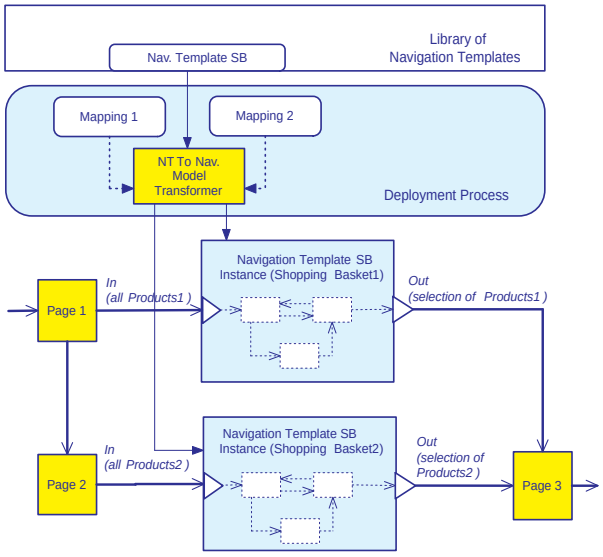

Figure 1. An example of a navigation model with deployed NT

deployment techniques may depend on concrete methodology/method used, in the following text we will explain principles of NT specification and deployment using a concrete (Hera) methodology and two examples. The first example will demonstrate multiple use of simple NT (guided tour) in a single application, and the second will highlight possible problems associated with mapping of TCM to a concrete domain and their solution.

3 Navigation Templates in Hera

For better explanation of NT we demonstrate its basic concepts using the Hera method and two examples. In the Hera design cycle, NT can be generated from a more abstract process model, or can be designed manually. The whole picture of the method and models is show in Figure 2. As aforementioned, a NT contains a TCM describing the structure of the information that is presented and processed by the NT, and contains an appropriate Template Application Model (TAM) that actually represents the navigation view over TCM. The *Articulations* (see [15]) represent the mapping from the TCM to a concrete CM, ”binding” the TCM to the concrete domain. The NT specification together with the *Articulations* is used by the *NT2AM Transformer* to generate a concrete (part of) AM describing the navigation structure and functionality of the concrete web application. The AM is then directly used by a Hera engine for online page generation for the web application. The architecture of a Hera system deploying NT is in Figure 2.

3.1 Brief Overview of Hera

Within the Hera project [6, 15] we investigate methods for specification of (dynamic) hypermedia presentations. The methodology determines a number of design steps resulting in a set of models. The conceptual design phase results in constructing a Conceptual Model (CM) defining the structure of source data used in presentations, the application design phase results in constructing an Application Model (AM) defining a navigation structure over the CM and eventual data manipulation associated with user actions, and the presentation design phase produces a Presentation Mode (PM) specifying the layout of presentations in a device-independent way. All models are in Hera expressed in RDFS [2].

These models are used by a Hera engine (a software module running as a servlet under a Web server) performing data retrieval (possibly from multiple different sources), and data transformations resulting in a presentation pages in different formats (Hera supports HTML, WML, and SMILE for presentations without data manipulations, HTML for presentations allowing forms and data manipulations). The bottom part of Figure 2 shows a Hera pipeline transformations, where retrieved data is transformed subsequently to a CM instance (CMI), AM instance (AMI), and a presentation in a concrete format (e.g. HTML). All intermediate data chunks are internally represented in RDF [8].

The Hera application modelling method is capable of expressing more advanced functionality than only a navigation view over a static data content. It supports modelling user inputs (by means of forms) and their processing (by means of data manipulation queries). Forms allow users to enter arbitrary information that can be used in data manipulation queries. Hera uses SeRQL [1] RDFS query language with slight modifications (queries are pre-processed by the Hera engine). The modifications include a default *session* namespace with a number of default parameters (e.g. *sliceid* for getting an instance of the root concept of the last constructed slice, *conceptid* for the name of the last constructed slice's root concept, etc.) that can be used in queries. Moreover new session parameters can be declared in AM (within *QueryResult* AM concepts). The *form* is a default namespace determining the form associated with a concrete query. Examples of queries are in Sections 3.3.2 and 3.3.4.

