University of Oslo
Department of Informatics

Database Solutions for Biological Systems

Michał Stefanczak
<michalst@ifi.uio.no>

michalst

31st October 2004

Contents

1 Introduction

In the recent years we have seen a rapid increase of different internet services. From simple web pages, through search engines to complex distributed systems (e.g. the interactive book store Amazone.com). All this is possible mainly due to reduced hardware costs, better bandwidth, reliable software and new and better standards. This also gives biological and chemical experts the ability to share and find new resources for their research. A biological model repository is such a resource that can be shared and used by different biological and chemical scientists. It is a middleware that enables exchange of biological and chemical data, and has the ability of both platform and language independence. A scientist can use this repository from different platforms, different client programming languages and from a web interface. This gives many new opportunities and enables broader knowledge of their research field.
This thesis will present three solutions of a scientific distributed system. It will be called the 'Biological Model Repository' (BMR), and will act as a repository for biological and chemical models. One implementation is available to downlaod on my homepage at the University of Oslo department of Computer Science [34]. Not all functionalities described in this document are implemented, only some essential ones. All models stored in this solutions are on the Systems Biology Markup Language format (level 2 version 1), it is a language defined in eXtensible Markup Language (XML) and presented in one of the chapters below. This solutions is meant as an example on a model repository and need far more testing than has been done before it can be used as a commercial package. However, this is a good starting point if someone desires to develop this kind of system.

2 Description of this thesis

A simple chapter that describes different chapters in this thesis, and how everything is organized.
Chapter 1 is the introduction, chapter 2 is this chapter. Chapters 3, 4, 5 are describing XML and languages defined in XML. Chapters 6 and 7 are describing different technologies. Chapter 8 is about the BMR application, while the last chapter is a conclusion. At the end of chapters 6 and 7 I have written a short section which tells how are the technologies presented in these chapters related to my application.
Words in *Italic* are technologies, specifications or other things that are explained in the Appendix A. The rest of the appendixes are showing SBML models, code examples, configuration files, database tables and a readme file.

3 XML and Web principles

Extensible Markup Language (XML) is a simple and flexible text format derived from Standard Generalized Markup Language (SGML) [8] [90]. Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. It is created and maintained by World Wide Web Consortium (W3C) [15]. This consortium consists of several big companies, including IBM, Microsoft, Oracle, and others. The key feature of XML is the ability to define other markup languages or other standards.

3.1 XML standards

Today there exists several XML standards, among them we have some quite known as HTML, XSLT and CSS. An up to date list of all XML standards can be found on the W3C site [15].

3.1.1 Hyper Text Markup Language (HTML)

Hyper text markup language (HTML) is probably the most known and used markup language in the world [50]. It is used to display content and presentation on every web page, on every web server in the world. Today HTML is defined in XML, but this language is much older than XML. It was developed by Tim Berners-Lee while he worked at *CERN*, and was popularized by the *Mosaic* browser developed at *NCSA* in the beginning of 1990s. In 1995 came HTML 2.0 and not much later a third version. In the late 1990s most browser vendors implemented HTML using different approaches. This ended up with pages looking differently in almost every browser. The early versions of HTML lacked a splitting mechanism between content and presentation, forcing a developer to mixed it up in one place. This gave complicated code, and did not guaranty the same layout and content across different platforms and in different browsers. All those problems where solved in the latest HTML version, which in fact was the first one to be defined in XML (HTML version 4.0). By splitting content and presentation into two separate languages, making HTML a more strict language and forcing vendors to implement the same features using standards, this was achieved.

3.1.2 Cascading Style Sheets (CCS)

Cascading Style Sheets (CCS) is a standard to describe presentation such as fonts, colors and styles in documents [78]. It is most often used in a combination with HTML to define style for HTML tags. Even with CCS we are not guaranteed that everything will look the same in all browsers. Take a look at this web site and you will find the different looks in different browsers on different buttons, select boxes and text inputs that are used on web pages [80].

3.1.3 Extensible Stylesheet Language Transformation (XSLT)

Extensible Stylesheet Language Transformation (XSLT) is a language to transform XML documents to different formats. It is a kind of script language that takes raw XML as input, and transforms it to any desired format (like PDF, HTML or Tex).

3.2 Parsing XML documents

At some point we will have to process XML documents to get a certain tag or tag value. This process is called parsing. There are two main approaches to parse XML documents, one developed by the W3C (Document Object Model also called DOM) and Simple API for XML (SAX) by an open source community. Both are widely used in different parsers and there exists several implementations that use a mixture of both. Some more or less known are *JDOM* and *JAXP*.

3.2.1 Document Object Model (DOM)

Document Object Model (DOM) is an API for an abstract document representation in memory. Once a document is in memory, different parts of it can be accessed and processed. Currently the DOM specification is maintained by W3C, and consists of three main specifications and several sub specifications. It is, beside HTML and CCS probably the most used W3C specification in the world. Every web browser that handles some kind of capture motion events, use this technology. This technology is a part of both the JavaScript and ECMA programming languages (see chapter 7 for more info). The draw back of this technology is that it consumes large amounts of memory. If we are dealing with small to medium documents that does not make any big difference. The difference comes when we try to load a 2 MB document into memory on an over loaded server. Although DOM is a very good and simple technology, it is not very good at larger documents.

3.2.2 Simple API for XML (SAX)

Simple API for XML (SAX) is another parsing specification for XML documents [92]. It has nothing to do with the W3C, since it is an open source community project. Their main goal was to make an XML parser specification that has performance as its main feature. SAX does not load an entire document into memory (as DOM), it iterates through it and handles any predefined events. Events can be different types of variables or predefined document characteristics, which the parser will notice and process further. This approach minimalisms memory usage, but has also several draw backs. While DOM can be used in browsers to check for user motion or input, it is impossible to do with SAX. On the other side, SAX is the preferred way when we deal with some kind of input parsing and output processing. For example in a server task that reads XML files, parses them and puts the data into a database, is it more efficient to use SAX.

3.2.3 Implementations

There exist several implementations of parsers for XML documents. The Apache open source community has a family of different XML parsers. They are fast, reliable and are distributed under an open source license (it means that they are free). The Apache XML parser, called Xerces (named after the Xerces Blue butterfly), is implemented in both Java, C/C++ and Perl [94]. It can be both used as a DOM and SAX parser , and even has the ability to transform documents based on the XSLT standard. This parser is based on the award wining IBM *XML4J* parser.

3.3 Web Principles

A part from standardizing XML and all related technologies, the W3C also standardizes the Web. Their goal is to make the Web more accessible, fun, safe and entertaining for a common non technical user. Web standardization is their de facto task, and XML is just a part of it. The Web standardization efforts, are sometimes referred to as Web principles.
Since the web principles can not be defined in a mathematical way (as for example XML or HTML), they are called recommendations or guidelines. There exist three different top level guidelines for web principles, these are:

Internationalization

Device Independence

Accessibility

3.3.1 Internationalization

The internationalization activity is mainly based around the ability to create and process information for a wide range of audiences around the world [6]. They have from the early stages promoted the use of *unicode* standards to identify and describe characters. Characters need to be identified and described in a proper way in order to make them understandable for humans. Imagine sending a Polish letter 'ł', which in fact is not a 'l', but can be misunderstood by a program to be a 'l', but the problem is not just with the charaters. In several languages the typography is also different from ours. For example in Arabic and Hebrew, text is written and read from right to left, while in Japanese text is often written from top to bottom.
Based on these requirements the Internationalization group works closely with other groups. Over the last 12 months they have reviewed and followed up discussions on several W3C specifications. These reviews and discussions often led to changes in functionality and in wording.

3.3.2 Device Independence

Due to the rapid increase of different devices that can access the Web, a device independence working group was established at the W3C [35]. Their mission is to avoid fragmentation of the Web into spaces those only are

accessible from certain types of devices. The goal of this working group is to develop ways for future web content and applications to be authored, generated, or adapted for a better user experience when delivered via many device types. In other words, there should not be any situations in the future where a user agent can not access a Web site, only because it was developed for some other kind of device.

3.3.3 Accessibility

The third and last guideline is the accessibility guideline, which consist of several other smaller guidelines for different accessibility contexts [36]. The goal of this working group is to make Web accessible to people with disabilities. Most documents are guidelines with checkpoints that tells the developer which decisions are better suited for the current environment. For example the Web Content Accessibility Guidelines 1.0 [3] discus Web content development. They have for example guidelines like:

> Provide equivalent alternatives to auditory and visual content.

That means that there always should be equivalent information to the auditory and visual content. Some people may have problems using images, applets or sounds, and a site should have the possibility to present the equivalent information in a textual format.

3.4 How this chapter is related to my thesis

It may see strange that all this is discussed in the first chapter. XML and XML technologies are related to almost every chapter in this text. Both MathML and SBML, which in fact, are a very central part of this thesis are defined in XML. Web principles are used (or should be) in the development of my application, since it is mainly web based, and those guidelines are a very important thing in software engineering.

4 Mathematical Markup Languages

In order to properly explain the different parts and aspects of Systems Biology Markup Language (see next chapter) we also need to explain the mathematics used in that language.

As the Web began to be more and more popular, more scientists began to publish their work on different sites. The biggest problem was to show mathematics on the Web. Before the Web era almost all mathematical equations where written down in TeX [47] format which was de-facto standard for exchanging scientific documents. It is a very good typing notation, but is not suitable for Web pages. Using TeX our equation is processed and becomes a picture and breaks fundamental principles of the Web: accessibility, internalization and device-independence (see chapter 2). Our equation in picture format (e.g. as a jpg picture) is probably displayed correctly in most browser, but will not be readable in a text browser (e.g. Lynx) or on a portable device such as a cellular phone. *Robots* that indexes the Web will also have problems with that kind of picture, and it will lead to negative results if someone tries to search it up using a search engine.
On the other hand it is also possible to use HTML. An equation will then be written as normal text and placed inside some HMTL tags. Unfortunatly HTML (see previous chapter on HTML discussion) was not designed to describe mathematical notation, nor any other scientific work. Another way to do it is to use some kind of 'ASCII art', such as this

```
a + b^2 + 12
------------
   15 * c
```

but this approach does also violate fundamental principles of the Web. The solution is a mathematical markup language, standard that can be implemented in software packages and used by all vendors. Especially is this crucial to browsers, because they show contents of a web page.
Currently there are several mathematical markup languages. Two of the most common are Mathematical Markup Language (MathML) [33] and OpenMath [66]. MathML is the most widely used, and is also used by the Systems Biology Markup Language. This chapter will describe both.

4.1 Mathematical Markup Language (MathML)

MathML is a W3C recommendation from february 2001. Currently the newest version is MathML 2.0 [22]. It is an XML application for describing mathematical notation and capturing both its structure and content. The goal of MathML is to enable mathematics to be served, received and processed on the World Wide Web, just as HTML has enabled this functionality for text. This language consists of two different markups: presentation markup, and content markup. The main difference is that presentation markup is more aided for processing by humans, while content markup is more aided for processing by machines or with machine support.

4.1.1 Presentation Markup

Presentation markup can be used to describe the layout structure of a mathematical notation. It describes an equation somewhat similar to the way

one would read it, defining elements such as subscripts, fractions and operators. This little example is showing a equation,

$$\frac{a + b^2 + 12}{15 \times c}$$

here is the same equation in MathML.

```
...
   <math>
     <mrov>
       <mfrac>
         <mrov>
           <mi>a</mi>
           <mo>+</mo>
             <mrov>
               <msup>
                 <mi>b</mi>
                 <mn>2</mn>
               </msup>
             </morv>
             <mo>+</mo>
             <mn>12</mn>
         </mrov>
         <mrov>
           <mn>15</mn>
           <mo>x</mo>
           <mi>c</mi>
         </mrov>
       </mfrac>
     </mrov>
   </math>
...
```

Below is the tree view of the presentation markup MathML example. Such a tree view makes it easier to understand MathML. This example is identical to the one above. The point is to show how MathML is connect and has an abstract tree representation.

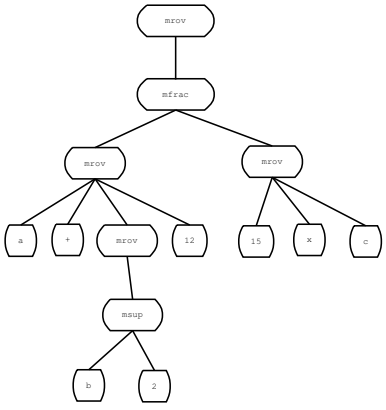

Figure 1: Tree view of the presentation markup MathML example

4.1.2 Content Markup

Content markup is more suitable for machines or machine processing than presentation markup. This can be fairly easy seen comparing content and

presentation examples (for content example, see below). Content markup consists of about 120 elements that accepts a dozen attributes. The 'apply' element is probably the most important content element. It is used every time a function or operation is applied to a collection of arguments. This element can also be seen as a branch. It also adds more meaning to formulas and equations giving them the possibility to be used by mathematical software. Content markup includes basic set of most standard areas of mathematics, such as arithmetics, algebra, logic, set theory, calculus, sequences, series, linear algebra and statistics.

