Reusable Navigation Templates to Support Navigation Design in Hera

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

Abstract

Reuse is a fundamental concept in 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 in the design of web applications. Concretely, we discuss the reuse of navigation templates to specify (parts of) navigation models in different application domains based on different data sources. While supporting this diversity of applications, at the same time navigation templates should allow 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 framework and its HPG software.

1 Introduction and Related Work

One of the major concepts in software design is the 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, and 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 the development of web applications. The support of navigation and the necessity of navigation modelling is what distinguishes web systems and web design methods from traditional systems and design methods. In the navigation modelling, reuse of navigation structure specification (often referred to as ”Web Patterns” or ”Web Design Patterns”) is a very useful technique, but its full exploitation is still an open problem. A good overview of common web patterns is presented in [11]. The descriptions of patterns presented there serve as a set of handy guidelines for web designers. Existing software libraries offer a wide variety of useful generic primitives that can be (re)used during the construction of web applications, but they rarely contain larger navigation patterns. Since navigation models are usually tightly coupled with concrete domains, specified by Conceptual Models (CM), the achievement of the domain portability is not a trivial task.

Current methods for web design already 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. Most of the units however are associated with concrete data (specified in a data model), so using such composed patterns for different domains is not trivial. Object-oriented approaches like OO-H [5], UWE [7], or OOWS [10] show a solid approach supporting object-oriented reuse techniques like class abstraction. The problem of domain portability of navigation models in object-oriented environments using the OOHDM method is discussed in [13]. The web design framework introduced there represents abstract navigation models that are isolated from concrete domains and can be instantiated to a concrete domain. The deployment process consists of the derivation of a concrete OOHDM model from an OOHDM-frame (a generic conceptual model). It is possible to build common navigation patterns, however the navigation classes are derived from conceptual classes describing a concrete domain.

In this paper we propose a practical approach for the design and deployment of domain portable reusable navigation patterns called Navigation Templates (NT). We focus on the explanation of the mapping from a Template Conceptual Model (TCM) describing the structure of data used within an NT, to a concrete domain conceptual model (CM).

For an NT, such mapping is defined for every concrete domain, and it is a kind of NT parametrization. It allows not only a relatively easy deployment of an NT to a concrete domain, but it also facilitates the specification of possible data manipulations in an NT. A mapping is similar to a data (schema) integration model, and we can benefit from existing knowledge in this field of research. The process of deployment of such an NT to a concrete domain can be automated by using an NT specification and an appropriate mapping to a concrete domain. This deployment process is demonstrated on two examples using the Hera framework [6, 16]. Despite the use of a concrete method for reasons of illustration the proposed approach of mapping NTs to concrete domains can be used for other methods.

Section 2 explains the requirements for NTs and the context of their usage. The core of the paper is Section 3 that explains the approach in detail using two examples in Hera and HPG (Hera Presentation Generator web server software). Potential mapping (data integration) problems and solutions are also discussed here. The current work on the creation of software tools supporting the design and deployment of NTs for Hera is briefly explained in Section 4 and the text is concluded by Section 5.

2 Navigation Templates Overview

A Navigation Template (NT) is a parameterized conceptual specification of a navigation structure. This specification has well-defined interfaces in terms of links and types of information they can carry. The NT parametrization allows its deployment within existing navigation models of similar applications based on different data domains. An example of a primitive NT is a user-selection device (a virtual shopping basket) that is used in a web application for an online sport equipment shop as a classical shopping basket, and in a university online library it can be used for instance as a tool facilitating the searching of publications by choosing topics of interest.

Besides the navigation structure, NTs define also some basic application logic (functionality), in most cases related to dynamic updates of the navigation structure and underlying data, both based on the user interaction. Although NTs represent conceptual reusable units, we demonstrate how they can be converted to a specification that is directly used by a web server software to provide desired functionality on the Web.

For the generation of a deployed NT we need two kinds of specification:

- An **NT specification** that contains two sub-models:
  - The **Template Conceptual Model** (TCM) describes the structure of data concepts (and their concept relationships) that are used in the description of the NT's navigation structure (TAM). Note that the content domain described by a TCM is not necessarily materialized, but the TCM is used in the parametrization when the NT is deployed.
  - The **Template Application Model** (TAM) describes the navigation structure of the NT and its application logic. It is based on the data defined in a TCM.
- An **NT parametrization** that defines the mapping from the TCM to a concrete domain. This mapping allows a (semi-)automatic translation of the NT into (parts of) a concrete navigation model.

