Model-Driven Development of Content-Based Image Retrieval Systems

Temenushka Ignatova, Andreas Heuer
Database Research Group
Department of Computer Science
University of Rostock
Albert-Einstein-Str. 21, 18051 Rostock, Germany
{temi,heuer}@informatik.uni-rostock.de

ABSTRACT: Generic systems for content-based image retrieval (CBIR), such as QBIC [7] cannot be used to solve domain-specific image retrieval problems, as for example, the identification of manuscript writers based on the visual characteristics of their handwriting. Domain-specific CBIR systems currently have to be implemented bottom up, i.e. almost from scratch, each time a new domain-specific solution is sought. Inspired by the recognition, that CBIR systems, although developed for different domain-problems, comprise similar building blocks and architecture, the idea of adopting model-driven development techniques for generating CBIR systems was elaborated. To support the design of domain-specific CBIR-Systems on a conceptual level by reusing data structure and functional interfaces a framework model is developed, which can be used to derive concrete domain-specific CBIR models. A transformation approach for the generation of a platform-specific implementation on top of an object-relational database from the concrete conceptual model is proposed. Finally, how these techniques can be applied for the design of a CBIR system for the identification of music manuscript writers based on the visual characteristics of their handwriting is demonstrated.

Classification and subject descriptors
I 4 [Image Processing and Computer Vision]; H 3.1 [Content Analysis and Indexing] I 4.10 [Image representation]

General Terms
Image processing, Content development, Image retrieval systems

Keywords: model-driven development, content-based image retrieval

Received 24 July 2006; Revised and accepted 17 July 2007

1. Introduction

During the research project eNoteHistory [2], in which a specialized CBIR system for the identification of writers of historical music manuscript was designed and implemented, existing CBIR systems were studied and classified according to their purpose into the following categories. Generic CBIR Systems (e.g., QBIC [7], imgSeek [10], IMatch [15]) make use of generic low-level features such as color, texture and shape and are not suited for carrying out a specialized image retrieval task. Specialized CBIR Systems, such as a system for recognizing similar images in a set of 2D-Electrophoresis Gel Images are implemented only for a special domain and are normally highly effective for this domain, but cannot be applied effectively in any other applications. CBIR frameworks (e.g., GIFT [13], PicSOM [11], VizIR [6]) offer extensible software architectures for developing domain-specific CBIR application. However, these frameworks are implemented for special platforms and do not offer flexible data storage possibilities.

The result of adapting these frameworks for a specific domain application is not a compact, specialized application, but rather an extended version of the large framework application.

To facilitate the development of tailor-made domain-specific CBIR systems the idea of incorporating model-driven development techniques for modeling and generating CBIR systems for various implementation platforms was investigated. This approach could be very useful for building scientific image retrieval applications, where images originate from various specialized domains. In Figure 1 an overview of the model-driven development architecture for CBIR systems is shown. Two main groups of techniques, which have to be provided, can be distinguished.

The first group comprises components for creating a platform independent model of the CBIR system. These components make use of the framework model proposed in this paper. The framework model provides a starting point for the conceptual modeling of the complex data structures, storage and retrieval operations of CBIR systems. The second group of techniques comprises components for transforming the concrete CBIR system conceptual model into a specific implementation. The generated core data structure and functionality of the CBIR system can be used by different client applications. Multiple client applications may be implemented to meet diverse user needs. Therefore, the aim of the current work is not to provide a particular graphical user interface for the system. Usually CBIR systems require the design of complex user interfaces to support user-friendly interaction with the CBIR core. Furthermore different technical environments, such as mobile devices, set special requirements on user-interfaces. Therefore, additional information apart from the basic functionality and data structure of the system has to be considered when modeling graphical

Figure 1. Model-driven development tools for CBIR systems
user interfaces. For the model-driven design of advanced user interfaces a technique based on task models has been proposed in [21].

In the following sections the conceptual framework model for the different parts of the CBIR system and possibilities for its mapping onto an object-relational database management system (ORDBMS) are described.