This example is showing the same equation as the presentation example above. Most of this example is self describing. In content markup there are distinctions between variables and numbers. For variables MathML uses the 'ci' element, and for numbers the 'cn' element.

```
...
   <math>
     <apply>
       <divide/>
         <apply>
           <plus/>
           <ci>a</ci>
           <apply>
             <power/>
             <ci>b</ci>
             <cn>2</cn>
           </apply>
           <cn>12</cn>
         </apply>
         <apply>
          <times/>
          <cn>15</cn>
          <ci>c</ci>
         </apply>
     </apply>
   </math>
...
```

Below is the tree view of the content markup MathML example. Such a tree view makes it easier to understand MathML. This is the same example as above.

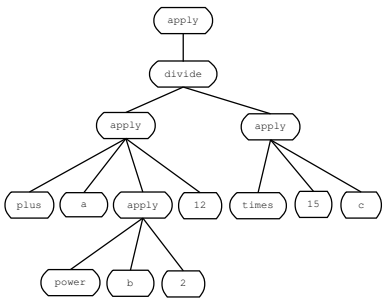

Figure 2: Tree view of the content markup MathML example.

4.1.3 Presentation vs. Content

The big question is when to use which. As a thumb role one can say that presentation markup is more suitable for human reading, while content is better for machines to process. However there is always that possibility to use both. A mixture of presentation markup and content markup is allowed by the MathML recommendation. However there are two ways to mix markup in MathML; mixed markup and parallel markup. Mixed markup is when both markups are present in a single tree (example follows) as one expression.

This example shows the mixed markup.

```
...
   <math>
     <apply>
       <mfrac>
         <apply>
           <mi>a</mi>
           <mo>+</mo>
             <apply>
               <apply>
                 <mi>b</mi>
                 <mn>2</mn>
               </msub>
             </apply>
             <mo>+</mo>
             <mn>12</mn>
         </apply>
         <apply>
           <mn>15</mn>
           <mo>x</mo>
           <mi>c</mi>
         </apply>
       </mfrac>
     </apply>
   </math>
...
```

The only difference between this example and these two on the top is use of 'apply' elements instead of 'mrov'. It is fully legal to combine those two markups in that kind of way.

Parallel markup is when both markups are present in a document to explicit provide presentation and content. Two identical equations are then represented presented both as presentation and content. If we put both examples from the section above in one document, we will get a parallel markup. This can be useful in situation, where two or more software packages read only one markup. However, problem can arise when one package updates one of the models.

4.1.4 Implementations

There are several software packages that supports MathML. For a complete list see the MathML page under 'software page' [33]. The most know are the *Amaya* browser/editor, *Mathematica* and browser support for several browsers including *Mozilla/Firefox*, Netscape, Opera and Microsoft IE. MathML is also implemented in several software packages related to the Systems Biology Markup Language (see next chapter).

4.2 OpenMath

The other big standard for representing mathematical expressions is the OpenMath [66]. This language was mainly a language for computer algebra packages, but evolved to a more common mathematical standard. It was originally developed during several workshops in the 1990s, and is now maintained by the openmath society (see [66] for more info). The representation of mathematics in OpenMath contains of a small set of 'expression tree' cuntructors on some basic objects (byte-arrays, strings, integers, variables) and on the usage of symbols defined in predefined libraries (called Content Dictionaries). Since this language is very similar to MathML, it may appear very competitionary, but it is rather complementary. This is because MathML deals principally with presentation of mathematical objects, while OpenMath is only concerned with their content. OpenMath can also be used with MathML. It is then embedded inside a MathML object, representing its content, while MathML represents its presentation.

4.2.1 Overall specification

A mathematical object can be represented by three layers in the OpenMath language. A layer is a representation of an abstract place conceived as having depth. These are:

the privat layer

the abstract layer

the communication layer

The private layer is used for the internal representation of a mathematical object. The abstract layer is used for the representation as an OpenMath object. While the communication layer is used for translating the OpenMath object to a stream of bytes.

There are two major encodings, one to XML format and the other to binary format. The strong side of OpenMath is the ability to construct several mathematical symbols. These symbols are grouped in official and unofficial mappings (libraries) called 'Content Dictionaries' (CD). These CD make the OpenMath language a less compact and unlimited language than the MathML. In other words the OpenMath standard is more suitable for more advanced uses and users.

The programs that acts as an interface between software systems and OpenMath are called 'phrasebooks'. These 'phrasebooks' translates the OpenMath objects, as defined according to the Content Dictionaries, to the internal representation used by the specific software package. The core of OpenMath is the OpenMath object model, not the XML encoding as in MathML.

4.2.2 OpenMath in use

There are not many packages that use OpenMath,but one is the AciveMath project [5]. ActiveMath is an adaptive, interactive learning environment for mathematics, it is being developed by two universities in Germany, the DFKI

[37] and the Saarland University [82]. This project use OpenMath objects to store mathematics in a backend database. A more comprehensive list of software related to the OpenMath project can be found on the OpenMath homepage [66].

4.2.3 OpenMath vs MathML

As mentioned above, OpenMath is very similar to the content part of MathML. It can also be mixed with MathML. This is mainly done by using MathML as presentation and OpenMath as content markup.
The OpenMath can be applied to arbitrary areas of mathematics without changing the language (new CDs are added). This is not possible with MathML, because MathML can not describe sematics of a mathematical object. MathML has just a small set of fixed symbols, so it is more suitable for low level mathematics. Expressing mathematics at high university level is much easier with OpenMath. The drawback of this language is its lack of software support. It is only implemented in some software package and is not so widely used than MathML. The solution is probably to try to use MathML, but for more complex mathematics its a better idea to use a mixture of both.

5 Systems Biology Markup Language (SBML)

Systems biology markup language (SBML) [51] is a markup language to describe models of biochemical reaction networks. It can be used to model/-describe metabolic networks, cell-signaling pathways, genomic regulatory networks and many other areas in Systems Biology (explained below). This language is maintained by the SBML group that was founded by JST ERATO Kitano Symbiotic Systems (Japan) [72]. Today the SBML Team is an international research team distributed at institutions around the world. The groups focus is on research and software development for systems biology. The SBML is a free and open language.

5.1 Systems Biology

What is Systems Biology? According to American Chemical Society it is "Integrative approach in which scientists study pathways and networks that will touch all areas of biology, including drug discovery" [88]. The Stuttgart University has it's own definition, according to them Systems Biology is a "Systematic approach, not focused on individual genes and individual proteins, instead interested in analyzing whole systems of genes or proteins by capturing information from many different elements of the overall system" [79]. Systems biology is a science involving biology, computation and analysis. The whole point is to understand biology at the system level, and to examine the structure and dynamics of cellular and organism function, instead of characteristics of isolated parts of a cell or organism. Systems biology is a quite new research field, one of the first papers came in 1998. Today we have several institutes and research facilities that spend much of their time and money on this field. A good overview over this research field has been written by Hiroaki Kitano, and can be found in Nature [46]. Systems Biology Markup Language is a tool meant to facilitate research in this field.

5.2 The Language

The first edition of this language came in 2001. That year is also the beginning of the SBML team and community. This first edition was called SBML level 1 version 1 [1]. In the summer of 2003 came level 1 version 2 [4]. There are only some small changes between these two versions. In late June 2003 came SBML level 2 version 1 [2], which will be focused on in this thesis. SBML level 2 version 1 will also be the markup language in which all models are stored in my solution discussed later in this thesis. The major change between level 2 and level 1 is the use of a separate mathematical language and content elements of MathML (see previews chapter) instead of self defined mathematical functions. SBML level 2 version 1 uses a subset of MathML 2.0 to define all mathematical functions [33].

5.3 Specification

This language contains several components. Each of which will be discussed in detail, and examples will be given. However the whole specification will not be discussed, just some essential parts. In appendix B and C one will find two examples of a SBML model. Appendix B shows the Beeleur-Reuter Mammalian Ventricular model from 1977 [32]. Appendix C shows a SBML model with aspects not covered by the model above. This model is based on pure fictional example. All examples below are taken from those two models.

5.3.1 Overall Definitions

All SBML models have the same form. They all consist of several parts, as listed below.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" level="2" version="1">
  <model id="ReactionAndEventExample" name="ReactionAndEventExample">

    <listOfFunctionDefinitions>
    ...
    </listOfFunctionDefintions>

    <listOfUnitDefinitions>
    ...
    </listOfUnitDefinitions>

    <listOfCompartments>
    ...
    </listOfCompartments>

    <listOfSpecies>
    ...
    </listOfSpecies>

    <listOfParameters>
    ...
    </listOfParameters>

    <listOfRules>
    ...
    </listOfRules>

    <listOfReactions>
    ...
    </listOfReaction>

    <listOfEvents>
    ...
    </listOfEvents>

</sbml>
```

Sometimes models also have HTML tags. These tags are encapsulated in parts called 'notes', as seen in this tiny example.

```
...
  <notes>
        <body xmlns=''http://www.w3c.org/1999/xhtml''>
           <p>Hellow World!</p>
        </body>
  </notes>
...
```

There are also some important definitions at the beginning of the model. These include annotations that are used in the model, model name, model id, SBML level and SBML version.

5.3.2 Mathematics in SBML

In the two first versions of SBML (level 1 version 1 and version 2) a self defined mathematical notation was used. In level 2 MathML was introduced, and will probably be used in the future releases. Currently only a subset of MathML elements are used in SBML models similar to that used by CellML. MathML elements can only be used in some containers like rules, functions, reaction kinetics, stoichiometries and events (see below). Elements for representing ordinary *differential equations* are included, but not for partial differential equations.

5.3.3 Function Definitions

The function definition part gives us a possibility to define our own function. It consist of an identification tag ('id'), optionally a name tag ('name') and a math element. This math element is a MathML 2.0 element.
This example shows the following equation.

$$\beta_{x1} = 0,0013 \frac{e^{-\left(\frac{V+20}{16,67}\right)}}{1+e^{-\frac{V+20}{25}}}$$

```
...
 <functionDefinition id="beta_x1">
       <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
        <bvar><ci>V</ci></bvar>
        <apply><times/>
        <cn>0.0013</cn>
        <apply><divide/>
        <apply><exp/>
        <apply><minus/>
        <apply><divide/>
        <apply><plus/>
          <ci>V</ci>
          <cn>20</cn>
        </apply>
        <apply>
          <cn>16.67</cn>
        </apply>
        </apply>
        </apply>
        </apply>
        <apply><plus/>
          <cn>1</cn>
        <apply><exp/>
        <apply><minus/>
        <apply><divide/>
        <apply><plus/>
          <ci>V</ci>
          <cn>20</cn>
        </apply>
        <apply>
          <cn>25</cn>
        </apply>
        </apply>
        </apply>
        </apply>
        </apply>
        </apply>
        </apply>
       </lambda></math>
      </functionDefinition>
...
```

The functionDefinition part is in fact MathML with a assignment field linking mathematics and SBML together. Functions defined in this part, can

be used other places in the model (e.g. in Rules or Reaction, see below for more info).

5.3.4 Unit Definitions

The unit definition part gives us a possibility to define our own units. There are several predefined units in the SBML specification, and these can be found on page 14 in the SBML level 2 version 1 specification [2]. This example shows two different units, 'per_millivolt' and per_millivolt_millisecond'. The first one consist of one predefined unit (volt), and the other one of two predefined units (volt and second). There are no restrictions on how many units a unitDefinition can have.

```
...
    <unitDefinition id="per_millivolt" name="per_millivolt">
        <listOfUnits>
          <unit kind="volt" scale="-3" exponent="-1"/>
        </listOfUnits>
    </unitDefinition>
    <unitDefinition id="per_millivolt_millisecond" name="per_millivolt_millisecond">
        <listOfUnits>
          <unit kind="volt" scale="-3" exponent="-1"/>
          <unit kind="second" scale="-3" exponent="-1"/>
        </listOfUnits>
    </unitDefinition>
...
```

5.3.5 Compartment Definitions

Compartments in a SBML model represents the actual structure in which the specie or reaction is located. Often they correspond to a part of a cell or a part of an organism. This example shows a compartment named 'cell'.

```
...
  <listOfCompartments>
       <compartment id="cell" name="cell"/>
  </listOfCompartments>
...
```

5.3.6 Species

Species represents variables that change during a reaction. It can both be (as in this example) differential equations or chemical reaction. They must also be located in a compartment. For example, the specie with id and name 'V' can represent a differential equation or a reaction in the cell, and has a certain initial amount of a unit.

