Modelibra Education

Dzenan Ridjanovic
Université Laval
2325 rue de la terrasse
Québec (Québec) G1V 0A6 CANADA
dzenanr [at] gmail [dot] com

Abstract - Education in information systems requires at least an exploration of alternatives in teaching software development. The following phrase is from the Free Software Foundation [1]: "To use free software is to make a political and ethical choice asserting the right to learn, and share what we learn with others." At first, it is not obvious how this freedom could be applied to education. In software, services around free software [2] provide an added value that customers are ready to pay for. In education, free course material has a potential of free software. Free course material supports the right to learn and share what we learn with others. However, educational services, such as teaching courses, may hopefully provide added values.

In order to provide some examples, I will focus in this paper on using free software in education of students in information systems. I will start with the issue of providing some basic development skills to new Web generations of students. In order to break from the old ways of educating, the free course material is spread over the Web with a base at the Assembla [3] development environment.

I. MODELIBRA

I teach courses in dynamic web development. A dynamic web page is created at a server from a static template and some data obtained dynamically (at the time of the page request) from a data store. A data store is usually a database. Consequently, a database knowledge is a often a prerequisite for a web course. Since I teach web courses to students of information systems at a business school, I have tried to avoid a lengthy sequence of programming and database courses before teaching web courses. With that in mind I have developed Modelibra [4] family of free software that is used to develop dynamic web applications based on domain models.

Modelibra is an open source software family of tools and frameworks to support domain-driven development [5] in Java [6]: a graphical tool for model design and code generation, a domain model framework, and a web component framework based on Wicket [7] for rapid development of dynamic web applications. Modelibra is hosted at Google Code [8].

In computer terms, a domain model [9] is a model of specific domain classes that describe the core data and their behavior. From an organizational perspective, a domain model is a model of the domain. Within this domain, an organization conducts its business. The memory of any organization may be represented conceptually as a domain model.

The heart of any software, including a dynamic web application, is a domain model. When a model is well designed and when it can be easily represented and managed in an object oriented language, a developer may focus on views of the software and they are what users care about the most.

Before diving into programming, I spend quite some time showing students how to design domain models. In the process of designing domain models I use ModelibraModeler, a graphical client tool that is capable of generating an XML [10] configuration of the model. This XML configuration is used to show students that a model may have different representations. A graphical representation of entities (concepts in Modelibra) with attributes (properties in Modelibra) and relationships (neighbors in Modelibra) between entities is suitable for the design of a domain model. An XML configuration is a better representation if we want to introduce the model to different software tools.

In Modelibra, an XML configuration of a domain model may be transformed, without programming, into XML data files and Java classes. XML data files are used as a data store, and Java classes are generated to add, remove, update and retrieve data in a domain model.

The code generated from a domain model is used to explain students the basics of object oriented programming in Java. A concept is represented in Modelibra as two POJO (Plain Old Java Objects) [11] classes. One extends the generic Entity class and the other extends the generic Entities class. Both generic classes are part of Modelibra and may be reused in different domains. For example, the WebLink concept has two specific classes: WebLink and WebLinks. The WebLink class describes the web link concept and the WebLinks class represents available web links.

The generic Entity class implements the IEntity interface, and the generic Entities class implements the IEntities interface. The easiest way to get a feeling about what Modelibra offers is to look at its main Java interfaces. An interface is a group of related methods without the implementation code. What is left in a method without its

implementation is called a method signature. The three main Modelibra interfaces are: IDomainModel, IEntity and IEntities.