2. Modeling of CBIR systems

For the design of almost each existing CBIR system a conceptual image data model has been used. These models have a lot in common, but very often they remain application specific, such as image models for the retrieval of medical or satellite image, images of human faces etc. Therefore, it has been an on-going aim for scientist to formalize a general image data model, which can be used for a broad range of application domains. Several domain-independent image data models have been developed in the early years of CBIR – AIR, VIMSYS, EMIR2. Brief overviews of these models are given in [4] and [20]. Another model is the object-oriented approach proposed in the DISIMA project from Oria and Özsu [14], which represents mathematical formalizations for the different levels of abstraction and views of an image. Santini and Gupta [16] propose an extensible feature management engine for image retrieval with an own object-relational database model. Other modeling techniques for image databases are Summary Tree used in the PIQ [19] model and UCDL cited in [17]. One of the numerous existing multimedia conceptual data models (see [8] for an overview) is MPEG7 [1] and it has also been defined for representing image content. An evaluation of the quality of the models with respect to flexibility, completeness, validity, understandability and implementability showed that none of these models provides well defined extensibility and adaptability interfaces for deriving a domain-specific model of a CBIR system.

Therefore, a generic and adaptable conceptual model for image retrieval systems (GiACoMo-IRS), based on the UML modeling paradigm was defined to support the modeling of concrete CBIR systems. The model aims at providing a general base for representing image data structure and retrieval functionality, support the implementation on a large number of platforms, provide explicit mechanisms for extending and adapting the model for domain-specific applications and achieve good understandability through a modeling paradigm, supported by a wide range of modeling tools and a comprehensive tutorial for applying the model for deriving a domain-specific application model. The main reasons for choosing the Unified Modeling Language were: the extensibility of the model, integration possibilities with other models of system components, the support for modeling the system behavior, and last but not least its broad usage in modeling tools. Although the basic concepts of the UML model do not have an extensive support and notations for adaptability and extensibility interfaces, extensions of the UML model, which aim at providing adaptability and extensibility patterns for the design of frameworks are proposed in [3]. The UML-F profile from [3] is used for representing adaptability and extensibility interfaces in GiACoMo-IRS.

The term “framework model” is used here to represent a set of UML classes, relationships between them and operations which provide reusable software architecture and building blocks for deriving domain-specific CBIR system models. In Figure 2 the different abstraction levels for the application of this modeling approach are shown.

The framework level represents the framework model, which instantiates the concepts of the UML metamodel. At the application level a concrete application-specific model for the eNoteHistory Image Retrieval System (IRS) is represented.

2.1 Modeling Data Structures

To begin with the design of the framework model, we turn to the most invariable part of a software application - the data structure. The aim is to obtain an abstract representation of the digital image data and a model which can be used for the content-based retrieval of images. It is important that as many as possible domain-specific representations of image data should fit in (be derivable from) this abstraction.

The generic image data model in [9] determines only the overall structure of the data, but does not give any concrete suggestions for the possible implementation of the abstract classes. Since the role of the framework model is to support the developer of domain-specific applications not only by providing a reusable architecture, but also by predefining reusable building blocks, the usage of concrete implementations of the abstract classes, which can be used in a broad range of CBIR applications, is suggested. These implementations are defined as “black boxes”, in terms of the definition found in [3], which can be used by the developer later through the composition and delegation concepts. In Figure 3 the predefined types of features, image metadata and region relationships of the framework model are shown as “black box” classes. The “white box” framework classes allow the adaptation of the framework through inheritance.

The class StillImage represents digital raster images in general. Specific types of digital images for a particular application can be derived from this class. The problem which has to be solved in GiACoMo-IRS is to make the proposed attributes of the class optional and adaptable, so that they can be freely included or omitted and changed in the domain-specific implementation of the abstract class. Normally the implementation is realized by directly inheriting from the
abstract class, which does not allow any changes on the inherited structure of the derived class. Therefore, since the representation of the raw image of some type is an obligatory attribute of StillImage an abstract class RawImageRep is defined and associated with the StillImage class through a mandatory association. Different types of image representations can be defined for a particular application. The optional attribute Thumbnail can also be regarded as a type of representation of the image. An image can be composed of multiple images, which are interrelated through the aggregation association. This association is optional, which is represented by the multiplicities at its both ends. At this stage no explicit methods are defined in GiACoMo-IRS. The representation of object behavior is described in the following subsection.

Content-independent information, such as creator, creation data etc., is represented by the class Metadata associated directly to the StillImage class to provide means to represent different types of content-independent data. TechnicalMetadata can be regarded as an implementation of this class, which is application dependent.