```
...
   <listOfSpecies>
     <species id="V" name="V" compartment="cell" initialAmount="-84.624"/>
     <species id="m" name="m" compartment="cell"  initialAmount="0.011"/>
     <species id="h" name="h" compartment="cell"  initialAmount="0.988"/>
     <species id="j" name="j" compartment="cell"  initialAmount="0.975"/>
     <species id="Cai" name="Cai" compartment="cell" />
     <species id="d" name="d" compartment="cell" initialAmount="0.003"/>
     <species id="f" name="f" compartment="cell"  initialAmount="0.994"/>
     <species id="x1" name="x1" compartment="cell" />
   </listOfSpecies>
...
```

5.3.7 Parameters

Parameters in an SBML model represents a variable for use in mathematical formulas. These formulas are written in MathML. Since they have a constant value for the duration of a simulation, they are called parameters instead of variables. Parameters can also be defined within individual reaction definition these are local parameters. Local parameters override any global parameters having the same name, which are defined in the parameters part. This example shows some parameters, and connections to unitDefintions.

```
...
  <listOfParameters>
      <parameter id="C" value="1.0" units="microF_per_cm2"/>
      <parameter id="i_Na" units="microA_per_cm2"/>
      <parameter id="g_Na" value="4.0" units="milliS_per_cm2"/>
      <parameter id="E_Na" value="50.0" units="millivolt"/>
  </listOfParameters>
...
```

5.3.8 Rules

Rules in SBML represents constraints on variables for cases in which the constraints cannot be expressed using reactions nor the assignment of an initial value to a component in a model. Rules are also divided into three kinds. Algebraic rules, assignment rules and rate rules.

If the left-hand side is a rate-change, and W is a vector of variables that may include x, we have a rate rule (differential equation)

$$\frac{dx}{dt} = f(W)$$

The left-hand side is a scalar, and x is a variable, and V is a vector of variables that does not include x, we have an assignment rule (similar to function definition)

$$x = f(V)$$

The left-hand side is zero, and W is a vector of variables, we have an algebraic rule

$$0 = f(W)$$

The first example is a rate rule taken from the Beeler-Reuter model. Both alpha_x1 and beta_x1 are assignments. The definition of beta_x1 can be found in the function definition example above.

$$\frac{dx1}{dt} = (\alpha_{x1}(1 - x1) - \beta_{x1}x1)$$

```
 ...
<rateRule variable="x1">
       <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply><minus/>
        <apply><times/>
        <apply>
          <ci>alpha_x1</ci>
        </apply>
        <apply><minus/>
        <apply>
          <cn>1</cn>
        </apply>
        <apply>
          <ci>x1</ci>
        </apply>
        </apply>
        </apply>
        <apply><times/>
        <apply>
          <ci>beta_x1</ci>
        </apply>
        <apply>
          <ci>x1</ci>
        </apply>
        </apply>
        </apply>
       </math>
      </rateRule>
```

The second example shows a assignment rule. This rule is very similar to the function definition part and shows assignment of the i_Na variable from the Beeler-Reuter model.

$$i_{Na} = \left(g_N a m^3 h j + g_{Nac}\right)\left(V - E_{Na}\right)$$

```
      <!-- Assignment Rules -->
      <assignmentRule variable="i_Na">
       <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply><times/>
        <apply><plus/>
        <apply><times/>
          <ci>g_Na</ci>
        <apply><power/>
          <ci>m</ci>
          <cn>3</cn>
        </apply>
          <ci>h</ci>
          <ci>j</ci>
        </apply>
        <apply><plus/>
          <ci>g_Nac</ci>
        </apply>
        </apply>
        <apply><minus/>
        <apply>
          <ci>V</ci>
        </apply>
        <apply>
          <ci>E_Na</ci>
        </apply>
        </apply>
        </apply>
       </math>
     </assignmentRule>
 ...
```

5.3.9 Reactions

Reactions in SBML represents any transformation, transport or binding process. It is typically a chemical or biological reaction, that can change the amount of one or more species.
This example shows the following example:

$$X \xrightarrow{kj} S$$

```
<listOfReactions>
        <reaction id="reaction1" reversible="false">
          <listOfReactants>
            <speciesReference species="X"/>
          </listOfReactants>
          <listOfProducts>
            <speciesReference species="S"/>
          </listOfProducts>
          <kineticLaw>
            <math xmlns="http://www.w3.org/1998/Math/MathML">
              <apply>
                <times/>
                <ci>k</ci>
                <ci>j</ci>
              </apply>
            </math>
          </kineticLaw>
        </reaction>
</listOfReactions>
```

5.3.10 Events

Events in SBML are descriptions of explicit instantaneous discontinuous states that changes in the model. This example shows the following event.

$$X \leq k \rightarrow S = 0$$

```
<listOfEvents>
      <event>
        <trigger>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <leq/>
              <ci>X</ci>
              <ci>k</ci>
            </apply>
          </math>
        </trigger>
        <listOfEventAssignments>
          <eventAssignment variable="S">
            <math xmlns="http://www.w3.org/1998/Math/MathML">
              <cn>0</cn>
            </math>
          </eventAssignment>
        </listOfEventAssignments>
      </event>
    </listOfEvents>
```

5.4 Software support

There are many software packages that implement the SBML standard (both levels). Those programs ranges from editors, parsers to complete cell designers and complex cell simulators. A complete list can be found on the SBML page [51]. Today there are over 60 software packages that support

this language, and each month this list grows with one or two. The most important are listed below.

libSBML is an open-source library providing an API for reading, writing and manipulating data expressed in the SBML. It is implemented in C and C++ and works on all major platforms [55].

MathSBML is an open-source library for working with SBML models in Mathematica [57].

SBW System Biology Workbench. This is an open-source library for modular, broker-based, message-passing framework for communication between application that aid in research systems biology [87].

The SBML is also used as a standard model for exchanging biological and chemical models by the BioSpice community [14]. BioSpice is a suite of software tools used to construct computer models that simulate the complex behavior of living cells. This package has been developed as an open source project, partly sponsored by the *DARPA* with contributions from all the key universities in the USA.

5.5 The Future

This SBML language is under continues evolution and development. In near future a new third level will be announced and more will probably come. There are also several mailing forums and editorable wikis on the SBML page, on which scientists and programmers discuss development of this language and additional software. Approximately 2-3 posts are written per day, which makes discussion quite moderate. Every one is free to join these discussions. Some of the key features announced with the new level are arrays, parameter sets and diagramic layout for figures.

5.5.1 New features (Only proposals)

These are some new features that are proposed on the SBML site. Beavare that these only are proposals, and can be left aside in a new level of this language.

Arrays One proposal is to use arrays to store different types of variables like parameters or species. This can be achieved by using self defined arrays, or to use arrays that are integrated into MathML (remember that SBML only use a subset of MathML operators).

Diagramic layout for figures The other proposal is concerning a posibility for including diagrams into SBML documents. Diagrams could be included several places in the model. For example there could be a posibility to use diagrams to describe the overall picture of how reaction, rules and function definitions are connected with compartments and species.

5.6 Other biological and chemical markup languages

There are several other both biological and chemical markup languages than the SBML. Since it is very easy to define new languages in XML, scientists and researchers are often developing small, non-complex languages to aid their exchange within their lab. Such small projects often develop into new standards.

5.6.1 CellML

CellML [12] is probably the closest related markup language to the SBML. This language is more suitable to describe the structure and underlying mathematics of cellular models than the SBML, which as described is aimed at exchanging information about pathways and reaction models.
Both teams are actively discussing how the two languages can be made work together without losing information by converting between these two formats. At the time of the writting no software has been publicated to transform between these two languages.

5.6.2 Mouse Annotation XML

Mouse Annotation XML (MaXML) is a specification to describe mouse *cDNA* annotation data [91]. Annotations is the information included with a DNA sequence, such as location of ions and description of biological functions. In other words, mouse annotation data is data used to describe genes in order to characterize them. It was developed by scientists in Japan to aid the description of cDNA. Currently there are three international *DNA data banks* which have collected annotation, but do not have any standard format for document exchange. MaXML can be partly seen as one such effort, although these DNA banks have recently began to export their models in XML format.

5.6.3 Chemical Markup Language

Chemical Markup Language is a language designed to describe molecular information such as chemical equations and reactions [49]. It is defined in XML a is widely used among chemical scientists in the US and by some companies. The list of companies that are using CML is confidential, and will probably be soon announced.

6 Databases

What is a database? Database is a common a word on a database management system (DBMS). This is the actual definition, but often they are defined as the actual data that is stored inside them. A DBMS is an application that has been vital to all approaches to store data on computers for the last forty years. They became a reality in the 1950 when programmers started to use different kinds of file systems to store data. In the 1960 and 1970 many new approaches where presented, implemented and used. Most databases where in the financial branch, especially in banks and used on stock exchanges. At the beginning the of 1970 Codd, a scientist from *IBM*, published a paper [13] where he presented a new approach in database theory. His article presented a mixture of mathematical theory with a practical approach, where he presented a relational model for databases. Ten years later became databases that implemented the relational model a de facto approach in database world.
A DBMS is a system to store large amounts of data. Data stored on disk will stay on disk, even if the designated server or application goes down - this is often referred to as persistent data. A database management system should also provide a programming interface and a transaction management system. A programming interface for relational database management systems is a structured query language, also known as SQL.

6.1 Transaction management system

A common feature in databases is a transaction management system. Transaction systems are much older than the relational model, but are a very important aspect in databases, and probably the most complex one. They act as an application to control in-out flow in the database system. All aspects of storing, reading and updating by concurrent users or programs are handled by transactional management systems. Their logic consist of four basic aspects which use the acronym ACID.

A Atomicity

C Consistency

I Isolation

D Durability

Atomicity means that every transaction needs to be completed or aborted. A transaction can not be partially completed, or partially aborted, because then it will violate the atomicity and consistency aspect. Consistency means that all constraints are fulfilled, in other words all data after a transaction is completed, consistent and written to disk or device. Isolation means every transaction has the impression of being the only one being executed. In other words, programs controlling flow to a DBMS, does not have to think about concurrent writes or reads. All this is done by the database management system. Durability is the fourth and last aspect of ACID covering data written to disk or device. It means that written data

will stay on disk or device, even if the server goes down. If the server had gone down during a transaction, data would be lost, even if some the data have been about to be writen disk.
As mentioned above, transactions are a very important aspect of databases. All existing databases have some kind of transactional management system integrated, one common thing for them all is the fact that their base logic is based on ACID.

6.2 Relational Databases and Structured Query Language (SQL)

As mentioned above the relational database theory where born in an article published by Codd in the 1970. This article presented both mathematical theory as relational algebra and a practical solution, in one extended paper. This was a revolutionary approach, that was fairly easy to understand and to implement.

6.2.1 Relational algebra

Relational algebra was first used by Codd to define a mathematical solution for his practical problem. This algebra is closely based on theoretical basis in set theory and first order predicate logic. Codd used it on sets of tuples (in other words on relations) that could be used to express typical queries about those relations. It consisted of five operations on sets, union, set difference, Cartesian product, selection and projection. The first databases that implemented Codd's relational algebra, mainly used this algebra as their query language. The only difference was the fact that they used bags instead of sets (sets do not allow duplicates, but bags do).

6.2.2 The Structured Query Language (SQL)

The first versions of query languages where almost pure implementations of Codd's relational algebra. This implementation of relational algebra was soon called a 'structured query language'. The problem was that theory, even when presented in a very elegant way in Codd's article, not always works in real world applications. Almost all DBMS vendors that based their implementations on relational algebra, ended up with different query language dialects. After several years, the first standardization came with the ANSI SQL, also known as SQL 1. SQL kept developing, and in 1992 came the second standard (SQL-92 or sometimes called SQL 2), and the last standardization in 1999, SQL-99 (SQL 3).
The SQL language is probably the biggest reason for the success of relational databases. It is quite easy to learn and a standardization gives one the opportunity to easy switch between different vendors. Although there exists several official and unofficial dialects, switching is not the biggest problem. The biggest problem is integrating SQL with todays programming languages. This problem is called the impedance mismatch. SQL has a very different data model from other languages. Most common modern programming languages use some kind of object-orientation, like Java and C++, or a more traditional C-like approach (like C and php). Both orientations are very different from the relational model, and gives programmers

and modelers a great challenge when designing a new system.

Modern SQL, which in fact is SQL-92 with some SQL-99 implementations, consist of three languages.

DDL Data Definition Language

DML Data Manipulation Language

DQL Data Query Language

All of them are defined in SQL, but the primary relational query language presented by Codd, where only the DQL. DDL is the language for defining the database. It defines and creates databases, tables, attributes and constraints. This little example shows creation of a person table with four attributes. SQL syntax used in the examples below examples is the same as that is used in the MySQL [62] database management system.