Examples of CM in a common graphical notation (with added multiple cardinality property expressed as a star along concept property arrows) are in Figures 4, 6, and 8. An AM contains basic building blocks called slices that describe the structure of navigation pages (or they parts since they can be nested), and their linking. Slices can have root concepts (from CM) notated as large ovals in the slice upper part. If a slice does not have the root concept, it is a constant

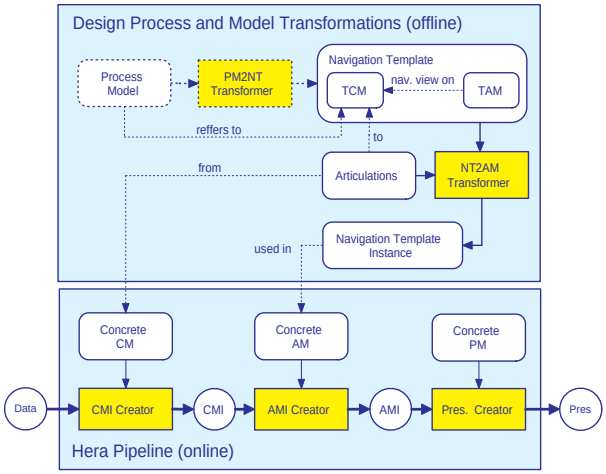

Figure 2. Architecture of a Hera system using NT

slice and can then have arbitrary content. If a target of a link is non-constant slice, the link then carries parametrization determining instantiation of the target slices (the anchor determines what instance of the target slice root concept is used for the target slice instantiation). A slice can contain attributes (literal properties in CM) from a root concept and attributes from concepts related to the root concept (bottom part of the slice shape) connected with the root concept by CM properties. Examples of AM are shown in Figures 5, 7, and 9.

3.2 Guided Tour Example

On the first example we demonstrate the multiple use of very simple NT (on purpose we chose here a very simple one to explain the principle). The NT represents a guided tour - step-by-step (one per web page) presentation of multiple concept instances. This concrete NT we deploy twice in a simple museum application. Once for the presentation of painters, and once for the presentation of paintings. For every concrete deployment we will show a set of articulations (mapping from TCM to CM).

Figure 3 shows the TCM and TAM of the guided tour NT. It uses a special type of slice (iterator) that allows easy implementation of guided tour. A slice of type *Iterator* comes with default *IteratorForm* providing a navigation facility through a collection of the *Item* instances and allowing to exit the iteration using the *Out* button. In the case of exit the instance of the last viewed *Item* is provided as the output parameter. The concrete data source containing information about painters and paintings is specified by its CM shown in Figure 4. Figure 5 contains an AM deploying the two instances of the Guided Tour NT. They contain attributes

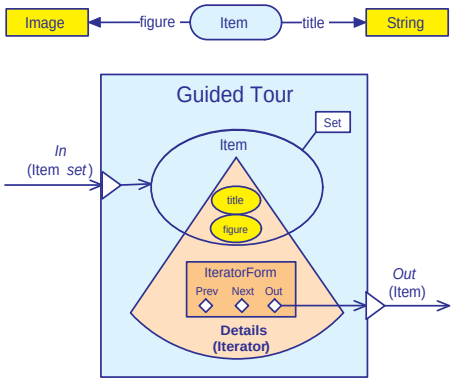

Figure 3. Guided Tour NT, TCM on top, and TAM on bottom

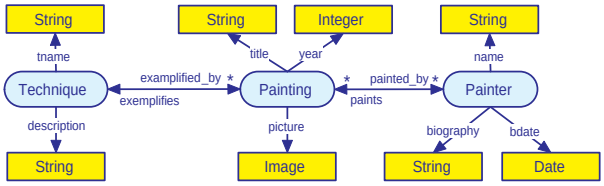

Figure 4. A concrete CM describing the structure of a data source containing paintings and painters information