The representation of the content of an image is based on spatial abstractions, which are derived from the segmentation of the image. Thereby, the content of an image is interpreted on the first place as a set of regions. These regions are represented by the abstract class Region. Each image can contain regions corresponding to segments of an image. These containment possibilities are modeled as an aggregation relationship. Each region can be characterized by its type (circle, ellipse etc.). In GiACoMo-IRS again a decision has to be made which attributes are mandatory and which optional. The type of a region can be regarded as a kind of feature associated with the region. However, for most applications it is necessary to assign some kind of localization information for the region, such as centroid coordinates, bounding box etc. This data is represented as an abstract class RegionLocalization, which is associated by an optional association to the class Region. Application specific regions can be defined by the developer by implementing the abstract class Region. In GiACoMo-IRS the association between regions is not restricted to an aggregation, so that also other than hierarchical organization of regions are possible. An association class Relationship is defined, which can be used to assign the appropriate type of relationship.

A feature can be assigned to each region of an image, whereby the whole image can also be described as a region. Various features can be defined to describe the content of images, by inheriting from the abstract class Feature. These can be low-level features such as height, width, dominant color or color histogram, shape, as well as high-level features, such as names of objects or concepts. Therefore, relationships between features have been defined with an association. These relationships can be used to link low-level features with high-level features for example, when the latter are derived from the first.

The “black box” classes represent examples of application-specific implementations of the abstract classes, which can be used in an application-specific model. The stereotype <<adapt-static>> of the realization association according to the UML-F Profile shows that the abstract classes can be furthermore adapted through subclassing at design-time. In order to keep the resulting model as compact as possible it should be possible to freely omit or exchange the “black box” classes. Therefore, in the current framework model the <<adapt-static>> stereotype means that the examples for the realization of the abstract class are optional for the application-specific model.

The specialization of the framework requires the following steps to be undertaken after the requirements to the application model are determined:

- Define an implementation for the StillImage class and one or more implementations for the RowImageRep class
- Redefine the association between StillImage and RowImageRep class for their implementations
- Optionally redefine the self-association of the StillImage class in its implementation class
- Optionally define one or more implementations of the Metadata class and redefine the association between
Metadata and StillImage for their implementations
• If required one or more implementations for the abstract class Region should be defined. In this case the association between the implementations of the classes Region and StillImage must be redefined.
• Optionally implementations for the associated classes RegionLocalization and Relationship can be defined and their associations with the implementations of the class Region must be redefined.
• Finally implementation classes for the abstract class Feature can be defined and the self-association can be redefined where necessary.

If the associations are needed in the derived classes they have to be redefined in their implementations using the association redefinition capabilities of UML. Association redefinition is a relative new concept in UML. A detailed discussion on association redefinition is given in [22]. Redefinition is more similar to method overriding than to specialization. It is necessary to use this concept because an abstract class is generally a class which is not instantiated and thus no objects of this class and its associations can be created, which can be further specialized. Moreover, in the case of complex abstract class hierarchy it is not straightforward to derive the associations between their subclasses automatically. It depends on the semantic of the subclasses if and how they can be associated to each other.

Attribute Relational Graphs (ARG) and 2D-Strings are two of the most common spatial relationships representations for image regions. An example for deriving an application-specific model of images the content of which is represented by image regions. The ARG is a straightforward to derive the associations between their subclasses automatically. It depends on the semantic of the subclasses if and how they can be associated to each other. In addition to the associations towards the regions of the image, the attributes of an ARG, represented as features, can be related to the ARG relations. Therefore, an additional application specific association between the classes ARGAttribute and ARGLocation had to be inserted. The rest of the ARGs data structure fits well into the framework model. In this diagram the UML notations for association redefinitions are left out to avoid overloading the diagram.

2.2 Modeling Functionality

Integrating retrieval functionality is the second step towards a conceptual image retrieval system model. In this section, the functionality groups, from a level-of-detail point of view are defined, and a general design approach for integrating extensible and adaptable functionality in the model is described. In the design of a CBIR system two kinds of functionality can be distinguished: - from the view point of a system user there is the application (system) functionality and - from the view point of the system developer there is the object functionality, which has to implement or provide the system functionality. The application functionality is modeled at first by means of UML use cases and activity diagrams in order to define the functionality of the application as a whole, which is required by the user. The two groups of operations, which a CBIR system has to support from the view point of a user, are Updates (Insert, Delete, and Update) and Queries. From the view point of a system developer each of these functions has to be integrated into the building blocks of the application. This is achieved by mapping the activities from the detailed activity diagrams onto classes and methods in the class diagram.