With a certain level of abstraction we can see an analogy between NTs and MDA [9] models. A main concern of MDA is the support for the design and integration of systems based on different platforms. In the same spirit, NTs represent domain independent models, so if we replace the notion of platform with the notion of domain, a PIM (Platform Independent Model) of MDA can be compared to a domain independent NT specification, and PSM (Platform Specific Models) to concrete deployed NTs. Analogically, domain independent NTs can be (semi)automatically deployed for different domains and frameworks using different transformation (deployment) tools.

2.1 Benefits of NT Reuse and Methodological Issues

The main benefit of building NTs and their later deployment lies in saving development effort for parts of web applications that can be used again and again for different domains. An interesting application of NTs is the composition of new web applications from an available NT on already existing domains (for instance on legacy databases). Figure 1 sketches how NTs can be deployed within a navigation model. The thick arrows represent hyperlinks (possibly) carrying parameters (the depicted internal structure of the NT does not reflect any real structure and is sketched only for illustration purposes). The thin arrows show the deployment process with the transformation of an NT specification to a concrete (part of) navigation model based on a CM. This transformation is automatic, but uses a manually built 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 (larger) domain). The real benefit of NT as a conceptual unit of reuse depends on several aspects including:

- generality of NT design, in the sense of how easily they can be used for different applications. This is in-

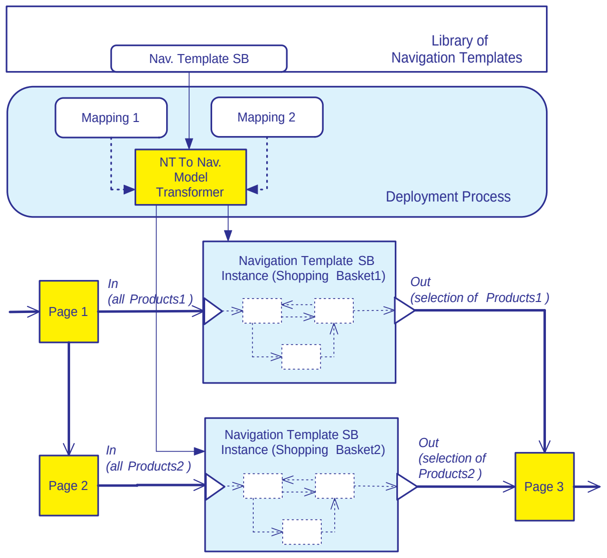

Library of
Navigation Templates
Nav. Template SB
Mapping 1
Mapping 2
NT To Nav. Model Transformer
Deployment Process
Navigation Template SB Instance (Shopping Basket1)
In (all Products1)
Out (selection of Products1)
Page 1
Page 2
In (all Products2)
Navigation Template SB Instance (Shopping Basket2)
Out (selection of Products2)
Page 3

Figure 1. NT deployment example

fluenced by a good selection of ”typical” web application patterns NTs represent, and also by minimizing the structure of NTs (smaller and simpler units are easier to deploy and specialize).

- complexity of NT deployment, including:
  - complexity of its parametrization. That can be influenced by proposing simple TCMs with a minimal number of mappings to a concrete domain. The mapping specification can be facilitated by design tools.
  - automation of the transformation of the NT and appropriate mapping to a concrete models and/or executable specification eventually. This is given by availability of appropriate software translators.

In the context of the NT design process, we consider two possible types of the design cycle:

- **Data-driven design**, where first the TCM (or a set of TCMs) is defined and then the TAM is built on top of it (in other words, first the data, then the navigation). This approach can be used when a simple and straightforward NT parametrization (mapping to a concrete domain) is vital. For instance, when we want to use concrete, complex legacy databases, TCMs are better starting points. A concrete web application based on such a legacy database then can be easily composed from already built NTs, because the structures of the concrete TCM are in accordance with the CM at hand.
- **Process-driven design**, where on the basis of user requirements, a process model is defined for an NT and then subsequently enriched with appropriate data model and data manipulation specifications (in other words, first the process, then the navigation). From the data model an NT specification can be derived. This interesting, process-driven approach is part of ongoing work and will be described in a separate paper.