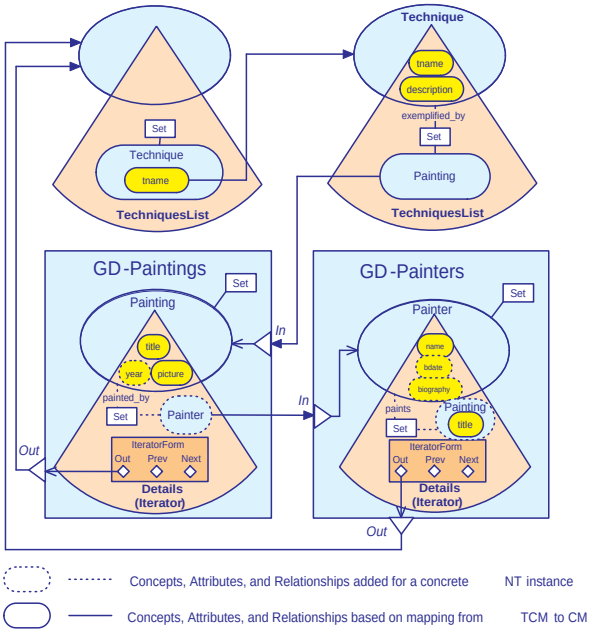

Figure 5. A concrete AM (paintings and painters) deploying instances of the Guided Tour NT

based on mapping from TCM to CM and also attributes not appearing in the original NT, but are added by the designer.

3.3 Publications Example

On this example we demonstrate possible problems with mapping from TCM to a concrete CM. This is basically a problem of data integration. We will show some typical cases appearing in RDFS schema integration and present some practical solutions covering the needs of our method (including rewriting of selection and data manipulation queries). Most of the problems are recognized and studied in [11, 13]. The example application is a publication database that stores publications, authors (researchers), and research groups. The Publications NT is specified in Hera and later deployed to existing data source with different data structure. A set of articulations is defined.

3.3.1 Template Conceptual Model

TCM contains only these concepts, concept properties, and literal properties that are necessary for describing the core navigation structure and functionality (we show only adding a publication) associated with a concrete NT. Figure 6 shows the CM.

3.3.2 Template Application Model

A TAM plays the role of navigation model and can contain specification of input forms and their processing. The Publication TAM is shown in Figure 7. The TAM consists of three slices presenting a (list of) groups (the *Group* slice), list of researchers in a particular group (the *Group.Researcher* slice), and a list of publications of a researcher (the *Researcher.Publications* slice). The *AddPaper* slice allows adding a publication created by a concrete (only one in our example) researcher. For the sake of simplicity the application is not complete, we omitted here more comprehensive data manipulations, like removing of publications, adding a new researcher, management of groups, etc. The *AddPaper* query is activated when the *AddPaper* form is submitted and is:

```
CONSTRUCT DISTINCT
   {P}rdf:type{tcm:Paper};
      tcm:ptitle(Title);
      tcm:url{URL};
      tcm:published_at{Published};
      tcm:author{Author};
      tcm:year{Year}
FROM
```

```
  {form:AddPaper}<form:Iptitle>{Title},
  {form:AddPaper}<form:Iurl>{URL},
  {form:AddPaper}<form:Ipub>{Published},
  {session:session}<session:resID>{Author},
  {form:AddPaper}<form:Iyear>{Year}
```

In this query a new instance of a paper is created where its properties are taken from the *AddPaper* form inputs. The value assigned to the *Author* variable is the URI of the root concept instance of the *Researcher.Publication* slice. The URI of the current researcher is set by the *SetResearcher* query that sets the value of the *resID* session variable. Note the *session* namespace indicating the session default or defined properties (of the unique *session* concept instance), and the *form* namespace indicating the form concept (instances) and its properties.