An entity belongs to its model (Figure 1). It has a configuration. An entity must have an artificial identifier called oid. An entity may have at most one user oriented identifier (id) that can be obtained by the getUniqueCombination method. If the index is configured to be used, there is a unique index for oid and a unique index for the id combination of properties and/or neighbors. In addition a non-unique index combination may be configured. Properties and neighbors of an entity may be updated by the update method. Properties of an entity may be updated by the updateProperties method. An entity may be copied. Only properties of an entity may be copied. A deep copy of an entity may be made by copying the internal tree that starts with the entity as the root of a tree of entities and neighbors. An entity may be compared with another entity to verify if they are equal based on different criteria. There are set and get methods for properties, parent neighbors and child neighbours.

```
public interface IEntity<T extends IEntity> extends Serializable,
    Comparable<T>, ISelectable<T>
{
    public IDomainModel getModel();
    public ConceptConfig getConceptConfig();
    public void setOid(Oid oid);
    public Oid getOid();
    public UniqueCombination getUniqueCombination();
    public IndexCombination getIndexCombination();
    public boolean update(T entity);
    public boolean updateProperties(T entity);
    public T copy();
    public T copyProperties();
    public T deepCopy(IDomainModel model);
    public boolean equalOid(T entity);
    public boolean equalUnique(T entity);
    public boolean equalProperties(T entity);
    public boolean equalParentNeighbors(T entity);
    public boolean equalContent(T entity);
    public void setProperty(String propertyCode, Object property);
    public Object getProperty(String propertyCode);
    public void setParentNeighbor(String neighborCode, IEntity<?>
      neighborEntity);
    public IEntity<?> getParentNeighbor(String neighborCode);
    public void setChildNeighbor(String neighborCode, IEntities<?>
      neighborEntities);
    public IEntities<?> getChildNeighbor(String neighborCode);
}
```

Figure 1

A collection of entities belongs to its model (Figure 2). Entities have a configuration. An iterator over entities may be defined. The number of entities determines its size. If the size is zero, entities are empty. An entity may be added to entities. An entity may be removed from entities. An entity that belongs to entities may be updated by using another entity. Properties of an entity may be updated by using another entity. It can be verified if entities contain an entity. An entity may be retrieved by its oid, by the unique combination, by the index, or by a property and its value. A subset of entities may be selected by a method that returns true when applied to an entity, by an index, by a property based on its value, by a parent neighbor, or by a selector for a more elaborate selection. Entities may be ordered by a property, or by a more elaborate comparator. A union or an intersection may be made out of two collections of entities. It can be verified if one collection of entities is a subset of another collection of entities. In Modelibra, a selection of entities or an order of entities produces a new collection of destination entities. The source entities are reachable from the destination entities. In Modelibra, it is preferable to work directly with entities. However, a Java list of entities may always be obtained. There are positional methods to find the first and last entities, to find the next and the prior entity based on a given entity, to locate an entity based on its position. A random entity may also be returned. Entities have a corresponding collection of errors. A copy or a deep copy of entities may be made. Entities may be exported, synchronized and cleaned.

```
public interface IEntities<T extends IEntity> extends Serializable,
    Iterable<T>
{
    public IDomainModel getModel();
    public ConceptConfig getConceptConfig();
    public Iterator<T> iterator();
    public int size();
    public boolean isEmpty();
    public boolean add(T entity);
    public boolean remove(T entity);
    public boolean update(T entity, T updateEntity);
    public boolean updateProperties(T entity, T updateEntity);
    public boolean contain(T entity);
    public T retrieveByOid(Oid oid);
    public T retrieveByUnique(UniqueCombination
      uniqueCombination);
    public T retrieveByIndex(IndexCombination indexCombination);
    public T retrieveByProperty(String propertyCode, Object
      property);
    public IEntities<T> selectByMethod(String
      entitySelectMethodName, List parameterList);
    public IEntities<T> selectByIndex(IndexCombination
      indexCombination);
    public IEntities<T> selectByProperty(String propertyCode, Object
       property);
    public IEntities<T> selectByParentNeighbor(String
      neighborCode, IEntity<?> neighbor);
    public IEntities<T> selectBySelector(ISelector selector);
    public IEntities<T> orderByProperty(String
      propertyCode, boolean ascending);
    public IEntities<T> orderByComparator(Comparator comparator,
      boolean ascending);
    public IEntities<T> union(IEntities<T> entities);
    public IEntities<T> intersection(IEntities<T> entities);
    public boolean isSubsetOf(IEntities<T> entities);
    public IEntities<T> getSourceEntities();
    public List<T> getList();
    public T first();
    public T last();
    public T next(T entity);
    public T prior(T entity);
```

```
    public T locate(int position);
    public T random();
    public Errors getErrors();
    public IEntities<T> copy();
    public IEntities<T> deepCopy(IDomainModel model);
    public void export(IEntities<T>
       takenEntities, boolean exportSensitive);
    public void synchronize(IEntities<T> takenEntities,
       IEntities<T> returnedEntities, boolean synchronizeSensitive);
    public void clean(IEntities<T> takenEntities, IEntities<T>
       returnedEntities);
}
```