The details of the NT specification and deployment techniques may depend on the concretely used approach. In the following text we explain the principles of NT specification and deployment using two examples. The first example demonstrates a multiple use of a simple NT (in this case a guided tour) in a single application, and the second highlights potential problems associated with mapping a TCM to a concrete domain and their resolution.

3 Navigation Templates in Hera

For a better explanation of the NT concept, we demonstrate its basic principles using the Hera framework and two examples. In the Hera design cycle, an NT can be generated from a more abstract process model, or it can be designed manually. The role of NTs in the method and its models is depicted in Figure 2. It has already been mentioned, that an NT contains a TCM describing the structure of the information that will be presented and processed, and it contains an appropriate TAM specifying the navigation view on the TCM. The *Articulations* (see [16]) represent the NT parametrization - the mapping from the TCM to a concrete CM, i.e. the ”binding” of the TCM to the concrete domain. The NT specification together with the *Articulations* are used by the *NT2AM Transformer* to generate a concrete (part of an) AM describing the navigation structure and functionality of a concrete web application. The AM can then directly be used by a Hera engine such as HPG-Java for the online generation of pages in the web application.

3.1 Brief Overview of Hera

Within the Hera project we investigate methods for the specification of (dynamic) hypermedia presentations and we build and maintain appropriate server software and design tools. The methodology determines a number of design steps resulting in a set of models (that specify how the hypermedia presentations get generated). The conceptual design phase results in a Conceptual Model (CM) defining the structure of source data used in presentations. The application design phase results in an Application Model (AM) defining a navigation structure over the CM, possibly with data manipulation associated with user actions. The presentation design phase produces a Presentation Model (PM) specifying the layout of presentations. All Hera models are expressed in RDFS [2].

These models are used by a Hera engine, in this case HPG-Java (a software module running as a servlet hosted by a web server), first performing data retrieval, and then performing data transformations resulting in presentation pages, possibly for different platforms and different formats (HPG supports HTML, WML, and SMIL for presentations without data manipulation, and HTML for presentations allowing forms and data manipulation). The bottom part of Figure 2 shows the transformations of the Hera/HPG pipeline, where retrieved data is transformed consecutively to a CM instance (CMI), an AM instance (AMI), and a presentation in a concrete format (e.g. HTML). All intermediate data chunks are internally represented in RDF [8].

The application modelling method is capable of expressing more advanced functionality than only a navigation view over static data content. It supports modelling of user inputs by means of forms allowing users to enter arbitrary information possibly exploited by data manipulation queries. All queries in Hera AM are expressed in the SeRQL [1] RDFS query language with slight modifications (queries are pre-processed by the Hera engine) including the management of session parameters (variables).

An AM contains basic building blocks called slices that describe the structure of navigation pages (or their parts since they can be nested), and their linking (see Figure 3). Slices can have root concepts (from the CM) drawn as large ovals in the slice upper part. If a slice does not have the root concept, it is a constant slice and it can have arbitrary content. If the target of a link is a non-constant slice, the link carries parametrization that determines the instantiation of the target slices (the anchor determines what instance of the target slice root concept is used for the target slice instantiation). The instantiation of slices can be determined also by queries associated with slices. A slice can contain attributes (literal properties in the CM) from a root concept and attributes from concepts related to the root concept (the bottom part of the slice shape) connected with the root concept by CM properties. User interaction is facilitated by the use of forms that carry a number of input fields users can fill in. Forms have associated processing queries that can retrieve data, change the data content, or update values of session variables.

3.2 Guided Tour Example

In the first example we demonstrate the multiple use of a very simple NT. The NT represents a guided tour - a 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 the TCM to a CM).