The *SetResearcher* query is very simple:

```
SELECT
  R
FROM
  {session:session}<session:sliceid>{R}
```

The *session:sliceid* is a default session variable containing the URI of the root concept instance of the last completely instantiated slice (thats why it is attached to the *AddPaper* slice and not to *Researcher.Publications* slice, although it contains an URI of the current *Researcher*). The value of the *R* variable is in the RDFS TAM specification assigned to *session:resID*. It is not visible from the TAM diagram, but together with the query definition it in RDFS file is:

```
<rdfs:Class rdf:ID="Query_ID101"
            slice:execute="Once">
   <rdfs:subClassOf rdf:resource=
      "http://wwwis.win.tue.nl/
      ~hera/ns/slice#Query"/>
   <slice:queryString>
      SELECT R
      FROM {session:session}
              <session:sliceid>{R}
   </slice:queryString>
</rdfs:Class>

<rdfs:Class rdf:ID="QueryResult_ID101"
            slice:resultName="resID"
            slice:useAsSessionVar="Yes">
   <rdfs:subClassOf rdf:resource=
      "http://wwwis.win.tue.nl/
      ~hera/ns/slice#QueryResult"/>
</rdfs:Class>
```

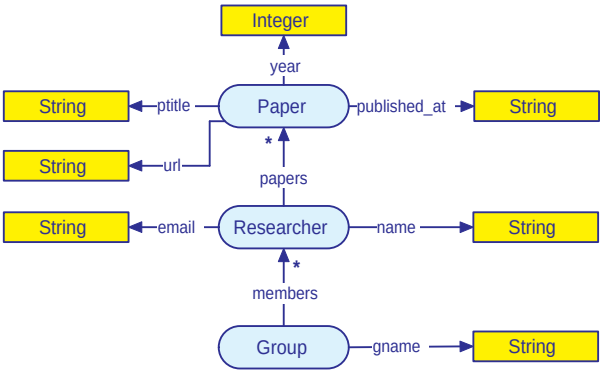

Figure 6. Template Conceptual Model

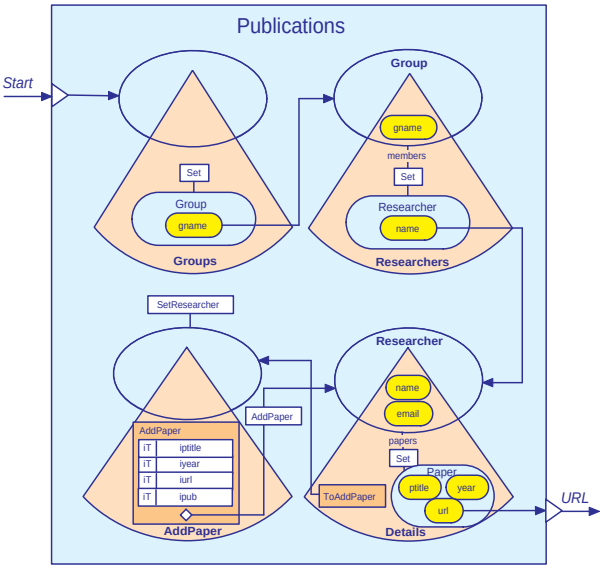

Figure 7. Template Application Model

3.3.3 Mapping NT to Concrete Domains

A necessary condition for automated transformation of a NT to AM for a concrete CM is existence of a mapping from