Figure 2

A selection of entities is a good example of using data in a domain model. There are two different ways of selecting entities. A single entity may be retrieved by one of methods in the IEntities interface whose names start with the retrieveBy prefix. A subset of entities may be selected by one of methods in the IEntities interface whose names start with the selectBy prefix. A retrieve method retrieves the first entity that satisfies a retrieval expression. If no entity is retrieved, **null** is returned. A select method selects a subset of entities that satisfies a selection expression. If none of entities is selected, the empty entities object is returned.

For the WebLink concept in a domain model, the method in Figure 3 retrieves at most one web link.

```
public WebLink getWebLink(String propertyCode, Object property)
{
   return retrieveByProperty(propertyCode, property);
}
```

Figure 3

The method accepts a property name (code in Modelibra) and a retrieval value (object) for that property.

The getWebLinkByName method in Figure 4 is a convenience method for the name property. It uses the getWebLink method, which in turn uses the retrieveByProperty method. The retrieveByProperty method is the public method in the IEntities interface of Modelibra.

```
public WebLink getWebLinkByName(String name)
{
   return getWebLink("name", name);
}
```

Figure 4

The getWebLinks method in Figure 5 is a convenience method for selecting web links based on a given property code and a property value. The selected web links represent the destination entities, while the current object represents the source entities.

```
public WebLinks getWebLinks(String propertyCode, Object
   property)
{
   return (WebLinks) selectByProperty(propertyCode, property);
}
```

Figure 5

If none of the source entities is selected, the destination object is empty.

There are many other possible selections of entities based on a single property that use a relational operator other than equal. For those selections of entities a property selector is used.

Modelibra provides a rich generic interface to a domain model. However, in a few lines of code, convenience methods for a specific model may be prepared.

Students in information systems aim to become one day managers. It is essential for future managers of developers to understand an importance of a high level interface of a framework such as Modelibra for the development productivity.

II. Education

I teach courses in web development to students in information systems. I use the Modelibra family of software to show students that a dynamic web application may be developed in relatively short period of time, once a domain is well understood and a domain model is designed. Students in information systems do not have a luxury of many technical courses. They have to understand the basics of software development in few courses. In order to limit the scope of technologies used, I have focused only on web development based on a domain model by using advanced tools and frameworks found in Modelibra.

In addition, it is essential for future managers of software developers to understand the basis of modern software development including simple project management guidelines. Today, web developers work in virtual teams cooperating on the Web. The core of development activities is a code repository with version management, such as Subversion [12], which is centralized on a server, or distributed on clients and servers, such as Git [13] and Mercurial [14]. In my opinion, it is easier to start with Subversion and use Git or Mercurial in advanced courses.

In my courses, I use the spiral approach [15] to software development. Learning new software concepts and technologies is a challenging task. Learning in spirals, from simple to more advanced concepts but with concrete software, helps students get a reasonable confidence level early on, and motivates them to learn by providing more useful application software. With each new spiral, the project grows and new concepts are introduced. A new spiral is explained with respect to the previous one. The difference between two consecutive spirals is that the next spiral has the new code introduced and the old code modified or deleted. This is called

learning by anchoring to what we already understand. With a new spiral, we can come back to what we did previously and improve it. In this way, learning in spirals can touch the same topic several times, but each time with more details in a better version.