The operations from the first group have very similar behavior. Therefore, as an example here only the Insert operation is considered. In Figure 5 the use case diagram of the Insert operation is shown.

Analogously, the use cases of the other update operations are defined. For each class in the current class diagram of the framework, which has to be made persistent in the system there is a corresponding use case in the Insert use case diagram. Each of these use cases are complete by themselves and so can be performed also separately from the others. The integrity constrains for inserting dependent objects such as Metadata, which have to be assigned to a particular Image and cannot exist alone are not represented through the use case diagram, but through the mandatory association with an image object shown in the class diagram. Some use cases may extend others if certain conditions are met. This means that the extending use cases are inserted in a specific point of the extended use case if the condition is true. For example, the regions of an image can be inserted during the insertion of an image object if a variable for segmenting the image <<Segment Image>> is set to true. An image must have at least one image representation, therefore, the Insert RawImageRep use case is mandatory included in the Insert Image use case through the <<include>> dependency. The <<include>> stereotype of the dependency means that the included use case is always included at a certain point of the including use case. An extending or included use case can be invoked more than one times from an extended or including use case, respectively.

In order to provide a more detailed description for the use cases shown in the use case diagram an activity diagram for each use case of an operation is designed. In Figure 6 the activity diagrams for the Insert Image use cases is shown.

This activity diagram includes anchors to the activity diagrams representing the included or extending use cases - Insert RawImageRep, Insert Metadata and Insert Region. From the defined concrete activities (not anchors to other activity diagrams) it is now possible to identify the single methods which have to be supported by the classes in the class diagram in order to integrate the Insert Image functionality in
the model. These methods determine the object behavior. In some cases new classes have to be added to the class diagram of the framework, such as the Image Storage Mechanism, the Segmentation Algorithm classes to represent an interface for certain functionality, which can have different interchangeable implementations in an application. These classes are used to provide the possibility to redefine and dynamically bind different implementations for a particular functionality. In order to enable the integration of different implementations for the segmentation of images depending on the application requirements the concept of template and hook methods is adopted in the class diagram. The same kind of template-hook combination can be used for the extraction of features in order to provide adaptation possibilities also for these methods. The usage of template and hook methods in framework architectures is described in [3].

The query model, also referred to as retrieval model, is created depending on the retrieval task, which has to be realized. We can define different retrieval models for the same data model in the same CBIR System. The retrieval models palette is quite rich. Therefore, we had to make some restrictions for the framework model. First of all, we defined the kinds of queries supported by the framework model as shown in Figure 7. The processing of similarity queries, based on global features, local features, metadata and structure of the images and the combination of these, as well as exact match queries on the latter were considered in the model. Furthermore, the metric approach for image similarity retrieval was chosen as the most used approach for image retrieval. The metric approach is based on comparing the feature representations of images in the database with the one of a query image, using a distance function. As a result the distances representing the degrees of similarity for all the images in the database are returned. The result can be assessed based on a k-nearest neighbor or range metric in order to return only relevant images.

The modeling of Query operations is performed analogously to the Update operations. The use cases are represented as activity diagrams in order to determine the needed methods which have to be supported, and map them to responsible classes. A query image is accepted as input of an image similarity query which is then analyzed to extract its structure in the form of regions and relationships between regions, and its content in terms of features. Additional metadata could be used to support the query processing. In the framework model the functions for the extraction of the feature representation are already defined, to support the Update/Insert of images. These can be used to analyze the query image when formulating the similarity query.

The queries based only on global features or metadata, or only structure can be processed in a straightforward way using a suitable distance function for their comparison. Thus, the framework model requires a distance calculation method in the corresponding classes - Feature, Metadata, and Region respectively. Combined queries and local feature queries, which require the combination of more than one feature and more than one region in the similarity measure, need additional aggregation functions in order to combine the distances from one or more features, or one or more regions. If we consider an image database with a set of images, where each image \( I \) has a set of regions \( R_i \), where \( r_i \) is the \( i \)th region in the image \( I \), which contains \( m \) regions.

\[
R_i = \{ r_i : i = 1.m \}
\]

Regions can be salient objects, image regions with homogeneous texture or color or simply blobs of any shape. Each region is represented by a set of features \( F_{r_i} \), where \( f_k \) is the \( k \)th feature of a region \( r_i \), which has \( s \) associated features with it.