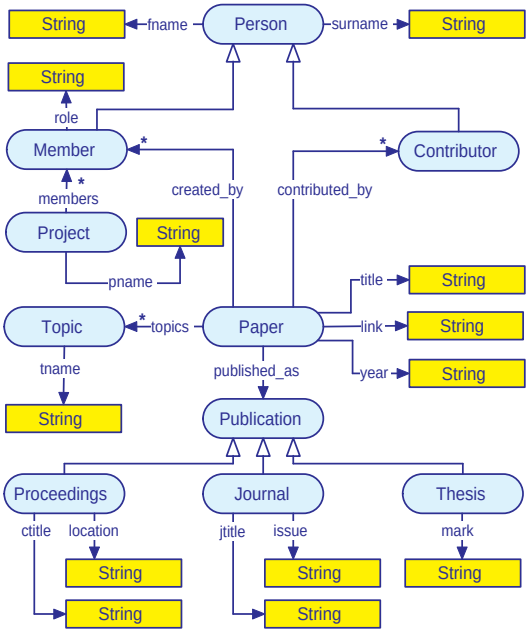

Figure 8. Concrete domain CM

the abstract CM to a concrete domain model. We demonstrate the specification of such mapping on an example and show (some of) possible situations and solutions. Figure 8 presents a concrete CM describing similar domain of the publications and researchers. This CM is slightly modified (for the purpose of highlighting of possible problems) version of the CM for a Publications application prototype running on the Hera engine. There are two categories of mapping. The first one is concept-to-concept that facilitates determination of root concepts and data manipulation queries during transformation of TAM slices into concrete AM slices, and the second is the attribute-to-attribute mapping that allows the transformation of slice attributes and is also used in query transformations.

We define now mappings of type concept-to-concept and attribute-to-attribute. We use path expressions specifying concept-properties chains in the form $\{Concept1\}property\{Concept2\}....$ Inverse properties are notated as $\{Concept\}property^{-1}$. If a value(s) of a TCM property is constructed from values of more properties in CM (concatenation) we write it $\{Concept1\}property1 \odot \{Concept2\}property2$. The fact that a value(s) of a TCM property is (are) retrieved possibly from more different CM properties is captured as $\{Concept1\}property1 \cup \{Concept2\}property2$. In this case the mapping is a union of values of given path expressions. In this example the mappings for concepts are:

- *tcm:Group* has no mapping (see mapping of its attribute further in the text)
- *tcm:Researcher* is mapped to *cm:Person*
- *tcm:Paper* is mapped to *cm:Paper*

For mapping of attributes we define articulations containing pairs of path expressions for TCM and corresponding CM path expressions, the main items (bullets) mean TCM path expressions, and subitems are corresponding CM path expressions:

- $\{Group\}gname$, the mapping does not exist. We will use constant mapping here
  - CM: a constant string
- $\{Researcher\}name$:
  - CM: $\{Person\}fname \odot \{Person\}surname$
- $\{Researcher\}email$, the mapping does not exist (the attribute is not present in the target domain model)
- $\{Researcher\}papers$:
  - CM: $\{Member\}created\_by^{-1} \cup \{Contributor\}contributed\_by^{-1}$
- $\{Paper\}ptitle$:
  - CM: $\{Paper\}ptitle$
- $\{Paper\}published\_at$:
  - CM: $\{Paper\}published\_as\{Proceedings\}ctitle \cup \{Paper\}published\_as\{Journal\}jtitle$
- $\{Paper\}year$:
  - CM: $\{Paper\}year$
- $\{Paper\}year$:
  - CM: $\{Paper\}link$
- $\{Paper\}url$:
  - CM: $\{Paper\}link$

Further details of mappings are explained in Section 3.4.

3.3.4 Deployed Navigation Template

The deployed AM (an AM generated from a NT using appropriate articulations) is shown in Figure 9. Some relationships semantically connecting different concepts and determining data instances to be presented are replaced by more complex queries, for instance the *PaperQuery* is a union of the two queries:

```
SELECT X
FROM {P}contributed_by{X}
```

and

```
SELECT X
FROM {P}created_by{X}
```

where P is the instance of the *Person* concept given by the *Perosn.Details* slice instance. The *AddPaper* query is transformed into two sets of queries:

```
CONSTRUCT DISTINCT
   {P}rdf:type{cm:Paper};
      cm:created_by{M}
FROM
   {session:session}<session:resID>{M}

CONSTRUCT DISTINCT
   {Proc}rdf:type{cm:Proceedings};
      cm:ctitle{Title};
      cm:link{URL};
      cm:year{Year};
      cm:ptitle{Published},
   {P}cm:published_at{Proc}
FROM
   {form:AddPaper}<form:Iptitle>{Title},
   {form:AddPaper}<form:Iurl>{URL},
   {form:AddPaper}<form:Ipub>{Published},
   {form:AddPaper}<form:Iyear>{Year}
```