Students are divided in teams and each team has a task to develop a different software application. For example, I teach a course called "Developing Domain Models with Modelibra". The course has its own space [16] at Assembla. The objective of the course is to teach students how to transform a design in the form of a graphical domain model into a collection of Java classes that provide update and search actions of data objects of the model.

The course [17] given in the Winter semester of 2009 has its own Wiki page at Assembla, where the course content is presented. The course uses a book [18] on Modelibra that is freely available to everybody through a Creative Commons [19] license. The book is also hosted at Google Code. The book has a sequence of spirals, one per chapter.

The course starts with the installation of Java and Eclipse [20]. Eclipse is an integrated development environment [21] that requires at least a JRE (Java Runtime Environment) on a client computer. Eclipse is used to connect a client with a Subversion server at Assembla. This connection is enabled by the Eclipse plugin called Subclipse [22]. Each project is stored in Subversion as an Eclipse project. This provides a fast access to student projects to a professor and to students. In this way, there are no zip files to send to a professor. Each team develops a different domain model in a fashion similar to the example from the book. Students are encouraged to study spirals from the book and, and in the spirit of open source software [23], even consult spirals of other teams. Hence, there is no need for any privacy. Since an Eclipse project exists both locally and remotely, there is no need for any special backups. In addition, a history of a project and its changes as spirals may be easily traced.

Assembla provides tools and services for accelerating software development. Its philosophy [24] is inspired by free software. In concrete terms, Assembla manages collaborative work-spaces (or spaces in short) for agile teams of developers. A developer, including a student, may become a member. A member may create multiple spaces. As long as spaces are public they are free. In order to finance its organization, Assembla also provides private spaces as paid services.

After registering, a student may spend some time exploring Assembla with the orientation [25] help. A team of students may create a separate public space and invite team members to join the public space. At least one student must be an owner of the space. A professor may be invited as a team member or even as one of owners in order to delete the space after the course is finished.

Using Eclipse properly is crucial for a good start of the course. Eclipse may be used to create a project with documents as it is done with the Modelibra book. This means, that students may start using both Eclipse and Subversion without any knowledge of programming. A series of free videos [26], hosted at SourceForge [27], about Eclipse and Java is a required "lecture" for students. The videos are done professionally and have a high pedagogical value. This is an example of course material contributed by a professional developer and not an academic.

Once, students feel comfortable with Eclipse and Subversion, they are introduced to basic Java programming [28] with Eclipse through another series of videos by the same author. The advantage of this approach is that students can focus on learning programming concepts, while Eclipse handles many syntax details. In order to liberate a professor from providing constant help to students, the basic debugging [29] of Java programming with JUnit [30] within Eclipse is introduced through the last series of videos.

It is essential that students may ask questions when they face an obstacle. Nabble [31] is a place where a professor may create a free forum for his course. In addition to forums, Nabble includes user groups, message boards, mailing lists, photo galleries, newspapers, blogs, etc.

Before the team work on a domain model begins, the Scrum [32] approach to project management is introduced. Scrum is a loose set of guidelines that govern the development process of a software product. Its main values are derived from the Agile [33] values of software development. Assembla provides a Scrum report, where each member for each spiral reports three points:

1. what I did;
2. what I am going to do;
3. what are my obstacles, and what do I need help with.

All this learning effort prepares students to work in teams on their own choice of a domain model with advanced technologies and modern software. Their work is public and easily traceable.

III. Conclusion

The Modelibra free software family consists of a graphical design tool, a domain model framework, a web component framework, and code generators for XML configurations and Java classes. Modelibra facilitates the definition and the use of domain models in Java. It uses Wicket for application views of domain models. Wicket is a web framework that provides web components to construct, in an object oriented way, web concepts, such as web pages and page sections. Modelibra interprets the application model and makes it alive as a default

web application, which may help developers validate and consequently refine the domain model. In addition, Modelibra has a collection of generic web components that may be easily reused in professional web applications to display or update entities.