```
CREATE TABLE person{
   personid   int(4) NOT NULL,
   personname varchar(20) NOT NULL,
   age        int(2) NOT NULL,
   phone      int(10)

};
```

These attributes are, as we see above; personid, personname, age and phone. Three of them are different sized integers, while personame is a 'varchar' with a maximum length of 20 charachters. 'Varchar' is the same thing in a SQL world as a common 'string' in imperative programming languages. The 'NOT_NULL' attribute represents a constraint, which indicates that an entity can not be empty or 'null' as it is called in the computer world.
After the creation of a database we use the DQL to extract information out of it. This is done by a query mechanism in SQL. All queries have a common form, and are build around three keywords. These are:

```
SELECT <attribute list>
FROM   <table list>
WHERE  <condition>
```

Others are also allowed, a complete list of allowed keywords can be found in all three SQL standards, and in the different DBMS manuals. This little example show a query that returns personname from the table 'PERSON'.

```
SELECT personname FROM person;
```

Note that this query should always return all personnames in this table. If we change the 'personname' with 'personid' in the query above, we should get a list with the same size as the first one. This is because of the create statement in the top query. According to this statement, three of four created attributes in the 'PERSON' table has a constraint not allowing null tuples to be added. Since our database management system has a transaction management system integrated, it should not allow updates with null tuples where they should not be.

The last part of SQL is the DML. This part of the SQL language is used to manipulate different parts of data, like updating tables, or updating tuples and inserting new data. Below is an example where we are inserting data into a person table.