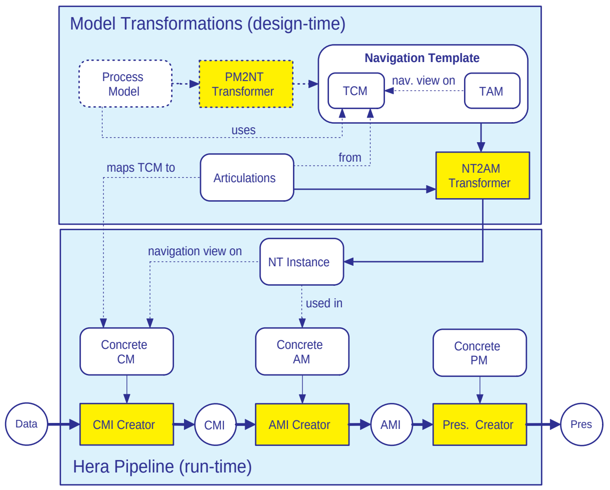


Figure 2. Role of NTs in Hera architecture

Figure 3 presents the TCM and TAM of the guided tour NT. It uses a special predefined sub-class of general slice (iterator) that allows easy implementation of a guided tour. A slice of type *Iterator* comes with a 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 shows an AM of a simple museum application using two instances of the Guided Tour NT. They contain attributes based on a mapping from the TCM to the CM and also attributes not appearing in the original NT, but added by the designer. The application presents a set of painting techniques (the *TechniquesList* slice), for every technique a list of paintings exemplifying the technique (the *Technique.PaintingList* slice), and the two guided tours for browsing through the painters and paintings (*GD-Painters* and *GD-Paintings*).

3.3 Publications Example

In the second example we demonstrate possible problems with mapping a TCM to a concrete CM. To a large extent, this is a problem of data integration. We show some typical cases appearing in RDFS schema integration and present some practical solutions covering the needs of our method (including the rewriting of selection and data manipulation queries). Most of the problems have been recognized and studied in [12, 14]. 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 an existing data

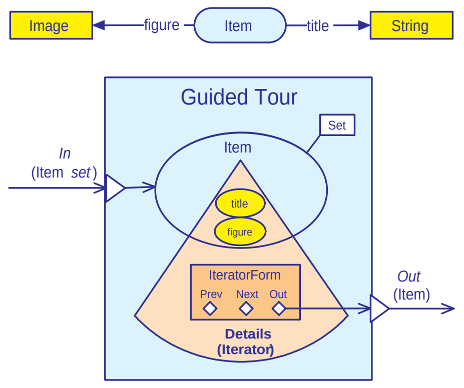


Figure 3. Guided Tour NT, the TCM on top, the TAM below

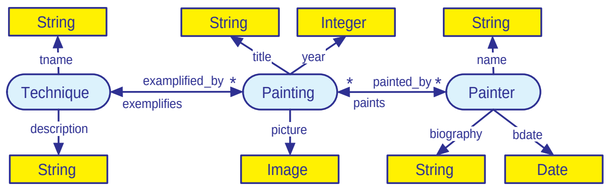


Figure 4. CM of a museum application

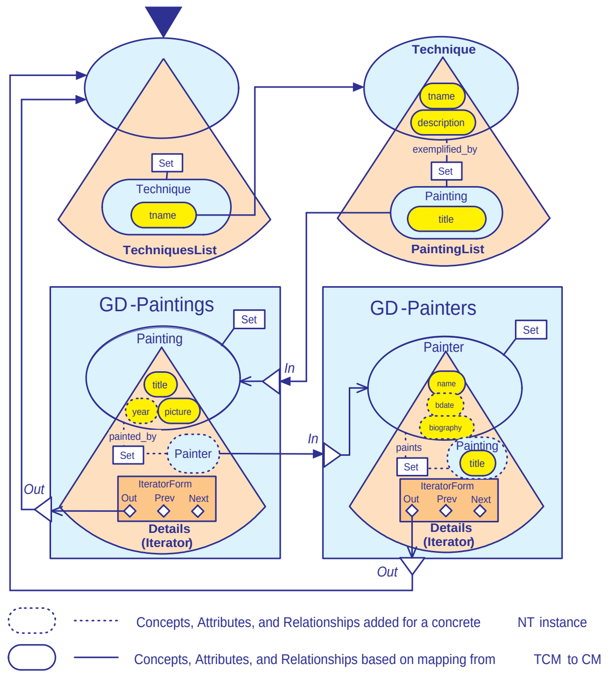


Figure 5. AM of a museum application deploying instances of the Guided Tour NT

source with a different data structure. A set of articulations is defined.

3.3.1 Template Conceptual Model

The TCM contains only those concepts, concept properties, and literal properties that are necessary for describing the core navigation structure and functionality (we show only the addition of a publication) associated with a concrete NT. The TCM of the Publications NT is presented in Figure 6.

