Visualization of Introspective Data Models with Tricia

Thomas Büchner, Florian Matthes, Christian Neubert

Chair for Informatics 19 (sebis), Technische Universität München
Bolzmannstr. 3, 85748 Garching, Germany

{buechner,matthes,neubert}@in.tum.de

Abstract

We present three different visualizations of data models, which assist the development process of Tricia, an open source Java platform used for the development of data intensive web applications. Tricia follows an introspective data model driven approach to system implementation where substantial parts of the application semantics are captured by domain-specific models (data model, access control model and interaction model). The visualizations presented in this paper are extracted from Java code by model introspection. Based on these visualizations we identify areas for further research and system development.

1 Introspective visualizations with Tricia

Tricia is an open source platform supporting model driven software development in the domain of data-intensive web applications, developed by members of our chair. The core of Tricia provides modeling languages tailored specifically to the needs of this domain and consists of the three frameworks: data modeling, access control and interaction.

Tricia is written in Java and follows the *introspective* model-driven approach[2]. That means, in Tricia (data) models are represented as Java classes, which instantiate and customize classes of the introspective framework core. These customizations have to follow specific introspective programming models, which restrict the expressiveness of the Java programming language. As a consequence, models of applications developed based on the Tricia frameworks can be extracted from source code at development time by introspection[2], and can be accessed as Java objects at runtime. These models are available throughout the entire application life cycle and custom views on these models can be provided for different stakeholders (e.g., software architects, software developers, customer).

Since all of the model views are generated by model introspection, they are always synchronized with the actual implementation of the system and can never 'lie'. Therefore, involved stakeholders can always rely on these views within the entire application life cycle facilitating long-live systems.

In this paper we focus on visualizations of the introspective data models. The exemplary visualizations introduced in this Section are based on the sample data model fragment shown in Figure 1, consisting of the concept `Blog` which references its containing `BlogPosts`.

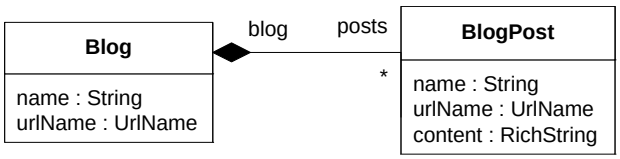

Figure 1: Exemplary concepts Blog and BlogPost in UML notation

Tricia currently supports three kinds of views on data models:

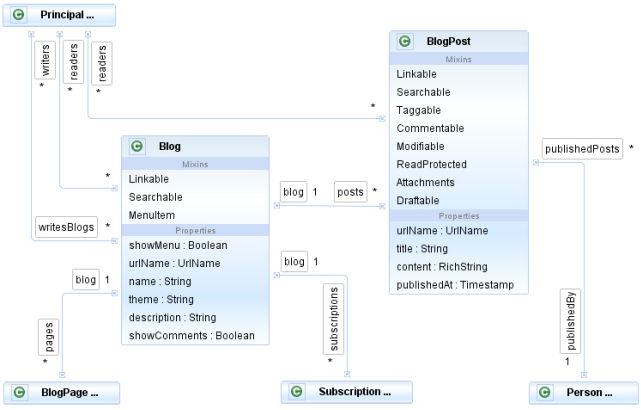

Figure 2: Introspective graphical view of the concepts Blog and BlogPost

A graphical representation similar to UML class diagrams shown in Figure 2. This visualization gives an overview of the concepts and their relationships. It hides implementation details in order to show the overall structure of a system. The graphical view is the main communication medium for developers and architects, but can also be used to discuss the structure with customers. The implementation of

the graphical view is based directly on the extracted model, and is highly customized.

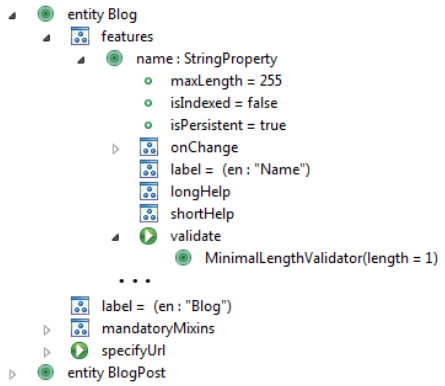

Figure 3: Introspective tree view of the concept Blog

The tree view (see Figure 3) contains all available model information. The amount of detail visible can be chosen by expanding and collapsing nodes. It is possible to navigate from the tree model view to the corresponding code artifact. The tree view of the data model is based on a generic tree view, which generically visualizes introspective models. Only few customizations were necessary to build this data model view.

```
entity Blog
  features
    name : StringProperty
      maxLength = 255
      isIndexed = false
      onChange
        updateUrlName (Blog.urlName)
      label =  (en : "Name")
      validate
        MinimalLengthValidator(length = 1)
      ...
    pages : ManyRole (BlogPage)
      isCascadeDelete = true
      label =  (en : "Pages")
      oppositeRole = blog : OneRole
      ...
  label =  (en : "Blog")
  mandatoryMixins
    Linkable
    Searchable
    MenuItem
  specifyUrl
    patternString = /blogs/{urlName},
      handlerClass = de.infoasset.blog.handler.blog.BlogHandler
entity BlogPost
  ...
```

Figure 4: Introspective textual view of the concept Blog

A textual view serializes all available model information in a readable notation as shown in Figure 4. This view provides the most detail and is suitable for discussing implementation of the data model extensively. This textual representation of the data model has the positive characteristic, that it can be edited and compared easily with many text editing tools.

Since Tricia is written in Java, we based the technological realization of model extraction and visualization on the Eclipse IDE. Therefore, all visualizations can be easily accessed during the development process, since they are realized by Eclipse plugins available to all Tricia developers.

2 Related Work

There exist numerous approaches to model driven web development, which all use the prevailing *generative* approach to modeling. The introspective approach extracts models from the source code through introspection, which improves the integration of the models with the underlying system [3]. Our introspective approach realized with Tricia is closely related to the one introduced in [1] where declarative model views are extracted from Java source code. In [1] this idea is being applied to behavioral models.

The idea of extracting and visualizing models from source code is also pursued by reverse engineering approaches. These approaches do not make assumptions about the internal structure of the system to be analyzed, which puts limits on the precision and expressiveness of the extracted models. Reverse engineering approaches are focused on the maintenance phase of the development process, whereas the introspective approach applies as well to the development phase.

3 Further Research

Currently we are in the process of examining the use of the presented visualizations in industry projects in order to find out, which view is useful for which stakeholder in which phase of the development process.

Another consideration relates to the question how round-trip engineering can be realized based on the introduced visualizations.

Furthermore, we are working to expand the presented approach to visualize introspective models of the Tricia access control and interaction frameworks.

References

[1] Moritz Balz, Michael Striewe, and Michael Goedicke. Embedding Behavioral Models into Object-Oriented Source Code. In Peter Liggesmeyer, Gregor Engels, Jürgen Münch, Jörg Dörr, and Norman Riegel, editors, *Software Engineering*, volume 143 of *LNI*, pages 51–62. GI, 2009.

[2] Thomas Büchner and Florian Matthes. Introspective model-driven development. In *EWSA*, pages 33–49, 2006.

[3] Thomas Büchner and Florian Matthes. Using Framework Introspection for a Deep Integration of Domain-Specific Models in Java Applications. In *Proceedings of the 1. Workshop des GI-Arbeitskreises Langlebige Softwaresysteme (L2S2): Design for Future - Langlebige Softwaresysteme*, pages 123–135, 2009.