and

```
CONSTRUCT DISTINCT
   {P}rdf:type{cm:Paper};
      cm:created_by{M}
FROM
   {session:session}<session:resID>{M}

CONSTRUCT DISTINCT
   {Proc}rdf:type{cm:Journal};
      cm:title{Title};
      cm:link{URL};
      cm:year{Year};
      cm:jtitle{Published},
   {P}cm:published_at{Proc}
FROM
   {form:AddPaper}<form:Iptitle>{Title},
   {form:AddPaper}<form:Iurl>{URL},
   {form:AddPaper}<form:Ipub>{Published},
   {form:AddPaper}<form:Iyear>{Year}
```

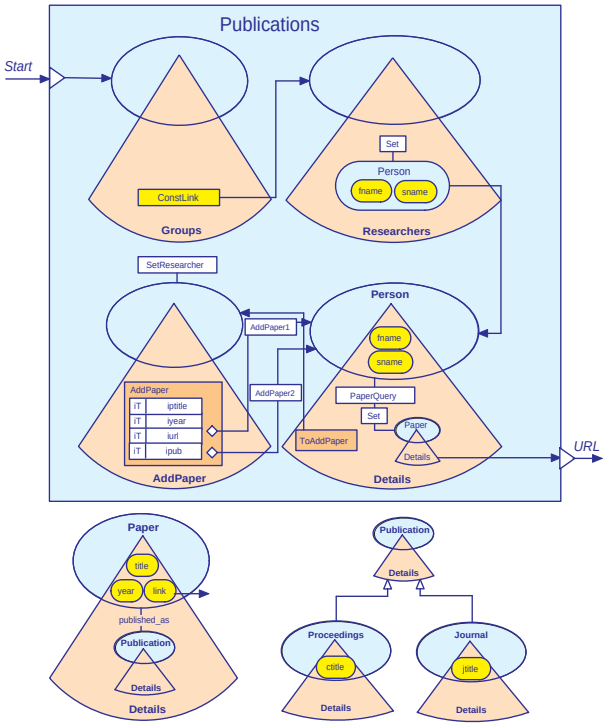

Figure 9. Deployed Publications NT

These two sets of queries are automatically generated (as well as two buttons in the *AddPaper* form - one for conference and one for journal papers) due to the different attribute structure for journals and conference papers. All models are simplified (and not entirely correct). We do not show all queries for adding papers for *Member* and *Contributor* types of authors (we show only for *Creator*). Really generated models would have two specializations of the the *Perosn.Details* and *AddPaper* slices.

3.4 Major Problems in TCM to CM Mapping

Considering even the simple example we showed few typical situations, where the mapping from TCM to CM is not trivial (like for instance the naming conflicts naturally solved by paired path expressions explained in Section 3.3.3). Most of possible situations are discussed and classified in [13]. Concretely we name:

- data representation conflict, where corresponding literal properties in TCM and a concrete CM have different data types. An example is the $\{Paper\}year$ property (*String* and *Integer* types).