```
INSERT INTO person ('personid', 'personname','age','phone')
VALUES (1,"Codd", 60,12345678);
```

There exist a famues quoutation saiying that SQL is the 'intergalactic data speak'. Due to its popularity, it is probably true.

6.2.3 Modern relational databases

Modern relational databases are often called object-relational databases. Object-relational databases are not the same thing as object-oriented databases, which is in fact a different paradigm. Since the release of Codds article many things have changed in computer science. During the last ten-fifteen years new needs have emerged, and databases had to switch their main orientation, from almost pure banking and broker applications, to *CASE* applications, different *GPS* applications to more common internet-stores. These and other requirements where the key factor to develop a new SQL standard. This standard describes new ways to handle complex data as objects, including time series, geospatial data and binary media, such as audio, video and images. Those complex data are often stored as binary objects, without any contraints. This approach is quite new for relational databases, and violates one of the basic transactional aspects - nainly atomicity. But due to new requirements, is this quite useful, when storing complex data. Just imagine to store an image in traditional relational database. Each character or byte need to be stored in almost one tuple, this gives long time for both storing and reading from the database. Storing all data in one tuple without any constraints is a much better solution. This SQL standard is called SQL-99 or SQL 3, which in fact is the object-relational standard. The problem with this standard is that it was developed in cooperation with all the biggest database companies and relational thinkers. As a result of this cooperation a huge SQL-99 standard was developed. It consists of more than fifteen hundred pages, a is not a trivial thing to implement. There seems that a new SQL standard or at least a new and lighter version of it is under way in the close future.
Todays database management systems that can be called relational databases, implements SQL-92 and some parts of the SQL-99 standard. Currently there are no databases that implements the complete SQL-99 standard.
Probably the best databases in the world, are implemented by the Oracle company [16]. Their newest database management system is oracle 10. The other leading vendors include Informix and IBM, and some open source projects like the PostgreSQL database [69] and MySQL [62].

6.3 Object Oriented Databases

In the late 1980 came the idea to link up databases and object-oriented programming languages. The object-oriented paradigm had been used with great success for several years, and was de facto programming paradigm for the future. It was and is a very good programming paradigm, but due to impendence mismatch, it is quite hard to make it cooperate with relational databases. An object-oriented database would be a good solution (see the OMG [31] and the ODMG [30] specifications for more details). Both the database and an application using it, would be expressed in the same language and could use the same objects or variables. This makes programming much easier for both the designer and the programmer. Just imagine doing all logic in one place, and not have to be concerned about reading or writing data to a database.

The problem with object-oriented databases is the fact that they are a new paradigm and all the existing relational database implementations are probably much better than the object-oriented. The biggest reason for that is that there is much more money behind relational databases, than the object-oriented.

Most applications use relational databases because they are much better and safer to use. The object-oriented often to not have a good transactional system, and are much slower on queries, which indeed are not much optimized. However, since most bigger applications written today use object-orientation, it is much easier to use the same paradigm on the database. The other problem is migrating from current database applications to future object-oriented databases. This is not so easy to do, mainly because, as mentioned above, relational databases are better and because programmers and designers are more used to relational paradigm than to object-oriented when it comes to databases. But there are also several benefits when using object-oriented databases. Imagine complex data, such as financial risk analysis system data, a world wide web document structure or even a hospital patient record system. All these system have one thing in common, that they consist of pretty complex data. This data can be stored and extracted from any kind of database management system, but it is much simpler to do it with a object-oriented database. Everything is stored as objects, so the only thing that a programmer needs to do is to make a persistent call in the programming language he is using. This is possible in relational databases too, using open database connectivity, but the code gets about 30-40 procent bigger. The programmer also do not need to wory about impedance mismatch, because everything are objects, and not relations.

Today, we have several more or less known implementations of object-oriented database management systems. There also exist several ODBMSs in use in production applications. Some of the more known are:

> *British Telecommunications* uses the Versant ODBMS for its integrated fraud management system [18].
>
> *SouthWest Airlines* uses ObjectStore for costumer who prefer to purchase tickets through the internet [65].

West McLaren Mercedes uses Jasmine for monitoring their formula 1 car's performance [19].

There also exist some open source ODBMS implementations like Ozone [67] and Goods (a Russian university project [28]).

6.4 Native XML Databases

XML was developed in the late 1990s, and soon projects for combining XML and databases also started. Pure XML is actually a database that is stripped of a transactional system and has no application interface. In a way this database is rather a step back, than a step in the future, since it is almost a pure hierarchical database which has their origin from 1950s and 1960s. Native XML databases are just to do one thing, mainly to store XML documents. If one desires to store other types of data, it is probably a better idea to use some other kind of database.
After some years came the first Native XML databases. Such a database can store XML, and that is pretty much the only purpose of it. Most native database implementations today, are pure open source implementations. Two of the most known are Apache's Xindice [89] and Wolfgang Meiers eXist [93]. They are more scientific applications than ones that are used in a real world environment, although they will work quite well if there is litle load on the application server and transactions are handled by the programmer.
Native XML databases store XML documents as some kind of persistent and compressed DOM, sometimes even other databases are used. They are, compared to other databases quite slow, but very good at handling XML documents and XML data. There are several things which most native XML databases have in common:

XML Storage - often data is stored in some kind of 'parent' XML document

Collections - many native XML documents manage collection, allowing the programmer to query and manipulate documents as sets

Queries - many implement XPath [53] and XQery [54], which are XML's query languages

These databases can be used in potentially every applications which include some kind of document storage, like catalog data, personal information in companies, some kind of spare parts database, and book or magazine libraries. They are lesser suited for storing other data structures that are not XML based.

6.5 How this is related to my thesis

The technologies presented above are part of the solutions presented in the chapter about the application. It is important to know how things are related to each other, before describing solutions.

7 Architecture

What is architecture? Architecture is a common word on building things, houses, roads, power plants, software, cars and many other things. In software engineering we have two main kinds of architecture, client side and server side. In web programming context it means that an client side application is an application that is processed in the client's browser. Server side application is an applications that is processed on the server.

7.1 Client side Architecture

Client side architecture consist of applications that can be processed on the client side. If we are thinking of web development, this is a web browser. A web browser is a relativity thin client and should not process heavy code. A web browser have mostly been used to process raw HTML, but during the lastest years much more has been added. The newest browsers on the market have integrated support for several new specifications like:

JavaScript A scripting language to capture input motion and input in the browser [41]

VBScript Virtual Basic Script is a Microsoft variant of JavaScript [83]

Flash A client side language from Macromedia [24]

Scalable Vector Graphics (SVG) A vector graphics specification from W3C [29]

Java Applets Small Java components that runs in users browser [10]

Hyper Text Markup Language (HTML) Pure HTML as wee see and use on different web sites

The problem with them is the fact that they consume much memory and processing time. This processing time makes page load more time consuming, this is specially crucial if a page consist of heavy graphics and large client side scripts. On the other way, client side architecture implementations are ideal to check for user input (JavaScript or VBScript) or show graphics that are processed on the client side (like SVG). Pure HTML is actually also a client side language. It relies on the http protocol to receive responses, which are translated to web pages wee see through our web browser.

7.2 Server side Architecture

Server side architecture application is as mentioned above, any applications that are processed on the server. This includes both databases and other applications, but since databases where discussed in the previews chapter, I will not talk much about them. Web servers are another example of server side architecture application. A typical Web server, such as Apache httpd Server [70] or Internet Information Server (IIS) [73], gets a request from a browser, does some processing and returns a response as a HTML page. These Web pages are called static web pages, because their content does

not change from request to request. Today we have many other solutions which make Web pages more dynamical. A dynamic web page is a page which returns an HTML file according to the kind of request that was sent to the server.

7.2.1 Modern Server Side Features - Component Based Architecture

Most modern web sites or applications that uses HTML as their user interface, runs some kind of web server with embedded support for one or more programming languages. Today many also use an approach that is called Component Based Architecture, which means that their applications consist of components. Components are applications, that usually are compiled and can be plugged in any kind of Component Based Architecture application or module. They are reusable, converting them between different applications is fairly easy, because they usually only need to be configured for the current environment. This make software development and maintenance much simpler, and allows in a much large scale to develop independent pluggable software modules that can be sold or reused by other companies or organizations. Today there two different Component Based Architecture specifications, developed by two different vendors.

Java Enterprise Edition J2EE developt by Sun Microsystems [20]

.NET Architecture developt by Microsoft [64]

Since my thesis mainly is focusing on open source software, only J2EE will be discussed. Microsoft .NET framework is currently only implemented by Microsoft. The reason is quite simple, Microsoft has not published their specification and will probably never do it. Sun on the other side has all their specifications freele available on their net site [59].

7.2.2 Java Enterprise Edition (J2EE)

J2EE consist of several smaller specifications like

Java Servlets [77]

Java Naming and Directory Interface (JNDI) [63]

Enterprise JavaBeans (EJB) [11]

Java Database Connectivity (JDBC) [43]

Java Server Pages (JSP) [68]

Java Message Service (JMS) [76]

Java Remote Method Invocation (RMI) [38]

Java Transaction API (JTA) [9]

and some others. They all need to be run on servers and use the same classes and libraries that the usual Java environment (called officially the Java 2 Standard Edition or just J2SE [21]). Since the J2EE specification is a relative open specification it is implemented by many vendors, both commercial and open source. But there are not many vendors that implements the whole specification, they usually implement Java Servlets and JSP or just the EJB specification, the other specifications are often implemented in all servers. JNDI is used to locate object in the server or servers, and JDBC is used for database connection and queries. There is a message service integrated into the J2EE called Java Message Service (JMS). JMS is used as a messaging standard that allows application components based on the J2EE to create, receive and read messages.
One very important thing to mention it the fact that J2EE implements the Model-View-Controller pattern. This pattern is widely used to design server side applications and modules. Pattern was invented and first used by Xeroc Parc employe Trygve Reenskaug in the late 1970s to implement a server side application, and became widely used after that. The pattern consist of only three parts - the Model, the View and the Controller. The idea is to split our application in those three parts. The Model is representing the different sessions and states in our application, often it also is connecting to a database. The Controller part is used to intercept all URL and redirects them to the appropriate model if necessary. The View part is the presentation layer of our application, or can sometimes also be called a 'Graphical User Interface' (GUI).

Java Servlets and JSP - Controller and View Java Servlets and JSP are probably the most implemented and used J2EE specifications. One of the reasons for that is the fact that they allow dynamic web pages and are quite easy to learn. The difference between Java Servlets and JSP is that Java Servlets are the Controller in the application, while JSP is the View. View is, as mentioned above, the GUI, which in this case is pure HTML sent to the client. JSP is kind of a scripting language which links HTML and Java. One can use HTML tags, and pure Java code in a JSP script. This script is then translated to a Java Servlet and compiled during the first request to server. Java Serlvets are the Controller part, and they control the flow of information, redirecting different responses and requests. Before Sun came up with JSP they used Java Servlets as both the Controller and View part, but since there was not any clear distinction between a Controller and a View (logic and presentation), this was not a good solution. JSP was introdused, but this does not solve the problem in a good way. Although there is a possibility to use an approach with both a Controller and a View using Java Servlets and JSP, there is nothing in the way to use just JSP. This problem may seem very trivial, but it does not stop the developer to put everything into a JSP script. In other words to both put logic and presentation into one script, which in fact is not a good thing. A solution to this problem is to restrict a programmer to put everything in one place, that is not possible with JSP and perhaps the biggest short come if this specification. A better alternative is to use some kind of WebMacro [86] implementation. WebMacroes is a open source template language to separate logic and presentation, which

also restrict the programmer or developer from puting too much Java into the script. There exists several implementations of WebMacroes, some of the more known are:

WebMacros own implementation

Apache's Velocity Project See [84] for more info

Cortrek's Coreportal Cortrek have a Machete server which implements the original WebMacro specification [17].

These implementations are external libraries that most often are packed as jar files, copied and configured into a J2EE server. There exist several J2EE servers that implement both the Java Servlets and JSP. Probably the most complete list of J2EE enabled servers can be found on The Serverside page [58], I will mention the most used ones that implements Java Servlets and JSP.

Apache's Tomcat Open source application [81]

Mort Bay Consulting's Jetty Open source application [44]

Bea's WebLogic Comersial application [85]

Oracle's Application Server 10g Commercial application with a Oracle database enabled [16]

Sun Java System Application Server Sun's commercial J2EE server [74]

Enterprise JavaBeans (EJB) - Model Enterprise JavaBeans (EJB) is the model part of the Model-View-Controller pattern. EJB is used handle all logic, database access and transaction management in a simple way. The newest EJB specification consist of three parts, mainly Session beans, Entity beans and Message-Driven beans. A bean is small independent component which can easily be deployed on any EJB enabled server. Session beans are actually divided into two kind of beans, stateless and satefull. Stateless session beans are components which do not have a state, it means that they do not change during requests from other beans or other components. Statefull session beans are components that have a state, and remember the last request or call from a bean or a component. Entity beans are beans which are used to store data and are persistent (like a database). A server which is EJB enabled uses an internal mapping to map entity bean objects into a database. Most EJB enabled servers offers mappings to several different databases, both relational and object-oriented. The third and last bean type is the Message-Driven bean, which actually is quite different from the two others. This bean use the Java Message Service to handle different type of message and queues. Message-driven bean is a new thing in EJB a first came with the EJB 2.0 specification.
There exist several EJB enabled servers, some of them do also implement Java Servlets and JSP. A complete list can be found, as mentioned above, on The Serverside page [58].

Apache's Geronimo Open source EJB enabled Server, can also be enabled with Tomcat or Jetty (the other server acts as a container) [27].

ObjectWebs JOnAS Open source EJB enabled Server, can also be enabled with Tomcat or Jetty (the other server acts as a container) [45].

*Professional Open Source Company's Jboss*Open source EJB enabled Server, can also be enabled with Tomcat or Jetty (the other server acts as a container) [42].

Bea's WebLogic Commercial application server, both a EJB and Java Servlets and JSP server.

Oracle's Application Server 10g Commercial application with a Oracle database enabled, and both a EJB and Java Servlets and JSP server.

Sun Java System Application Server Sun's commercial J2EE server (as above, but with no database).

7.3 How this is related to my thesis

The technologies presented above are part of the solutions presented in the next chapter. It is important to know how things are related to each other, before describing solutions.

8 Biological Model Repository

This is the main chapter in this thesis, showing and describing a Biological Model Repository (BMR). The BMR is a model repository for Systems Biology Markup Language models. Currently only available for level 2 version 1 models, but can be modified to store other levels, and even models in other languages. Three solutions will be presented and discussed, but only one is implemented.

8.1 Specification

The idea is to make a repository for biological models. A repository where scientists and students or other interested people can freely browse and both download and upload new models or edit existing ones. Such repository can be placed on one server, and used by all clients around the world. It could both be implemented using commercial or open source applications, but I have chosen to use open source tools since they are free and well documented, it is better to use it.
Someone once said that a picture is better than a thousand words. Unified Modeling Language (UML) is that kind of modeling language which is more suited for describing a system than any kind of plain language spoken and written by humans [52]. It is more precise, and much better for design of large computer systems.

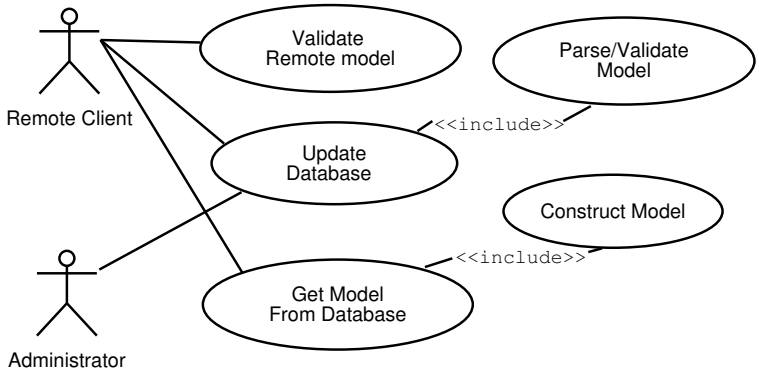

Figure 3: Use case over Biological Model Repository

On the figure above we see a use case over the BMR. It shows us the different aspects that will be covered by this application. The main point with a model repository is to have a persistent storage for large amount of biological and chemical models. This use case shows that we need a way to control this storage by using some kind of updating system for storing models, some kind of querying system to get a certain model and some kind of editor to construct new and edit existing models. Additionally we will also need a system to validate our models, and even to translate them into other languages or formats. The BMR will also have two kinds of end users, some will have the opportunity to administrate both the server and

repository, while others will only use it to store and receive models.

The next step after showing system requirements with a use case is to design a more concrete object model. Such an object model will show different packages and their communication with each other. The figure below is showing an object model over BMR that is encapsulated into smaller packages.

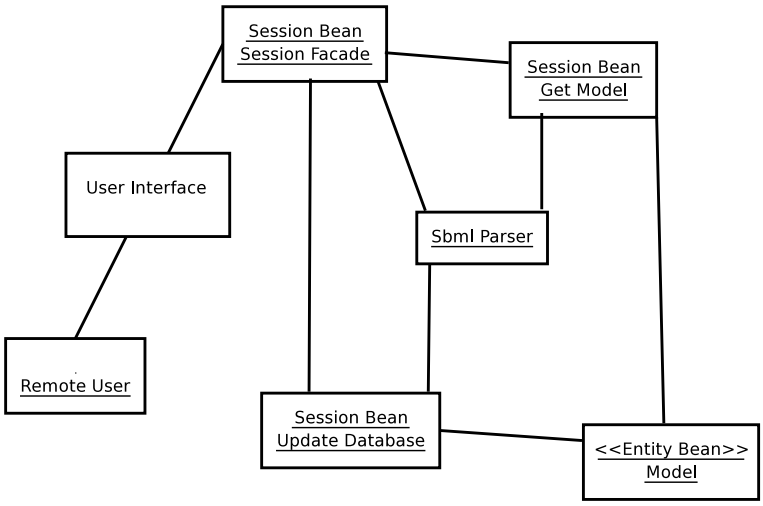

Figure 4: Object Model showing BMR encapsulated into smaller packages

As we see from the object model, it is divided into six smaller packages. These are

Remote User

User Interface

Session Facade

Get Model

Update Database

Model

SBML Parser

Remote User This package represents a web browser. It can be any kind of browser, and shows how it can communicate with the BMR. It interacts only through a 'User Interface', communicating directly with it. Since this thesis is not about browser specifications or implementations, this package is only used to illustrate how a user can interact with BMR.
The rest of the packages can be divided into three parts, as in the Model-View-Controller pattern. In this case 'User Interface' consist of a View and a Controller. The rest of the packages are the Model part of the pattern, representing the logic and the database.

User Interface User interface consist of two packages, one which is a View object and the other is a Controller object. The View object is responsible for user interface, while the other is controlling the information flow between the actual user interface, a web browser and the rest of the application.

Session Facade This facade represents a controlling unit for the Model, controlling input and output. All communication to user from and to database is going through this object. A facade is implemented to make refactoring and maintaining easier. If some one desires to implement a feature that will allow BMR to handle other markup languages like CellML or CML, a natural thing to do is to start with this object. A reference to the new package is created her, and every communication to and from this package is through this controlling unit. A facade is a natural thing in bigger applications and is widely used. Without a facade every method pointing to a service served by the application had to be located in the respective class (in a facade every method pointing to a service is located in the facade). It works, but in case of refactoring or adding new functionalities, it is easier to update methods in one place.

Get Model Get Model represents objects that are used to get model or models from the database. One model can be returned, or a list of all models. Returned models are parsed, so this object gets raw SBML as output. This package communicates with a facade, database and a SBML parser.

Update Database Update Database represents objects that are used to update database with new models, or edit existing ones. This package communicates with a facade, database and a SBML parser.

Model Model represents the actual representation of a SBML model in a database. All models are stored inside this structure, and almost all other packages communicate directly or indirectly with this package. This is the SBML model.

SBML Parser The SBML Parser package is a utility library to assist with translating SBML documents into objects that can be manipulated by other programs. This library is used by all of the reading and writing packages in this application.

8.2 Choice of technologies

In order to make an implementation of the specification above, we need to choose which technologies we will use. This is a crucial task, because wrong choices can have huge impact on the implemented application. Application can have serious bottlenecks in vital parts, or it may only work with certain libraries or features. This is especially crucial when designing distributed applications or applications that may bee used by large group of people. The BMR can potentially is that kind of application. It can be used by large

group of people, and due to high load, it may be distributed across several servers. This application have certain requirements to chosen technologies, they need to be:

Scalable

Ditributed

Secure

Easy to maintain

Easy to upgrade

Configurable

Supportalbe

These are the most important requirements that our chosen technologies need to fulfill in order to make them applicable in the BMR.
In order to support persistent storage of SBML models we need some kind of database. We need a server application that can communicate with both a web browser (i.e. to output HTML) and a database. This needs to handle some logic, in other words needs to be able to communicate with third party libraries. These libraries can be written by the BMR developer or can be different utilities written by independent developers. Below I will present three solutions, based on the above requirements, and discus pros and cons with. All solutions will be presented with open source servers or libraries. One solution is implemented and is available for downloading on the internet.

8.2.1 Solution I

The first solution presented in this thesis is probably the simplest, and use few components and specifications. The idea is to make it as simple as possible using only a combination of three technologies. Probably the simplest *GUI* can be made by the help of HTML mixed up with a second language to handle the underlying logic. This logic can be handled by php, which is an ideal language for such a task. Php have also the possibility to communicate with a database, both MySQL and PostgreSQL are supported. XML databases can not be used, because all open source databases are written in Java, and do not provide an API for php. Both HMTL and php can be run on an Apache Web server (Apache httpd server). The database could be installed on the same server, or on another.

8.2.2 Solution II

The second solution is more complex than the first one, and is using more technologies and specifications. The idea is to use JSP for GUI programming. JSP enables use of Java and enables linking with other J2EE technologies. In order to get the full advantage of J2EE technologies there is a good idea to use Java Servlets for handling requests and responses, and

use EJB to handle business logic. Since EJB are used, a relational database should be the preferred database to use. EJB use entity beans to handle persistent and session beans to take care of the business logic. A new and interesting feature with entity beans is the ability to use container persistence. Container persistence enables a programmer to let the server handle relational to object mapping. The programmer only writes an entity bean, which on booting is translated to a class that handles the relational to object mapping. Jboss server integrated with a *Tomcat container* handles container persistence, so a natural thing is to choose a relational database like MySQL.

8.2.3 Solution III

The third and last solution is a more scientific and experimental approach. The idea is to store a SBML model in its natural environment, a native XML database. Since, as mentioned above, the existing open source databases only supports Java, this language must be used to handle business logic. The best way to handle both the business logic and to make an GUI is to only use a JSP and Java Servlets running on a *Tomcat server*. There exist several open source XML databases, and the solution is to use Apaches Xindcie or eXist.

8.2.4 Discussion around solutions

The question is which of these solutions to use? There is probably not any good answer to this question, since they all have strong and weak sides. Much also depends on the programmer who is to implement on of those solutions. If he is very good and has a worked long time as professional using one of the technologies, a solution which use this technology is the preffered one.
If we look closely at the three solutions we will soon find out that they are chronological ordered. From the oldest paradigm to the newest. The paradigm behind the first solution is the oldest one. Combining HTML and a script language that both handles logic and database access is about from mid 1990s, making it a kind of old solution to use. The third and last solution a more experimental approach to use. Most native XML databases that are used today (both commercial and open source) has implementations that are not high in version numbering. Both eXist and Xindcie have not yet passed version 1.0. On the other hand seems the second solution to be too complex and difficult for such a simple task. There also exist several other issues to take into consideration, like security, support, scalability, and ans. The point is that there is no clear solution or answer when it comes to solution consideration.

8.3 Implementation details

I have chosen to use solution II for the implementation of my BMR. There are several reasons why I have chosen to use this solution. J2EE paradigm is widely used today, and is probably the best solution among the other

two. I also have more experience using Java, than using php or any other scripting language.

8.3.1 A word about configuration of J2EE applications

In order to run applications on J2EE enabled servers one needs to configure them in a proper way. Configuration is done by using and editing configuration files, and restarting the server. If the application is implemented in a proper way and all standards are followed, its easy to switch vendors, because only the configuration needs to be changed.

Configuring applications for Jboss In order to run an application on a Jboss server, it needs to be configured using an ejb-jar.xml file and packed as a *jar file*. This configuration file gives the server information about beans that he will deploy. Sometimes the server also requires a jboss specific configuration file (jboss.xml) if the configuration is to needs to be more specific. The ejb-jar.xml file is part of the J2EE specification. Both files can be found in the appendix of this thesis, and will be further discussed below.

Configuring applications for Tomcat Applications running on Tomcat do also need to be configured. This is happening in the way as for Jboss applications, but the main configuration file is named web.xml. If there are some configurations that needs to be more server specific the jboss-web.xml is used. However, only the web.xml is part of the J2EE specification. Both files can be found in the appendix of this thesis, and will be discussed further below.

8.3.2 Code organization

I have tried to organize code in a more or less logical structure, using the same code organization as presented by Floyd Marinescu [56].
The top package has the same name as the acronym used for the Biological Model Repository - bmr. Ideally it should begin with the url for the organization I am working for, url for the department I am working for, url for section in the department I am working for and them the acronym for the project name. So correct code organization for the top level package should have been no.uio.ifi.sc.bmr or no.simula.sc.bmr (sc is scientific computing). Due to simplicity I have chosen to just use bmr.

bmr.db Entity Beans - The Database

bmr.ejb Top level package for Enterprise Java Beans - The logic

bmr.ejb.facade Facades

bmr.ejb.service Service classes - classes offering services for Session Facade

bmr.ejb.sbml2 SBML2 Session Beans

bmr.ejb.sbml2.interfaces Intefaces for SBML2 Session Beans

bmr.utils Top level package for different utilities

bmr.utils.dbkeys Key generating classes for Entity Beans

bmr.utils.sbml2 SBML2 utilities

bmr.web Top level package for Web - The GUI/View and Controller

bmr.web.txt Package containing txt files like JSP scripts and HMTL files

Since most of the code organization is similar to the object model for packages and for use cases, it will be presented in the same way further below. The only difference is the bmr.utils package, which is not show on any model over, but will be presented as the first package in the next subchapter. All code fragments presented below are available in the appendix and in the code available to download on my web page [34].

8.3.3 Utility classes for other packages

In order to make a functional application there is always a need to have some kind of utility classes available. These classes do things like defining lists (for example like the java.util.List interface), generating database keys, extracting common information from models, ans. Most of todays software packages or applications have some kind of utility packages.
During the development process of the bmr there have emerged some new needs for common utility classes like:

A database key generator

Database keys for Entity Beans

A class for extracting MathML from an SBML model

A static class for SBML model specific parameters

A database key generator Since every database key has to be unique I have chosen to make a key generator class. This class is a modified example from Floyd Marinescus book (EJB Design Patterns [56]). Below is a code fragment showing parts of the key generator method:

```
public synchronized String getUUID(){
  ...
  InetAddress inet = InetAddress.getLocalHost();
String ipAdr = inet.getHostAddress();
int idHash = System.identityHashCode(this);
String thisHash = String.valueOf(idHash);
SecureRandom seeder = new SecureRandom();
int random = seeder.nextInt();
String secureRandom = String.valueOf(random);
long timeNow = System.currentTimeMillis();
String currentTime = String.valueOf(timeNow);
uuID = currentTime + ipAdr + idHash + secureRandom;
...
```

The idea is to use a combination of several variables to make a unique key. The variables are the ip address for the server, a hashcode for the current object, a random number and current time in milli seconds. By using these

variable we should achieve an almost unique key even if we are running a cluster of servers or several objects that are trying to access this method concurrently. The uuID variable is a string that contains a unique identifier.

Database keys The idea is taken from a book about EJB by Ed Roman (Mastering EJB [71]). In order to generate an Entity Bean we need a class that is serializable to hold on the key. A common Java String class can be used, but it is better to use a self constructed class that implements a serializable object. There are no special methods in this class, so I will only show it in the appendix. Every Entity Bean has its own database key class. The name convention for database key holders is the name of the Session Beans local object interface with a 'PK' end. The Session Bean communicates with the Entity Bean which is the database (see below for more information).

Sbml2 Math class The Sbml2Math class is located inside the bmr.utils.sbml2 package. It is designed for very simple MathML extraction from the SBML2 model. Below is a code fragment that shows parts of the getMath method:

```
...
  TransformerFactory xformFactory = TransformerFactory.newInstance();
  Transformer transformer = xformFactory.newTransformer();
  Source input = new DOMSource(node.getChildNodes().item(i));
  stringWriter = new StringWriter();
  Result output = new StreamResult(stringWriter);
  transformer.clearParameters();
  transformer.transform(input,output);
  math = stringWriter.getBuffer().toString();
...
```

In order to get the MathML document I used the Xerces and Xalan libraries to extract it from the SBML2 model. The math variable at the bottom is a String that will be converted to an object before it will be returned to the caller class.

Sbml2 Parameter class The SbmlParameter class is located inside bmr.utils.sbml2 package. It only consist of static strings that represents the SBML2 model. For more info take a look at it in the appendix.

8.3.4 The Database - Entity Beans

The database consist of several Entity Beans, these are located in the bmr.db package. In order to use Entity Beans they must be configurated in the ejb-jar.xml file and must consist of three classes. I will describe the Event Bean class, its interfaces and the configuration in the ejb-jar.xml file.

The EJBLocalHome interface The EJBLocalHome interface is used to define callable methods in that are associated with the current bean. Every communication between the parent bean and other beans are throught this interface. In other words is this interface used to define callable methods for the parent Entity Bean. Below is a code fragment from the bmr.db.EventLocalHome interface showing several callable methods:

```
...
EventLocal create(String eventID,String eventName,
Object trigger,Object delay,String timeUnits,
String modelID) throws  CreateException;
public EventLocal findByPrimaryKey(EventPK key) throws FinderException;
public Collection findEvents(String modelID) throws FinderException;

...
```

In order to insert data into the Entity Bean (and the database) we need to call the create(..) method in this interface. The create method takes several variable as input (see below) and returns an EventLocal object (see next paragraph). There also exists two other methods, a finByPrimaryKey method (a key object as input) that returns an EventLocal object (see below) and a findEvents method (a modelID as input) that returns a java.util.Collection (see below).

The EJBLocalObject interface This interface defines methods for the Enity Bean (and tables in the database). Below is a code fragment from the bmr.db.EventLocal interface:

```
...
public abstract String getEventID();
public abstract String getEventName();
public abstract void setEventName(String eventName);
public abstract Object getTrigger();
public abstract void setTrigger(Object trigger);
public abstract Object getDelay();
public abstract void setDelay(Object delay);
public abstract String getTimeUnits();
public abstract void setTimeUnits(String timeUnits);
public abstract String getModelID();
public abstract void setModelID(String modelID);
...
```

As we see in the example above there are defined get and set methods for all the variables send in the create(..) method above. Since eventID is a key for the parent bean (Entity Bean EventBean described below) it can not be changed once initiated by the create(..) method. The same methods are also defined in the parent bean class.

The EntityBean class This is the Entity Bean, a bean class which is used to communicate with the EJBLocalObject interface and a database. It actually represents the database table for this bean. Below is a code fragment that shows the bmr.db.EventBean:

```
public EventPK ejbCreate(String eventID,String eventName,
Object trigger,Object delay,String timeUnits,String modelID)
throws CreateException{
  setEventID(eventID);
  setEventName(eventName);
  setTrigger(trigger);
  setDelay(delay);
  setTimeUnits(timeUnits);
  setModelID(modelID);

  return new EventPK(eventID);
}
```

This fragment show how variables are put into the bean and later into the database. The set methods described above are found in both this Entity Bean and in the EBJLocalObject interface. During server booting is this

Entity Bean translated into create statements and a table in the database is created. Create statements from the database, that are based on all the Entity Beans in the BMR are show in the appendix.

The configuration in the ejb-jar.xml file In order to make a bean functional on a server (both the Session Bean and Entity Bean) it must be configured properly. Configuration of all beans happen in the ejb-jar.xml, here is a fragment for the configuration of the bmr.db.EventBean.

```
<entity>
<ejb-name>EventLocal</ejb-name>
<local-home>bmr.db.EventLocalHome</local-home>
<local>bmr.db.EventLocal</local>
<ejb-class>bmr.db.EventBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.EventPK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>EventBean</abstract-schema-name>
        <cmp-field>
<field-name>eventID</field-name>
</cmp-field>
<cmp-field>
<field-name>eventName</field-name>
</cmp-field>
    <cmp-field>
<field-name>trigger</field-name>
</cmp-field>
<cmp-field>
<field-name>delay</field-name>
</cmp-field>
<cmp-field>
<field-name>timeUnits</field-name>
</cmp-field>
<cmp-field>
<field-name>modelID</field-name>
</cmp-field>
    <query>
   <query-method>
    <method-name>findEvents</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM EventBean AS o where o.modelID LIKE ?1]]>
   </ejb-ql>
  </query>
 </entity>
```

In the beginning of this fragment we see the bean name, and pointer to where to find the different interfaces and where the bean is located. It also shows us which variables are handled by the bean, name for the table in the database and definition on a finder method. In this case it is the findEvents, with its ejb-ql query. The same finder method is also defined in the EJBLocalHome interface. More information about this finder method will came further below.

8.3.5 The Facades

The Facade in the BMR consist of two facades. One Session Facade and one Message Facade. The difference is that a Message Facade handles messages, while the other one handles normal callups. In order to handle file upload in a more effective way, I have chosen to use a Message Facade. Both implementations will be discussed in this section.

Session Facade Since this Session Facade is a Session Bean that also have two interfaces and a ejb-jar.xml file configuration, I will describe it in the same way as above.

EJBHome interface This is the home interface for the Session Facade. It only represents one create() method which returns an EJBObject. The code is shown in the appendix.

EJBObject interface This interface is similar to the one in the previews chapter. It represents the callable methods that other beans or applications can call. The reader is probably surprised why this is an EJBObject rather than an EJBLocalObject. The answer lies in the fact that EJBLocalObjects can only be called by beans on the same server, while EJBObjects can by called by anyone. I have chosen to implement this that way in order to force a third party programmer or designer to use both of my facades. They are the only interfaces that can be called by any other component. A example below shows a fragment of the bmr.facade.BmrFacade interface which is an EJBObject:

```
...
public String getModel(String modelID) throws RemoteException,EJBException;
public Collection browseDatabase() throws RemoteException,EJBException;
public String getMathAsC(String modelID) throws RemoteException,EJBException;
public String getMathAsXML(String modelID) throws RemoteException,EJBException;
...
```

The methods above are the services provided by this application. Only the getModel(..) and the browseDatabases() are implemented. A method to update the database is implemented as part of the Message Facade.

Session Bean Session Bean is the bean where all the logic is handled. It implements all the methods of the EJBObject interface. Since code fragments of it will be show in the sections below, I will not discuse it in this paragraph. The complete code of this Session Bean (bmr.facade.BmrFacadeBean) is shown in the appendix.

The configuration in the ejb-jar.xml file In order to make a bean functional on a server, we need to configurate it properly. This configuration is done in the ejb-jar.xml file. Below is a part of this file concerning only a part of the Session Facade configuration:

```
<session>
   <ejb-name>BmrFacade</ejb-name>
   <home>bmr.ejb.facade.BmrFacadeHome</home>
   <remote>bmr.ejb.facade.BmrFacade</remote>
   <ejb-class>bmr.ejb.facade.BmrFacadeBean</ejb-class>
   <session-type>Stateless</session-type>
   <transaction-type>Bean</transaction-type>
    <ejb-local-ref>
    <description>Reference to SbmlFile</description>
    <ejb-ref-name>ebj/bmr/ejb/service/SbmlFileLocalHome</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <local-home>bmr.ejb.service.SbmlFileLocalHome</local-home>
    <local>bmr.ejb.service.SbmlFile</local>
    <ejb-link>SbmlFile</ejb-link>
    </ejb-local-ref>
     <ejb-local-ref>
```

```
        <description>Reference to database</description>
    <ejb-ref-name>ejb/bmr/db/ModelLocalHome</ejb-ref-name>
    <ejb-ref-type>Entity</ejb-ref-type>
     <local-home>bmr.db.ModelLocalHome</local-home>
   <local>bmr.db.ModelLocal</local>
   <ejb-link>ModelLocal</ejb-link>
    </ejb-local-ref>
   </session>
```

The almost only difference from the configuration above are the links to other beans. We see one link to a Entity Bean (ModelLocal) and the other to a Session Bean (SbmlFile).

Message Facade A Message Facade is implemented to handle the file upload in a more efficient way. If a Session Bean had bean used, it would require more time spend on waiting for the parsing and inserting of a model. A Message Facade (which is a Message Driven Bean) only gets the model, and sends it away. After the model has been send away, the bean is terminated and the client application (which send the file) is free to do other things. A background process initiated by the Message Driven Bean is performed until the model as been parsed and inserted. I use the Message Facade to do that kind of processing, an example will be given in the 'Update Database' section of this chapter. Message Driven Beans only consist of one file, but require to be configurated in both the ejb-jar.xml file and in the jboss.xml file. This is probably due to jboss requirements.

The configuration in the ejb-jar.xml file This is a example of the ejb-jar.xml part that configures the Message Facade, which in fact is only one file, bmr.ejb.facade.BmrFacadeMDB.