Figure 6. Activity diagram for the insert operation

Figure 7. Use cases for query operations
Different types of features can be associated with an image region, such as color histogram, bounding box of the region, associated names of concepts etc. If a query image $Q$ is given, then the query would also be represented by the set of regions of the query image and their corresponding features:

$$R_Q = \{r_j : j = 1..n\}, F_Q = \{f_i^Q : l = 1..t\}$$

In order to find all similar images of the query image in the database the query image has to be compared with each database image and the similarity or the distance between the query image and the database image has to be derived. The distance $D$ between the two images $I$ and $Q$ can be represented by the distance or similarity of their region sets $R_I$ and $R_Q$, respectively as follows:

$$D(I, Q) = D(R_I, R_Q)$$

Where the distance between the two sets of region can be represented by an aggregate function $f$ on the distances $d$ between the feature sets, representing the single regions:

$$D(R_I, R_Q) = f\left(d\left(F_I, F_Q\right)\right)$$

The function $f$ has to combine the distances between multiple regions of the query and the target image. Depending on the application this function can have different semantics, e.g., - it can simply check if for each region of the query image there exists a region in the target image where $d(F_i^Q, F_j^Q) \leq \epsilon$, or - calculate the needed transformations, which need to be performed on the query regions to receive the target regions. In this function additional factors can be considered, such as the weights of the regions, the number of regions in the query and in the target image. Spatial relationships between regions, reflecting the structure of the image can be also integrated in the similarity measure.

The distance between a region $i$ of the query image and a region $j$ of the target image is an accumulating function $g$ on the basic distances between the single features $f_k^i$, associated to the region $i$ in image $I$ and $f_k^j$ to region $j$ from image $Q$:

$$d\left(F_i^Q, F_j^Q\right) = g\left(\delta_{f_k^i, f_k^j}\right)$$

The function $g$ has to accumulate the distances between different types of features. If the different types of features can be represented in the same feature space, then the function $g$ can be the weighted sum of all single feature distances. The function $g$ represents the distance between two regions in terms of their features. The basic function $\delta$ represents a distance function for a feature space. It is defined only for feature values of the same feature type.

In Figure 8 the integration of the metric approach methods and classes for the “Query images by local features” case in the framework model is shown. The class diagram depicts also the classes and methods, responsible for the image segmentation and feature extraction. The feature specific distance functions are defined as methods of the Feature classes, on which they are applied. The aggregate distance functions for combining the distance measures of different features or regions are defined in the parent class in the image data structure hierarchy. Since different implementations of distance functions exist, e.g., Manhattan, Euclidean, Hamming distance etc., in this case, also the template-hook methods are used to define the distance measure functions. This approach requires overriding the hook methods or implementing their classes to provide adaptation possibilities. The combination of features or other characteristics of the images to perform a query requires the introduction of weights for the participating partial queries. In many applications these weights are empirically predetermined, but they can also be acquired from the user during the query formulation process. Therefore, we leave these outside the framework model.

There are two mechanisms to adapt the functionality of the framework. Firstly, the methods of the abstract classes can be redefined in their subclasses and secondly the hook methods can be redefined (implemented) by subclassing the hook classes. The first method is carried out on the result of the data structure adaptation - the derived concrete classes can override the abstract methods, and the second method requires deriving new classes from the hook classes and redefining the required methods. Which one of the

![Figure 8. Class diagram of the “image insertion” and “query by local features” classes and methods](image)
methods should be used depends on whether the application should support different algorithms for the same functionality which should be interchangeable in the application.

In Figure 9 the adaptation of the framework to support a concrete Insert Image functionality is shown. Only one storage algorithm for images and raw image representations, respectively, should be defined in the application therefore the adaptation of the storeImage() and storeRawImageRep() function is done by overriding the methods in the subclasses of StillImage and RawImageRep. The Unification pattern is applied here for adapting the functionality. The adaptation of the segmentImage() function is done based on the Separation pattern. For each hook class of the segmentation algorithms a specialization in the application domain is defined and the association between the hook-class and the template class is redefined in the application domain. Different implementations of the hook-method for image segmentation are provided by subclassing the hook class.