3.3.2 Template Application Model

Unlike the first example, here the TAM also contains the specification of input forms and their processing. The Publications TAM shown in Figure 7 consists of three slices presenting research groups (the *Group* slice), listing researchers in groups (the *Group.Researcher* slice), and listing publications of a researcher (the *Researcher.Publications* slice). The *AddPaper* slice allows adding a publication created by a concrete researcher (only one in our example). For the sake of simplicity the application has not been detailed completely. We omit here more comprehensive data manipulations, like removing publications, adding a new researcher, and managing groups. The *AddPaper* query is activated when the *AddPaper* form is submitted:

```
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 represent the ID of the last presented researcher (refreshed by the *SetResearcher* query during the *AddPaper* slice instantiation and temporarily stored in the *session:resID* session parameter). The *SetResearcher* query is very simple:

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

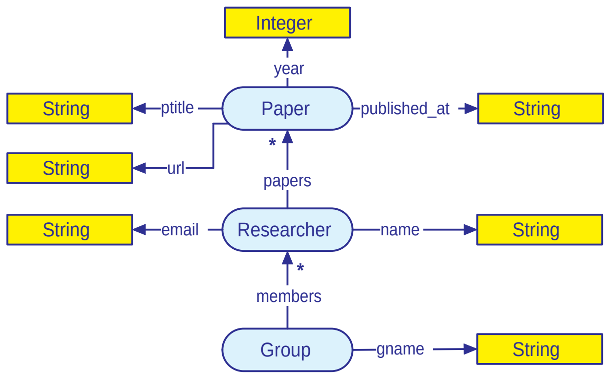


Figure 6. Publications TCM

The *session:sliceid* is a default session variable containing the URI of the root concept instance of the last completely instantiated slice (that is why it is attached to the *AddPaper* slice and not to *Researcher.Publications* slice, although it contains the URI of the current *Researcher*). The value of the *R* variable is in the RDFS TAM specification assigned to *session:resID*. The complete query and session parameter specification does not appear in the TAM diagram, but it is in the TAM RDFS file:

```
<rdfs:Class rdf:ID="SetResearcher"
            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>
```

3.3.3 Mapping NTs to Concrete Domains

A necessary condition for the automated transformation of an NT to an AM for a concrete CM is the existence of a mapping from the abstract TCM to a concrete domain model. We demonstrate the specification of such a mapping using the Publications example and show possible situations. Figure 8 presents a concrete CM describing a domain of publications. There are two categories of mapping. The first one is concept-to-concept, which facilitates the determination of root concepts and data manipulation queries during transformation of the TAM slices into concrete AM slices. The second one is attribute-to-attribute, which allows the transformation of slice attributes and is used in query transformations as well. We now define mappings of type concept-to-concept and attribute-to-attribute. We use path expressions specifying concept-property chains in the form $\{Concept1\}property1\{Concept2\}...$. Inverse properties are denoted as $\{Concept\}property^{-1}$. If the value of a TCM property is constructed from the values of several properties in the CM (concatenation), we write it as $\{Concept1\}property1 \odot \{Concept2\}property2$. The fact that the value of a TCM property is retrieved from several CM properties is captured as $\{Concept1\}property1 \cup \{Concept2\}property2$. In this case the mapping is a union of values of the given path expressions. The mapping of concepts relies on the uniqueness of the concept names (in other case we would need to use path expressions as well), in our example the mappings for the concepts are:

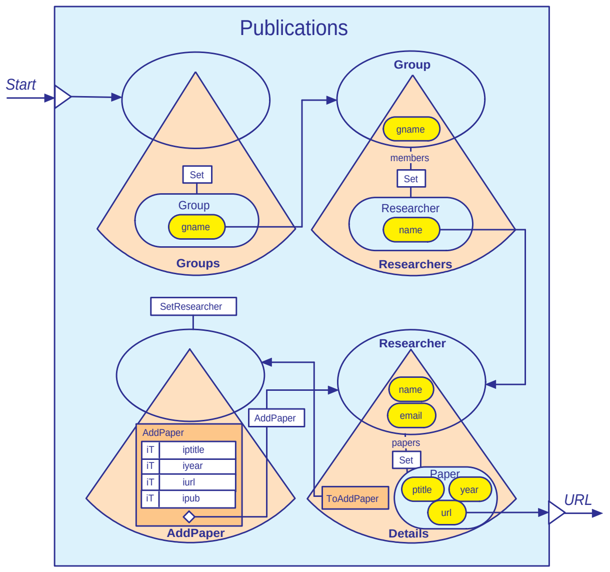


Figure 7. Publications TAM

- for *tcm:Group* no mapping is defined
- *tcm:Researcher* is mapped to *cm:Person*
- *tcm:Paper* is mapped to *cm:Paper*

For the mapping of attributes we define articulations containing pairs of path expressions for the TCM and the CM. The mapping is described in Table 1. Further details of the mappings are explained in Section 3.4.

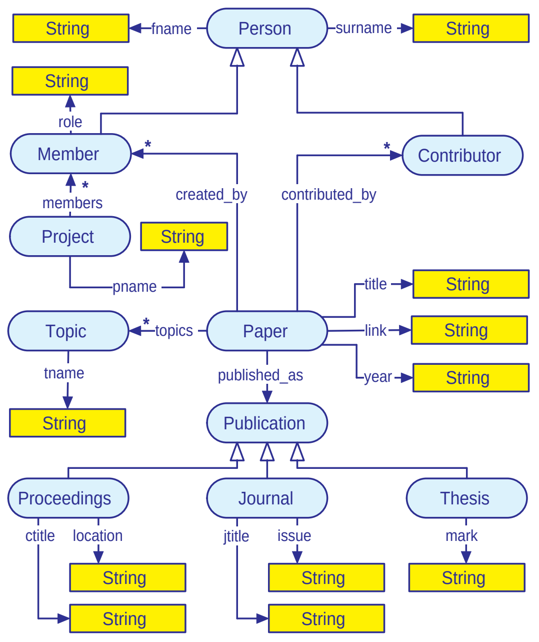


Figure 8. Concrete Publications domain CM

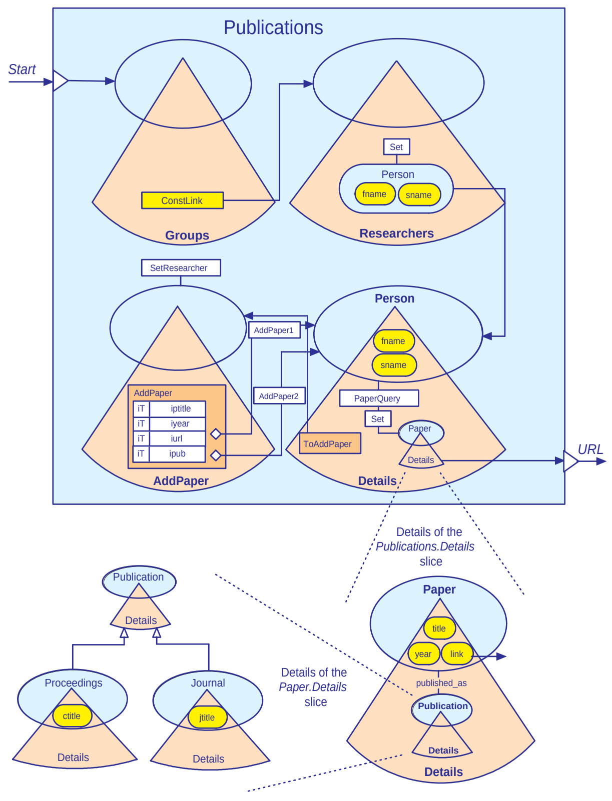


Figure 9. Deployed Publications NT

Attribute in TCM	Mapping to CM
$\{Group\}gname$	no appropriate range concept in CM, and thus no mapping. In CM it will be represented by a constant string (a name of a working group)
$\{Researcher\}name$	$\{Person\}fname$ $\odot$ $\{Person\}surname$
$\{Researcher\}email$	no appropriate attribute in CM
$\{Researcher\}papers$	$\{Member\}created\_by^{-1}$ $\cup$ $\{Contributor\}contributed\_by^{-1}$
$\{Paper\}ptitle$	$\{Paper\}title$
$\{Paper\}published\_at$	$\{Paper\}published\_as$ $\{Proceedings\}ctitle$ $\cup$ $\{Paper\}published\_as$ $\{Journal\}jtitle$
$\{Paper\}year$	$\{Paper\}year$
$\{Paper\}url$	$\{Paper\}link$

Table 1. TCM to CM attribute mapping

3.3.4 Deployed Navigation Template

When the *Articulations* are specified, an appropriate deployed NT can be generated. A deployed NT is an (part of) AM. In this process slice relationships based on the TCM are replaced by those based on the CM at hand. Due to possibly different schema structures of a TCM and a concrete CM, in some cases simple slice aggregations based on a single CM relationship must be replaced by more complex queries. For instance, this is the case when for a path expression in the TCM with the length (number of properties in the path expression) one, there exists a corresponding path expression in the CM with length more than one (the result would be a join query). During the deployment process the *papers* slice aggregation in the original TAM *Researcher.Details* slice is automatically transformed to a query (we name it here *PaperQuery*) that is a union of the two queries (for the *Person* subclasses *Member* and *Contributor*):

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

and

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

where *P* is an instance of the *Person* concept given by the *Person.Details* slice instance.

The *AddPaper* query appearing in the original TAM is during the deployment process automatically transformed into two different queries, one for the *Proceedings* subclass of *Publication*:

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

and one for the *Journal* subclass of *Publication*:

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

The queries create different *Publication* types (subclasses) with different attributes. The user of the application decides what kind of *Publication* he wants to add. This is facilitated by two buttons (one for each subclass and executing the first or the second query) that are automatically generated (during the NT deployment process) and placed to the *AddPaper* form (see Figure 9). Note the simplification we made here due to the lack of space. Both queries are applicable for the *Member* type of *Researcher* (can be determined by the *cm:created_by* property appearing in the CONSTRUCT clause). In a real example, every form button would execute two optional queries depending on the type of last visited *Researcher* (*Contributor* or *Creator*).

3.4 Major Problems in TCM to CM Mapping

We can highlight a few typical situations, where the mapping from the TCM to a CM is not as straightforward as for instance the naming conflicts naturally solved by paired path expressions explained in Section 3.3.3. In this section we mention the conflicts and their possible solutions. This descriptions are used as guidelines for developing the NT deployment software (*NT2AM Transformer* in Figure 2). Most of the possible situations have been discussed and classified in [14]. Concretely we name:

- Data representation conflict: corresponding literal properties in the TCM and a concrete CM have different data types. An example is the $\{Paper\}year$ property (*String* and *Integer* types).
- Missing literal property conflict: a TCM concept attribute does not have its counterpart in the CM. An example would be the $\{tcm:Researcher\}tcm:email$ attribute.
- Concept-property and property-concept conflicts can appear when a concept in the TCM is modeled as a (literal) property in the CM and vice versa.
- A few cases of schema isomorphism conflicts:
  - A TCM concept does not have its counterpart in the CM. An example would be the *tcm:Group* concept.
  - A TCM concept literal property has only a reversed counterpart in the CM. An example is $\{tcm:Researcher\}tcm:papers$ that can be mapped to $\{cm:Member\}cm:created\_by^{-1}$ (or to $\{cm:Contributor\}cm:contributed\_by^{-1}$).
  - A TCM literal property is mapped to (composed of) multiple attributes in the CM. An example is $\{tcm:Researcher\}tcm:name$ that is mapped into a concatenation of $\{cm:Person\}cm:fname$ and $\{cm:Person\}cm:surname$.
- Generalization conflicts where a TCM concept is mapped into a CM concept with more specializations. An example of this would be the $\{tcm:Paper\}tcm:published\_at$ literal property that can be mapped to $\{cm:Paper\}cm:published\_as\{cm:Proceedings\}cm:ctitle$ and to $\{cm:Paper\}cm:published\_as\{cm:Journal\}cm:jtitle$

We do not mention other possible conflicts such as integrity constraint conflicts that can arise when more sophisticated constraints are imposed on the concepts and their properties.

3.4.1 Data Representation Conflicts

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

3.4.2 Missing Literal Property Conflict

In the case of such a conflict such a literal property (attribute) is omitted in the resulting concrete AM. An 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

This conflict appears if a concept in the TCM is modeled as a literal property in the CM or vice versa. An example of the property-concept conflict is $\{tcm : Paper\}tcm : published\_at$ that is mapped to $\{cm : Paper\}cm : published\_as$. This conflict is discussed in Section 3.4.5.

3.4.4 A TCM Concept Does not Have a Counterpart in the CM

In this case the *NT2AM Transformer* must replace the missing concept with a single (virtual) constant concept, so all its attributes are constants. For instance, the example publication CM is intended for a single research group, so the group name 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. During the transformation process these slices are replaced by constant slices.

3.4.5 Generalization Conflict

This problem typically occurs when the TCM concept has specializations with a different property structure. An example is the 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 needs to cover the following two situations (as well as some other problems, but they appear to be simpler):