- missing literal property conflict, when a TCM concept attribute does not have its counterpart in CM. An example would be the $\{tcm:Researcher\}tcm:email$ attribute.
- concept-property and property-concept conflicts that can appear if a concept in TCM is modelled as a (literal) property in CM and vice versa.
- a few cases of schema isomorphism conflicts:
  - a TCM concept does not have its counterpart in CM. An example would be the *tcm:Group* concept.
  - a TCM concept literal property has only reversed counterpart in CM. An example is $\{tcm:Researcher\}tcm:papers$ that can be mapped into $\{cm:Member\}cm:created\_by^{-1}$ (or even $\{cm:Contributor\}cm:contributed\_by^{-1}$)
  - a TCM literal property is mapped (composed of) to multiple attributes in CM. An example is $\{tcm:Researcher\}tcm:name$ that is mapped into concatenation of $\{cm:Person\}cm:fnamname$ and $\{cm:Person\}cm:surname$
- generalization conflicts, when a TCM concept is mapped into CM concept with more specializations. An example of this would be the $\{tcm:Paper\}tcm:published\_at$ literal property that can be mapped into $\{cm:Paper\}cm:published\_as\{cm:Proceedings\}cm:ctitle$ and into $\{cm:Paper\}cm:published\_as\{cm:Journal\}cm:jtitle$

We do not mention all possible conflicts as for example integrity constraints conflicts that can arise when a more sophisticated constraints are put to concepts and their properties.

3.4.1 Data Representation Conflicts

In this case data types of corresponding literal properties are not compatible. A simple type conversion is made, concretely the type of a conflicting TCM attribute is transformed (possible during the model transformation since we can transform schemata) to the data type of corresponding CM attribute. Applied to our example the $\{tcm:Paper\}year\_at$ type *Integer* in the TAM is changed to *String* in the resulting AM.

3.4.2 Missing Literal Property Conflict

In this case such literal property (attribute) is omitted in resulting concrete AM. The example is the *email* attribute of the *Researcher.Details* in TAM that does not appear in the resulting AM (see Figures 7 and 9).

3.4.3 Concept-property and property-concept conflicts

If a concept in TCM is modelled as a literal property in CM or vice versa, in our example it is not present, but the second reversed problem property-concept would be the mapping of $\{tcm:Paper\}tcm:published\_at$ to $\{cm:Paper\}cm:published\_as$. This conflict is discussed in Section 3.4.5.

3.4.4 A TCM Concept Does not Have its Counterpart in CM

In this case the *NT2AM Transformer* must replace the missing concept with a single (virtual) constant concept, so every its attributes are constants. In our example the designers of the concrete domain model aimed it just for single (our) group. The group name then will be replaced with a constant string. The replacement by a constant is needed due to the fact that some top-level slices can be based on (non-existing) concepts. These slices are during the transformation process replaced by constant slices.

3.4.5 Generalization Conflict

This problem can typically occur when the TCM concept has specializations with different structure of its properties. The example is mapping of the $\{tcm:Paper\}tcm:published\_at$ attribute that can be mapped into $\{cm:Paper\}cm:published\_as\{cm:Proceedings\}cm:ctitle$, but also into $\{cm:Paper\}cm:published\_as\{cm:Journal\}cm:jtitle$ depending on the type of the publication (*Proceedings* or *Journal*).

The solution to this problem should be considered for two situations (as well as for other problems, but they appear to be simpler):

- Transformation of slices for presentation purpose (i.e. transformation of SELECT queries). In this case the result should the union of two queries containing both path expressions.
- Transformation of data manipulation queries. For the data consistency reasons the type of the manipulated (especially when new instances are created) concept should be determined, despite the fact there is no notion of these specializations in TCM. One of existing solutions we propose is the automated generation of the selection input field that allows the user to choose the type of concept to be created. In the example it would be selection between the *Proceedings* and *Journal* concepts when adding new publication.

3.4.6 A TCM Concept Property Has Only Reversed CM Counterpart

This situation occurs when a TCM property does not have direct counterpart in CM, but there is a CM property with

inverse semantics. There is no direct example, but $\{tcm : Researcher\}tcm : papers$ can be mapped to inversion of the union of $\{cm : Paper\}cm : created\_by$ and $\{cm : Paper\}cm : contributed\_by$.