The behavior of the objects until now referred only to the implementation of the behavior of the system. It is expressed through the operations of classes. However, we have to note, that there are no means to insure the validity of the two behavior models (system behavior and object behavior) automatically, because there are many possible ways to realize the system behavior through the class methods.

Apart from application specific behavior objects need to implement some implicit behavior, which determine their lifecycle. Generally these methods comprise of (see also [5] Chapter 6): constructor/destructor, identity, equality (shallow, deep), assignment, copy (shallow, deep), equals. For each class in our model these operations are defined implicitly.

Now that the basic data structure and functionality of a CBIR system in the conceptual framework model, which can be adapted and extended with the terms of the UML model for a specific CBIR application are determined, we can turn to the next step of the model-driven development - the mapping of the conceptual model onto an implementation platform.

3. Mapping onto an Implementation Model

A domain-specific model derived from the framework model should be implementable on any specific implementation platform. We consider the implementation onto an object-relational database management system (ORDBMS), based on the standard SQL:2003 [18]. Our choice in favor of this environment was made in order to demonstrate the possibility for image database developers to design and implement customized database extensions for storing and querying images by content in ORDBMS. We regard existing database image extensions (e.g. IBM AIV-Extenders) as CBIR applications, belonging to the first group of CBIR systems, mentioned in section 1 - generic systems. Hence, we do not have the possibility to use or adapt these extensions for a specific application domain. Furthermore, the creation of a conceptual model and mapping it onto a specific implementation model such as the relational model are standard steps in the design of database systems. Altogether, we consider the implementation onto an ORDBMS as an adequate example and test case for the developed concepts.

A mapping mechanism for the conceptual model has to be defined to assure a consistent implementation. Since our generic model has been built using UML constructs a suitable mapping of the UML concepts onto the ORDBMS model has to be defined. For the mapping of UML class diagrams onto object-relational models and specific database management system models an informal methodology has been proposed in [12], which has been applied to formulate graph-based transformation rules. These rules can be used to automate the transformation of UML into SQL:2003. This methodology, however, is not exhaustive and needs to be extended to support the transformation of operations for example. At the current stage of our research we have elaborated mapping rules for the UML class diagrams and are working on their formalization and implementation. However, a fully automatic mapping mechanism cannot be defined. The developer has to support the mapping process by making certain decisions or performing adaptations to the models by hand. We distinguish three kinds of mappings:

- Not-mappable concepts: not all of the concepts defined in the conceptual model can be mapped directly onto the object-relational model (e.g. attribute properties, such as private, public). These elements are thus omitted during the mapping, but the developer should be informed about that.
- Multiple mapping possibilities: Sometimes there are multiple ways to represent conceptual elements in the logical model (e.g. represent methods as user-defined functions or stored procedures). This requires the developer’s decision and/or the usage of default values during the mapping process.
- Implementation specific concepts: Some concepts
from the logical model cannot be represented in the conceptual model. This requires further adaptation of the logical model by the developer. There are different ways to accomplish this: e.g. map the UML conceptual model onto a DDL Script as far as possible and perform all further adaptation in the script. Another possibility is to create a UML Profile for representing object-relational logical models and map the conceptual UML model onto another UML model, representing the logical model for an object-relational database and perform all the further adaptations in the logical UML model.

The classes from the conceptual model, which need to be made persistent by a StorageMechanism, which in this case is the ORDBMS, are mapped onto user-defined types and corresponding typed tables. The associations between these classes are mapped onto integrity constraints in the database. The classes, which do not require persistence, do not need to be represented through typed tables. Generalization is also supported by the SQL:2003 standard.

The mapping of behavior is divided again into mapping of system behavior and mapping of object behavior. The object behavior is represented in SQL:2003 through the methods of user-defined types. The signature and body of these methods are separated in SQL:2003. In UML only the signature of an operation is given in a class diagram. We consider providing the implementation of the methods directly in the programming language, e.g. Java. In Figure 10 on the left side the UML diagram class and method are shown and on the right side the translation into SQL:2003.