- Transformation of slices for presentation purposes (i.e. transformation of SELECT queries). In this case the result should be the union of two queries containing both path expressions.

- Transformation of data manipulation queries. For the data consistency reasons the type of the manipulated concept should be determined (especially when new instances are created), despite the fact that there is no notion of these specializations in the TCM. One of the possible solutions is the automatic generation of a selection input field allowing the user to choose the type of concept to be created (according to existing concept subclasses). In the example it would be a selection between the *Proceedings* and *Journal* concepts when adding a new publication.

3.4.6 A TCM Concept Property Has Only a Reversed CM Counterpart

This situation occurs when a TCM property does not have a directly matching counterpart in the CM, but there is a CM property with inverse semantics. There is no direct illustration of this in the example, but $\{tcm : Researcher\}tcm : papers$ can be mapped to the union of the inversions of $\{cm : Paper\}cm : created\_by$ and $\{cm : Paper\}cm : contributed\_by$.

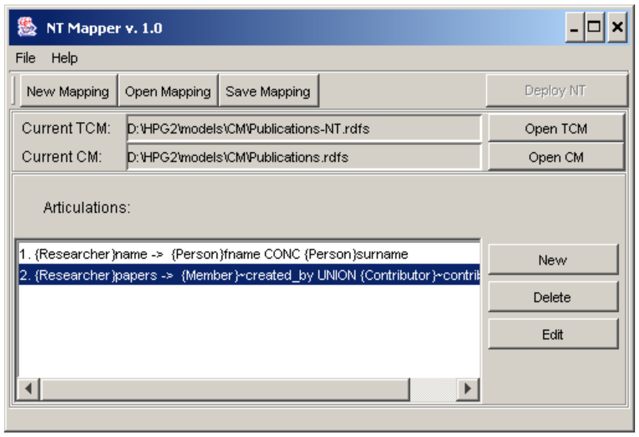
3.4.7 A TCM Literal Property is Mapped to a Concatenation of Multiple CM Literal Properties

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

4 Implementation

The usefulness of the approach described in this text relies to a large extent on the availability of tools supporting the design and automated deployment of NTs. The most essential tool is the *NT2AM Transformer* (Figure 2) that transforms an NT specification to a concrete Hera AM or a part of it using the mapping to a concrete domain CM. This tool is a single Java application that reuses some classes from the Hera Mediator [16] for the processing of articulations.

A design support tool for the graphical specification of mappings (articulations) is currently under development, and uses part of the functionality of the EROS RDFS Explorer [15] that provides an 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 main window of the tool is shown in Figure 10. The NT graphical design tools are based on existing Hera CM and AM Builders (for the construction of the TCM and TAM) that are also used for the regular (graphical) design of Hera applications (i.e. without using an NT). These tools are being updated for specification of the NT interfaces.



Current TCM: D:\HPG2\models\CM\Publications-NT.rdfs

Current CM: D:\HPG2\models\CM\Publications.rdfs

Articulations:

1. {Researcher}name -> {Person}fname CONC {Person}surname
2. {Researcher}papers -> {Member}~created_by UNION {Contributor}~contrib

Figure 10. The main window of the mapping tool

5 Conclusion

In this paper we have shown the principles of building NT specifications that are portable over domains. These principles use existing expertise of modelling techniques and data integration. In the implementation we exploit software packages we have already developed, for instance the Hera Mediator and the EROS RDFS explorer. Although we chose a concrete (Hera) method for demonstrating the approach, we believe that the idea of mapping from a TCM to a concrete CM is rather universal. The advantage of our method compared to some other approaches lies in the possibility of precise specification of the navigation structure and the data manipulation within an NT that is subsequently automatically transformed to appropriate specification matching a concrete domain. Thus, this approach and its implementation will facilitate the reuse of navigation primitives in web engineering.

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] J. Miller J., Mukerji. Mda guide version 1.0.1. *OMG*, June 2003.

[10] 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.

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

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

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

[14] 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.

[15] 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.

[16] 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.