The educational objective of Modelibra is to involve students to work in teams over the Web by using collaboration technologies. All software and documents produced in a project are kept freely on a server. Software versions are maintained in a Subversion repository. Project members use a Subversion plugin in Eclipse to share the code. Different web technologies are used freely to communicate among team members, to collaborate on common tasks, and to give presentations to other team members.

For students, free software is important for multiple reasons. Most of it is legally free. They can create an exciting development environment on their personal computers. There are many open source projects that can be an incredible source of learning. A student can open the software code and see how programmers have organized it, in a similar way that a student of literature learns the most by reading classics. An advanced student may even a join an open source project, which is the best way for a young software developer to get an international recognition. For companies, organizations, governments and students in developing countries, free software provides a way to prevent the widespread illegal copying of software, and an opportunity to raise the level of software development to international standards.

What are advantages of providing course material for free on the Web? The main advantage is in improving quality of the material, in the same way that quality of free software is improved by both using and developing software by many interested people. As it is not easy to become an active developer of free software, the same may be applied to the development of course material. A software developer does not need a high degree to become a useful team member. If he is active in helping software to become better, a team may promote him in a regular member. There are so many skillful professionals in information systems that may potentially contribute to production of shareable course material.

In the spirit of free and open source software, I would like to organize development of new courses in teams by using cloud computing [34]. Good candidates are a course in Python [35] programming and a course in Google Application Engine [36]. All documents and software used in such a course would be public and free. Anyone teaching similar concepts would be invited to use freely the course material. Those who are interested in contributing results of their work to improve the existing material and introduce new one are welcome to contact me by sending an email to dzenanr [at] gmail [dot] com.

REFERENCES

[1] http://www.fsf.org/about/what-is-free-software
[2] http://en.wikipedia.org/wiki/History_of_free_software
[3] http://www.assembla.com/
[4] http://www.modelibra.org/
[5] http://domaindrivendesign.org/resources/what_is_ddd
[6] http://java.com/en/
[7] http://wicket.apache.org/
[8] http://code.google.com/p/modelibra/
[9] http://en.wikipedia.org/wiki/Domain_model
[10] http://en.wikipedia.org/wiki/XML
[11] http://en.wikipedia.org/wiki/Plain_Old_Java_Object
[12] http://subversion.apache.org/
[13] http://git-scm.com/
[14] http://mercurial.selenic.com/
[15] http://en.wikipedia.org/wiki/Spiral_approach
[16] http://www.assembla.com/wiki/show/ul-fsa-model
[17] http://www.assembla.com/wiki/show/ul-fsa-model/DomainModels2009
[18] https://docs.google.com/View?id=ddcvsc47_769gzmnfjht
[19] http://creativecommons.org/
[20] http://www.eclipse.org/
[21] http://en.wikipedia.org/wiki/Integrated_development_environment
[22] http://subclipse.tigris.org/
[23] http://en.wikipedia.org/wiki/Open_source
[24] http://www.assembla.com/about
[25] https://www.assembla.com/user/orientation
[26] http://eclipsetutorial.sourceforge.net/workbench.html
[27] http://sourceforge.net/
[28] http://eclipsetutorial.sourceforge.net/totalbeginner.html
[29] http://eclipsetutorial.sourceforge.net/debugger.html
[30] http://www.junit.org/
[31] http://www.nabble.com/
[32] http://www.codeproject.com/KB/architecture/scrum.aspx
[33] http://en.wikipedia.org/wiki/Agile_software_development
[34] http://en.wikipedia.org/wiki/Cloud_computing
[35] http://www.python.org/
[36] http://code.google.com/appengine/

Ridjanovic, D., "Modelibra Education", *The 34th International Convention on Information and Communication Technology, Electronics and Microelectronics, Computing Sciences and Software Engineering*, MIPRO 2011, Opatija, Croatia, http://www.mipro.hr/MIPRO2011/ELink.aspx, May 23 - 27, 2011.