During the mapping process the declaration of the method in the relational model is based on the signature of the method in the UML class diagram. The implementation of the method can be added to the database user-defined functions either from a predefined library of the framework model or from a customized implementation of the developer. For mapping the system behavior we have to use the interface provided by the ORDBMS in terms of SQL data manipulation and data query language and the extensibility options in terms of user-defined functions and stored procedures. There is more than one possibility to realize the Insert operation for images. One possibility is to encapsulate the extraction of the regions and features and their insertion into the database in a user-defined function, representing the constructor of a StillImage object. Another possibility is to make use of the TRIGGER object in an ORDBMS to invoke the region and feature constructors for extracting the regions and features of an inserted image. In both cases the following SQL statement should be issued to insert an object of the type StillImage into the database:

[Oracle syntax:]  
INSERT INTO SCHEMA.IMAGE  
VALUES (StillImage(URL))

The efficient processing of queries is one of the main advantages of database management systems. The query processing operations supported by the DBMS are generic and therefore domain independent. They are global for the whole system. However, they have been defined to apply them on standard basic data types (alphanumeric data types). In order to support application specific queries such as the similarity queries on the StillImage data type we have to provide special operations for comparing object of this type. Once again we can map the methods realizing the query operation from the UML class diagram onto methods of user-defined types in the database. The possible usage of these methods in the SQL query can be as follows:

[SQL:2003 syntax:]  
SELECT *  
FROM getSimilarImages(URL, threshold)

With this implementation the whole query processing algorithm is encapsulated into the user-defined function getSimilarImages(...). The main disadvantage of this approach is that it does not allow for any query optimization on behalf of the DBMS. Therefore, we suggest the usage of clustering methods for building predefined clusters of similar images by specific features or a combination of those. Hence, the user-defined function for comparing the query image with the ones in the database would have to be applied on the first place only onto the cluster representatives of the images.

In this way we can improve the efficiency for the processing of the similarity queries.

For a simple CBIR system model basic mapping mechanisms, such as class-to-table, attribute-to-column, were implemented as a plug-in of IBM Rational Rose Data Modeller1. The disadvantage of this implementation was the missing possibility to seamlessly add particular implementations for operations to be used in the resulting source code. Therefore, the aim which we further pursue is a specialized application for the model-driven development of CBIR systems on top of ORDBMS. This application shall be realized as an Eclipse IDE Plug-in and should provide a complete model-driven development workflow for CBIR. The Eclipse Modeling Framework2 will be used to implement the modeling components of the application comprising a conceptual modeling step with GIACoMo – IRS as a starting point followed by a model-to-model mapping to generate an ORDBMS specific model of CBIR, which can be refined manually in the modeling environment. The ORDBMS specific model can then be forwarded to the code generators, which shall implement the creation of source code based on the model. And finally the generated source code can be further refined in the Eclipse by the developer to add for example the needed operation algorithms.

4. Applying the Model-Driven Techniques for the Development of the eNoteHistory CBIR System

In order to verify the applicability of the proposed techniques for the development of different domain-specific CBIR applications an extensive set of example applications should be considered. This process requires the availability of the automated tools. Up to now we have applied these techniques manually for the modeling of the eNoteHistory CBIR application for writer identification in music manuscripts.

The application steps involved in the automatic handwriting analysis and content-based retrieval are carried out in the following order. At first, for all digital scores in the database, for which information about the scribe (e.g., name of scribe) exists, image processing algorithms are applied to extract the visual features of the images, representing the handwriting characteristics. **Figure 11** shows the recognized objects in the manuscript. For each recognized object: note heads, note stems, bar lines a set of geometrical features is extracted, such as: height and width of the bounding box, radius of the bounding ellipse, x, y coordinates of the centroids, orientation etc. The handwriting characteristics of scores with unknown scribes can subsequently be compared with the set of extracted features in the database and using distance metrics for calculating the similarity between features a query result of the type: a list of k-most similar scores with associated scribes can be generated.

For this database application we have derived the model shown on **Figure 12** from the framework model. The eNoteImage class represents the set of scanned page images of music manuscripts. In addition to the TechnicalMetadata a class Image LibraryMetadata has been defined, which was derived from the abstract class Metadata. The associations between the eNoteImage and LibraryMetadata and TechnicalMetadata, respectively, have been redefined. The names of the redefined associations, such as “has metadata” are left unchanged in order to recognize easier that they are redefined. Two classes of raw image representations are defined for the eNoteImage. The multiplicities of the associations “has raw imagerep” can be redefined to allow only one eNoteRawImage for an eNoteImage. Two types of regions are derived from the class Region, ROI (Region of Interest) and MusicObject, which is further specialized in NoteStem, NoteHead, and Clef. The latter represent region, which have been identified as the corresponding music elements. Unidentified regions can be stored as MusicObjects. A Region of Interest is the region of the digital image which contains only the relevant information – the staff lines without the edge of the paper and notes at the corners of the page, such as page number. A ROI contains the music objects as shown by the redefined “related to” association. Furthermore music elements can have directional relationships, which can be used for example to identify if a note head belongs to a note stem. The localization information for a ROI is represented in a separate class and for a music element inside the MusicObject class. The class eNoteFeature represents the set of features used to describe the regions of
an eNoteImage. For the current application only a shape descriptor for the regions of music elements is applied to compare their similarity.