3.4.7 A TCM Literal Property is Mapped to a Concatenation of Multiple CM Literal Properties

This is a case when an attribute is mapped to a concatenation of multiple literal properties. An example is the concatenation of $\{cm : Person\}cm : fnamname$ and $\{cm : Person\}cm : surname$ for the $\{tcm : Researcher\}tcm : name$. The solution is replacing of one TAM slice attribute in TAM by more attributes in the resulting AM

4 Software Tools

The usefulness of the approach described in this text relies in large extent on availability of tools supporting the NT design and automated deployment. The most essential tool is the *NT2AM Transformer* (Figure 2) that transforms NT specification using the mapping to a concrete domain CM to a concrete Hera AM or its part. This tool is a single Java application that reuses some classes from Hera Mediator [15] for the processing of articulations.

A design support tool for the graphical specification of mappings (articulations) is currently under development, and is based on an extended version of the EROS RDFS Explorer [14] that offers a convenient interface for building SeRQL queries, and thus also supports the building of path expressions which are the essential part of articulations. It will allow rapid and easy specification of needed articulations. The NT graphical design tool are based on existing CM and AM Builders (for construction of TCM and TAM) that are also used for graphical design of regular Hera applications (not using NT), the tools are being updated for specifications of NT interfaces.

5 Conclusion

In this paper we showed principles for building domain portable navigation templates reusing existing modelling techniques (from Hera in our case), existing knowledge (from the field of data integration), and also existing software (from the Hera Mediator). Although we chose a concrete method for demonstrating the approach, we believe that the idea of mapping from TCM to a concrete CM is rather universal. The advantage of our method compared to some other approaches lies in the possibility of accurately specifying data manipulations within a NT that are automatically transformed into proper data manipulation for a concrete domain where the NT is deployed.

References

[1] Openrdf, the serql query language, rev. 1.1. `http://www.openrdf.org/doc/users/ch06.html`.

[2] D. Brickley and R. V. Guha. Rdf vocabulary description language 1.0: Rdf schema. *W3C Recommandation 10 February 2004*.

[3] S. Ceri, P. Fraternalli, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann Publishers Inc., 2003.

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison Wesley, Reading, MA, 1995.

[5] J. Gomez and C. Cachero. Oo-h method: Extending uml to model web interfaces. *Idea Group Publishing*, pages 144–173, 2003.

[6] G. Houben, F. Frasincar, P. Barna, and R. Vdovjak. Modeling user input and hypermedia dynamics in hera. In *International Conference on Web Engineering (ICWE 2004), Munich, Germany*, 2004.

[7] N. Koch, A. Kraus, and R. Hennicker. The authoring process of the uml-based web engineering approach. In *Proceedings of The First International Workshop of Web-Oriented Software Technology*, 2001.

[8] F. Mannola and E. Miller. Rdf primer. *W3C Recommandation 10 February 2004*.

[9] O. Pastor, J. Fons, and V. Pelechano. Oows: A method to develop web applications from web-oriented conceptual models. In *Proceedings of International Workshop on Web Oriented Software Technology (IWWOST)*, 2003.

[10] G. Rossi, . Lyardet, F, and D. Schwabe. Patterns for e-commerce applications. In *Proceedings of Europlop 2000*, 2000.

[11] K. Sattler, S. Conrad, and G. Saake. Interactive example-driven integration and reconciliation for accessing database federations. *Inf. Syst.*, 28(5):393–414, 2003.

[12] D. Schwabe, G. Rossi, L. Esmeraldo, and F. Lyardet. Engineering web applications for reuse. *IEEE Multimedia*, pages 2–12, Spring 2001.

[13] A. Sheth and V. Kashyap. So far (schematically) yet so near (semantically). In *Proceedings of the IFIP WG 2.6 Database Semantics Conference on Interoperable Database Systems (DS-5)*. North-Holland, 1993.

[14] R. Vdovjak, P. Barna, and G. J. Houben. Eros: A user interface for the semantic web. In *7th World Multiconference on Systemics, Cybernetics and Informatics*, 2003.

[15] R. Vdovjak, F. Frasincar, G. J. Houben, and P. Barna. Engineering semantic web information systems in hera. *Journal of Web Engineering (JWE), Rinton Press*, 2(1-2):3–26, 2002.