The operations, defined in the model are intended to be used on one side to create the data, which has to be derived from the image by segmentation or feature extraction and on the other side to support similarity queries on the images, by providing a distance function for the features representing the content of the images. The store operations should implement a storage mechanism for making the corresponding instance persistent if no such mechanism is provided by the platform. The operation `segmentImage()` can be used to create the instances of regions for a specific image. For extracting the features of a region the corresponding feature extraction function of a feature should be implemented. The retrieval functionality is based here on the metric approach and the comparison is done based on local feature similarity. Therefore, for each feature type a `compareWith(Feature)` operation should be implemented. The accumulated distance function of the feature distances should be calculated by the `compareByFeaturesWith(anotherRegion)` operation. And finally the aggregation operation for calculating the similarity between the whole images should be provided in the `compareByLocalFeaturesWith(anotherImage)` function.

The instantiation of the GIACoMo-IRS model for eNoteHistory was carried out by reverse engineering the existing implementation on top of an ORDBMS. Therefore, it can be assumed that the conceptual model is also implementable onto an ORDBMS system. This particular CBIR application is also a good example for the need of diverse multiple use interfaces of such systems. Music scientists are one type of user for the system. They require the score identification functionality in order to derive or prove theories about the origin of historical manuscripts. On the other hand there are the librarians, who require the possibility to view and edit the bibliographical or physical metadata of the digital manuscripts. A third group of users are musicians, who need the music manuscripts to adapt them for a performance. In order to satisfy these needs a web-based client application for the eNoteHistory CBIR system was implemented [2], which offers different functionality for different user groups. The model-driven development of these CBIR graphical user-interfaces would require additional models to be provided.

4. Summary and Future Work

In this paper, the investigations on the idea of employing a conceptual framework model - GIACoMo-IRS - as the basis for a model-driven development approach for CBIR systems was represented. The Universal Modeling Language was employed for elaborating an abstract representation of still image data and its content, as well as for modeling extensible system and object operations of a CBIR system. The eNoteHistory CBIR system for scribe identification in music manuscripts was instantiated from the framework model to verify the applicability of the framework model. Techniques for the generation of an implementation of the concrete CBIR model on top of an ORDBMS were presented. These techniques have been developed only conceptually and therefore extensive tests for the applicability for the development of arbitrary CBIR systems have not been carried out until now. A future aim is to implement a specialized model-driven development tool as an Eclipse Plug-In to support the modeling and generation of domain-specific CBIR systems.

For the design of the query operations we have chosen to integrate the metric similarity retrieval approach in the framework model. Further the possibilities to model the classification of images according to their content have to be investigated. Classification enables the assignment of objects (e.g., images), respectively instances (e.g., features) to different predefined classes (e.g., concepts). Thereby, the input instances have to be already assigned to specific classes, which is represented by a special nominal attribute of the instances, called the class attribute. The aim of such data mining algorithm is the development of a model, which can be used to assign a value to the class attribute of a new instance. In order to model the classification of images we need to include additional data structures and corresponding methods in the framework model, such as classes for representing cleaned training and test data sets, classes for defining interfaces for the creation of the data mining model, testing the model and the classification of new instances, as well as attributes for storing the model and its training and test instances.

For the design of the UML-framework model some of the concepts for modeling extensibility, defined in the UML Profile for framework architectures (UML-F Profile) [3] have been used. There is further potential for refining the notations of the framework model and for preparing a manual for the adaptation of the framework model for domain-specific applications, in order to provide more compliance with the UML-F Profile and to insure an efficient usage of the framework, respectively.

As an implementation platform we have chosen an ORDBMS, which however, does not restrict the possibility to use other platforms for the realization of the CBIR system.

References

Descriptions. In IEEE Transactions on Neural Networks, Special Issue on Intelligent Multimedia Processing 13, pages 841–853.