```
<message-driven>
    <ejb-name>BmrFacadeMDB</ejb-name>
<ejb-class>bmr.ejb.facade.BmrFacadeMDB</ejb-class>
<transaction-type>Container</transaction-type>
<message-driven-destination>
   <destination-type>javax.jms.Queue</destination-type>
</message-driven-destination>
  <ejb-ref>
        <description> Reference to InsertIntoDbBean </description>
        <ejb-ref-name>ejb/bmr/ejb/service/InsertIntoDbLocalHome</ejb-ref-name>
        <ejb-ref-type>Session</ejb-ref-type>
        <home>bmr.ejb.service.InsertIntoDbLocalHome</home>
        <remote>bmr.ejb.service.InsertIntoDb</remote>
        <ejb-link>InsertIntoDb</ejb-link>
  </ejb-ref>
</message-driven>
```

A part from defining a reference to an other bean (InsertIntoDbBean discused below), it also shows what kind of message system this is. This bean is using a javax.jms.Queue to send messages.

The configuration in the jboss.xml file Below is an example of a jboss.xml file that configures the bmr.ejb.facade.BmrFacadeMDB:

```
<?xml version="1.0" encoding="UTF-8" ?>
<jboss>
  <enterprise-beans>
    <message-driven>
      <ejb-name>BmrFacadeMDB</ejb-name>
      <configuration-name>Standard Message Driven Bean</configuration-name>
      <destination-jndi-name>queue/BmrFacadeQueue</destination-jndi-name>
```

```
    </message-driven>
  </enterprise-beans>
</jboss>
```

It shows how a jndi name is associated with a queue message type.

8.3.6 Update Database

This service is a bean that gets messages from the Message Facade, parses them, and inserts models into the database. The SBML parser functionality will be shown in the SBML Parser section.
First a model is received in the onMessage method. Then the bmr.ejb.service.InsertIntoDbBean is lookedup (see example below). After the lookup is the SBML2 model object send to the bmr.ejb.service.InsertIntoDbBean.

```
public void onMessage(Message msg) {
  if (msg instanceof StreamMessage){
try{
StreamMessage bm = (StreamMessage)msg;
Object o = bm.readObject();
System.out.println(o);
insertIntoDbLocalHome = lookUpInsertIntoDb();
InsertIntoDb insertIntoDb = insertIntoDbLocalHome.create();
insertIntoDb.upDateDb(o);
}catch (JMSException jms){
jms.printStackTrace();
}catch (CreateException c){
c.printStackTrace();
}catch (RemoteException r){
  r.printStackTrace();
}
}
}
```

A method to lookup the InsertIntoDbBean.

```
private InsertIntoDbLocalHome lookUpInsertIntoDb(){
Object objref = null;
try{
Context initial = new InitialContext();
objref = initial.lookup

("java:comp/env/ejb/bmr/ejb/service/InsertIntoDbLocalHome");
}catch (NamingException n){
throw new EJBException(n.getMessage());
}
return (InsertIntoDbLocalHome)objref;
}
}
```

In the bmr.ejb.service.InsertIntoDbBean is the object translated to a ByteArrayInputStream and send for SBML parsing (the getDocument(..) method). SBML parsing and the insertion into database is described in a section below.

```
public void upDateDb(Object o)throws EJBException,RemoteException{
byte []bytes = (byte[])o;
ByteArrayInputStream byteInStream = new ByteArrayInputStream(bytes);
Document doc = getDocument(byteInStream);
parseModel(doc);
System.gc();
}
```

After parsing and insertion, a garbage collector (System.gc()) is called to clean up. Java has an automatic garbage collection (gc), but sometimes it is better to force gc to clean up the memory.

8.3.7 Get Model From Database

This service is used to get SBML2 models from the database. It consist of two services, one that returns all models, and the other is returning one SBML2 model based on the modelID which is send with the method.

Show All Models This is probably the simplest method in the entire application. It is based on the browseDatabase() method from the Session Facade. A collection is returned that consist of all model names, and all modelids in the database. This is illustrated in the code fragment below:

```
public Collection browseDatabase() throws RemoteException,EJBException{
Collection co = null;
try{
modelLocalHome = lookUpModelLocalHome();
co = modelLocalHome.findAllModels();
}catch (FinderException f){
f.printStackTrace();
}

return co;
}
```

The findAllModels() method in the ModelBean is actually an *ejb-ql* expression. This expression is located in the ejb-jar.xml file. This query language is similar to the SQL but it is more a *OQL* variant. The query used to get a java.util.Collection of SBML models is shown below:

```
...
<abstract-schema-name>ModelBean</abstract-schema-name>
   <cmp-field>
<field-name>modelID</field-name>
</cmp-field>
<cmp-field>
<field-name>modelName</field-name>
</cmp-field>
<query>
   <query-method>
    <method-name>findAllModels</method-name>
        <method-params></method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM ModelBean AS o]]>
   </ejb-ql>
 </query>
...
```

All queries in this language are associated with the description in the ejb-jar.xml file. In this case all objects that are not empty are returned.

Get Specific Model The getModel method that takes a modelID as input returns a string. This string is a complete SBML2 model. The code to get the specific model starts in the bmr.ejb.facade.BmrFacadeBean, and is show below:

```
public String getModel(String modelID) throws RemoteException,EJBException{
String model = null;
try{
SbmlFileLocalHome sbmlfileLocalHome = lookUpSbmlFileLocalHome();
SbmlFile sbmlFile = sbmlfileLocalHome.create();
model = sbmlFile.getSbmlModel(modelID);
}catch(CreateException c){
c.printStackTrace();
}
return model;
}
```

As input we are getting a modelID. The bmr.ejb.service.SbmlFileBean is called and a modelID is sent to it. The code for bmr.ejb.service.SbmlFileBean concerning the input from the previous code is shown below:

```
public String getSbmlModel(String modelID) throws EJBException {
File f = null;
sbmlString = SbmlParameters.XML_FILE_INFO;
sbmlString += "\n";
sbmlString += SbmlParameters.SBML_FILE_BEGIN+"\n";
sbmlString += "<model id=\""+modelID+"\" ";
if(getModelName(modelID) != null)
sbmlString += "name=\""+getModelName(modelID)+"\"> \n";
else
sbmlString += "> \n";
/** listOfFunctionDefinitions **/
sbmlString += getFunctionDefinitions(modelID);

/** listOfUnitDefinitions **/
sbmlString += getUnitDefinitions(modelID);

/** Compartments **/
sbmlString += getCompartments(modelID);

/** Species **/
sbmlString += getSpecies(modelID);

/** Parameters **/
sbmlString += getParameters(modelID);

/** Rules **/
sbmlString += getRules(modelID);

/** Reactions **/
sbmlString += getReactions(modelID);

/** Events **/
sbmlString += getEvents(modelID);

sbmlString += SbmlParameters.SBML_FILE_END;

return sbmlString;
}
```

This is the actual method that constructs an SBML model from the database. All of the get methods returns a string containing a part of the model. One of the get methods is shown below:

```
private String getEvents(String modelID){
String events = null;
try{
eventsHome = (EventsHome)lookUpGeneral
("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/EventsHome");
Events ev = eventsHome.create();
events = ev.getEvents(modelID);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException f){
f.printStackTrace();
}

return events;
}
```

In the code fragment above are all events returned based on a input from the current modelID. The code for the EventBean that is returning the event part of the SBML2 model is shown in the section below.

8.3.8 The SBML Parser

The SBML Parser is split into two parts, one for inserting the model and the other one for returning the model. It uses a DOM implementation to parse

models, that is found in the Apaches Xerces parser project.

Inserting the Model This part is closely related to the 'Update Database' section. After getting the SBML model, the model is parsed. The code below shows a part of that kind of parsing.

```
...
}else if(compartments.item(i).getNodeName().
equalsIgnoreCase("listOfEvents") == true){
for(int j=0;j<compartments.item(i).getChildNodes().getLength();j++){
if(compartments.item(i).getChildNodes().item(j).getNodeName()
.equalsIgnoreCase("event") == true){
try{
if(eventsHome == null)
eventsHome =(EventsHome) lookUpGeneral
("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/EventsHome");
Events event = eventsHome.create();
event.insertIntoDb(compartments.item(i).getChildNodes().item(j),id);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException c){
c.printStackTrace();
}
}//end if
}//end for
...
```

It is more or less messy code but should be understandable. First there is an if test on the current node, to determine if is a 'listOfSpecies'. If that is the case, the EventBean is looked up, and a node with a modelID (the 'id' variable) is send to EventBeans insertIntoDb method.
Inside the EventBean the insertIntoDb(..) method is called. This method is shown below:

```
public void insertIntoDb(Node node, String modelID){
this.modelID = modelID;
if(node.getAttributes().getNamedItem("id") != null)
eventID = node.getAttributes().getNamedItem("id").getNodeValue();
else{
eventID = "empty";
random = new Random();
double num = random.nextDouble();
eventID = String.valueOf(num)+eventID;
}
if(node.getAttributes().getNamedItem("name") != null)
  eventName = node.getAttributes().getNamedItem("name").getNodeValue();
if(node.getAttributes().getNamedItem("timeUnits") != null)
  timeUnits = node.getAttributes().getNamedItem("timeUnits").getNodeValue();
for(int i=0;i<node.getChildNodes().getLength();i++){
if(node.getChildNodes().item(i).getNodeName().equalsIgnoreCase("trigger") == true){
sbmlMath = new SbmlMath();
trigger = sbmlMath.getMathML(node.getChildNodes().item(i));
}else if(node.getChildNodes().item(i).getNodeName().
equalsIgnoreCase("delay") == true){
sbmlMath = new SbmlMath();
delay = sbmlMath.getMathML(node.getChildNodes().item(i));
}else if(node.getChildNodes().item(i).getNodeName().
equalsIgnoreCase("eventAssignment") == true){
try{
eventAssignmentsHome = lookupAssignments();
EventAssignments ea = eventAssignmentsHome.create();
ea.insertIntoDb(node.getChildNodes().item(i),eventID);
}catch(NamingException n){
n.printStackTrace();
}catch (CreateException c){
c.printStackTrace();
}
}//end else if
  }//end for
try{
  eventLocalHome = lookupDb();
```

```
  EventLocal eventLocal = eventLocalHome.
  create(eventID+modelID,eventName,trigger,delay,timeUnits,modelID);
}catch (NamingException e){
  throw new EJBException(e.getMessage());
}catch (CreateException c){
  c.printStackTrace();
}
}
```

The node is then checked for parameters and other nodes. If a MathML node is encountered the SbmlMath class (described earlier) is called, and a object containing MathML is returned. There are also a test for 'eventAssignment'. If it is true, an EventAssignmentBean is called and current node with eventid is send. At the bottom of this method is the database looked up, and updated with variables. Notice how I determine the identificator for the database table. I use current eventID with current modelID, this solution should guarantee a unique identificator. To determine the model of an EventBean I also store the modelID of the model.
This parsing and storing approach is used in all beans. I used a very simple algorithm to determine which Entity Bean is related to which. The top level Entity Beans does not know anything about their childtren, only children have pointer to their parents. This is an abstract tree structure, where every child has its parent identificator and its identificator consist of his identificator plus his parent identificator. By doing so I achieve the possibility that two or more SMBL2 variables in different models having the same identificator can be stored in the database without any problem.

Returning the Model This part of the section is about returning and building up the model from Entity Beans (database tables). It is also very close to the last part discussed in the section above. After the getEvent method in the bmr.ejb.service.SbmlFileBean is called, the getEvents method inside bmr.ejb.sbml2.EventBean is called. This part is shown in the code fragment below:

```
public String getEvents(String modelID){
String eventString = null;
eventString = SbmlParameters.LIST_OF_EVENTS_DEF_BEGIN;
try{
eventLocalHome = lookupDb();
Collection col = eventLocalHome.findEvents(modelID);
Iterator it = col.iterator();
while(it.hasNext()){
eventString += SbmlParameters.EVENT_DEF_BEGIN;
EventLocal eventLocal = (EventLocal)it.next();
eventString += " id=\""+eventLocal.getEventID()+"\"";
if(eventLocal.getEventName() != null)
eventString += " name=\""+eventLocal.getEventName()+"\"";
if(eventLocal.getTimeUnits() != null)
eventString += " timeUnits=\""+eventLocal.getTimeUnits()+"\"";
eventString += ">";
if(eventLocal.getTrigger() != null)
eventString += getTrigger(eventLocal);
if(eventLocal.getDelay() != null)
eventString += getDelay(eventLocal);
eventString += getEventAssignment(eventLocal);
eventString += SbmlParameters.EVENT_DEF_END+"\n";
}
}catch (NamingException n){
n.printStackTrace();
}catch (FinderException f){
f.printStackTrace();
}
eventString += SbmlParameters.LIST_OF_EVENTS_DEF_END;
```

```
return eventString;
}
```

As mentioned above is this method constructing an eventString and is sending it back to caller. This string consist of all events in the current model. The events are returned on the basis of modelID as input. Entity Bean that represents events has a findEvents method which based on the modelID returns all events having the desired identificator (this query is shown further below). The rest of this method is pretty forward, because it only checks if something is different from 'null' and then adds it to the return string. Something similar is happening with the event assignments too. The getEventAssignment(..) method is shown below:

```
private String getEventAssignment(EventLocal eventLocal){
String eventAssignment = null;
try{
EventAssignmentsHome eventAssignmentsHome = lookupAssignments();
EventAssignments eventAssignments = eventAssignmentsHome.create();
eventAssignment = eventAssignments.getEventAssignments(eventLocal.getEventID());
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException c){
c.printStackTrace();
}
if(eventAssignment != null)
return eventAssignment;
else return "";
}
```

As we see is this method also pretty easy to understand. The EventLocal object has an get method which returns an identificator which then is sent to the event assignments Entity Bean. This beans reacts in the same way as the parent class. The code showing the findEvents(..) query inside, which in fact is identical with the findEventAssignments(..) method inside the event assignments Entity Bean, is shown below:

```
<abstract-schema-name>EventBean</abstract-schema-name>
        <cmp-field>
<field-name>eventID</field-name>
</cmp-field>
<cmp-field>
<field-name>eventName</field-name>
</cmp-field>
    <cmp-field>
<field-name>trigger</field-name>
</cmp-field>
<cmp-field>
<field-name>delay</field-name>
</cmp-field>
<cmp-field>
<field-name>timeUnits</field-name>
</cmp-field>
<cmp-field>
<field-name>modelID</field-name>
</cmp-field>
    <query>
   <query-method>
    <method-name>findEvents</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM EventBean AS o where o.modelID LIKE ?1]]>
   </ejb-ql>
 </query>
```

This query is similar to the one above, but the only difference is the input parameter that a string that is a modelID. Adding a input parameter reduces the returned collection, and we get only the event part of the SBML2

model that has the same modelID as the input parameter. This approach is identical in all other beans.

8.3.9 User Interface - GUI

The user interface is split into both a view and controller. Due to a relative small application and non-complex interface, are the boundaries between view and controller pretty small. Both of the HMTL files I have are pure view part, while both servlets and the JSP are some kind of mixture between these two. All of these files are also show in 'The working version' chapter as screen capture image from my computer. However I will explain some of them in this chapter.

Database content JSP This JSP script is calling the browseDatabase method inside the bmr.ejb.facade.BmrFacadeBean. The code that does it is shown below:

```
...
Context initial = new InitialContext();
Object objref = initial.lookup("BmrFacade");
BmrFacadeHome bmrHome = (BmrFacadeHome)objref;
BmrFacade bmr = bmrHome.create();
Collection col = bmr.browseDatabase();
Iterator it = col.iterator();
...
```

This collection is then iterated and we have the ability to click on the desired model (see below for code fragment).

```
...
<a href='ViewModel?sbmlModel=<%=modelLocal.getModelID()%>&type=text'>  Export!</a>
<a href='ViewModel?sbmlModel=<%=modelLocal.getModelID()%>&type=xml'>XML Export </a>
...
```

If we click on the model we will be redirected with the modelID input to the ViewModel Java Servlet. The model download is available in two formats. One as a pure XML file, and the other as a part of a HTML document. The second approach is good use if there exist some HMTL tags inside the SBML model.

View Model Java Servlet This servlet handles the file download from the server. There is only one method that does it, and it is a doGet method that is show below:

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException{
textType = request.getParameter("type");
if(textType.equalsIgnoreCase("text"))
response.setContentType("text/html");
else
response.setContentType("text/xml");
out = response.getWriter();
String model = request.getParameter("sbmlModel");
if(textType.equalsIgnoreCase("text"))
out.println("<html>");
try{
Context initial = new InitialContext();
BmrFacadeHome bmrFacadeHome = (BmrFacadeHome)initial.lookup("BmrFacade");
BmrFacade bmrFacade = bmrFacadeHome.create();
String sbmlModel = bmrFacade.getModel(model);
if(textType.equalsIgnoreCase("text"))
out.println("<p>"+sbmlModel+"</p>");
```

```
else
out.println(sbmlModel);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException c){
c.printStackTrace();
}
if(textType.equalsIgnoreCase("text"))
out.println("</html>");
}
```

This code is pretty simple. We start with getting the BmrFacade using JNDI lookup. Then the getModel(..) method is called and a string containing the model is returned.

Update Database Java Servlet The third and last file discused in this package is used when someone desires to upload a file. The fileUpLaod.html file is used to browse the users directory and letting him choose a file. Then after pushing the submit button the file is send to this servlet. The code for the doPost method is shown below:

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
   throws IOException, ServletException{
  response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
DiskFileUpload fu = new DiskFileUpload();
fu.setSizeMax(-1);
try{
List fileItems = fu.parseRequest(request);
FileItem fileItem = (FileItem)fileItems.get(0);
byte[] fil = fileItem.get();
Properties properties = new Properties();
properties.put("java.naming.factory.initial","org.jnp.interfaces.NamingContextFactory");
properties.put("java.naming.provider.url", "localhost:1099");
   Context context = new InitialContext(properties);
  QueueConnectionFactory factory = (QueueConnectionFactory)context.
lookup("UIL2ConnectionFactory");
QueueConnection connection = factory.createQueueConnection();
QueueSession session = connection.createQueueSession(false,Session.AUTO_ACKNOWLEDGE);
Queue queue = (Queue) context.lookup("queue/BmrFacadeQueue");
   QueueSender sender = session.createSender(queue);
   StreamMessage objMsg = session.createStreamMessage();
  objMsg.writeBytes(fil);
    sender.send(objMsg);
    sender.close();
}catch (FileUploadException fue){
    out.println("<p>"+fue.getMessage()+" FileUpLoadException</p>");
out.println("</html>");
}catch (NamingException n){
out.println("<p>"+n.getMessage()+"NamingException</p>");
out.println("</html>");
}catch (JMSException j){
out.println("<p>"+j.fillInStackTrace()+"JSMExce</p>");
out.println("</html>");
   }catch (Exception e){
   out.println(e.getMessage());
   }
out.println("<p>Writen to database!</p>");
out.println("<p><a href=''>Look at the writen file</a></p>");
out.println("<p><a href='/bmr/index.html'>Back to main menu</a></p>");
out.println("</html>");
}//end doPost
```

In order to reduce complexity with uploading files I have used an Apache Common FileUpLoad package [23]. This package enables upload of files in an very easy way. After getting the file, is it transferred into a byte array and send to the bmr.ejb.facade.BmrFacadeMDB as a stream message.

8.4 The working version

The BMR that I have designed and coded in this thesis do work, and this chapter will show some screen shots from it taken on my computer.

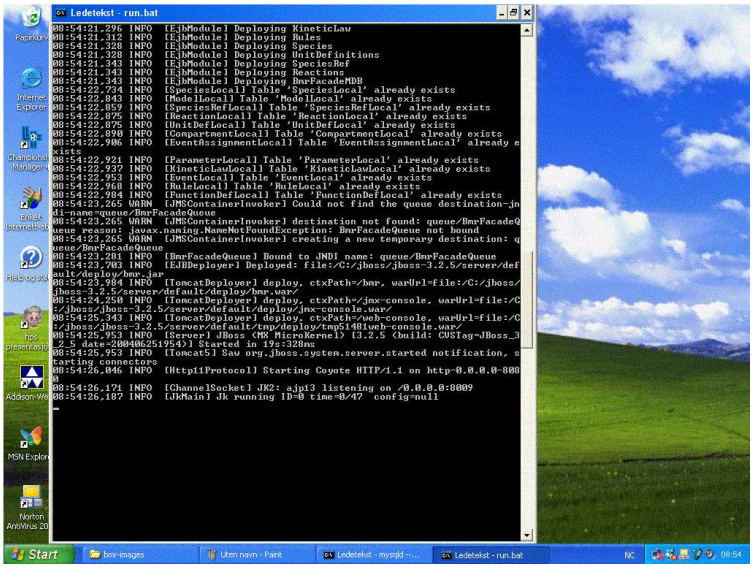

Figure 5: Booting of the BMR server

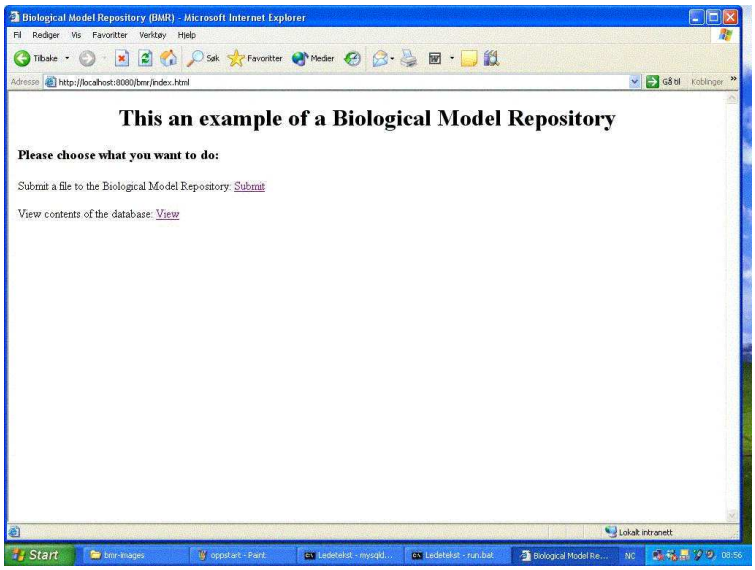

Figure 6: Index file of the BMR

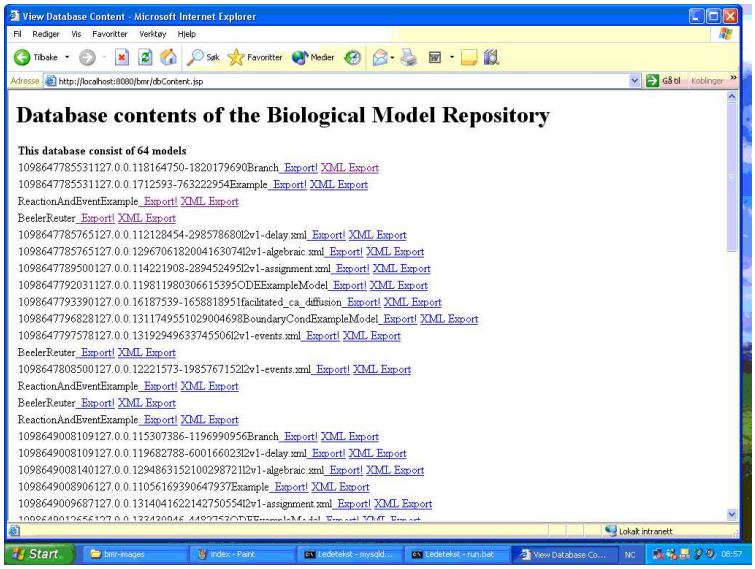

Figure 7: Content of the BMR

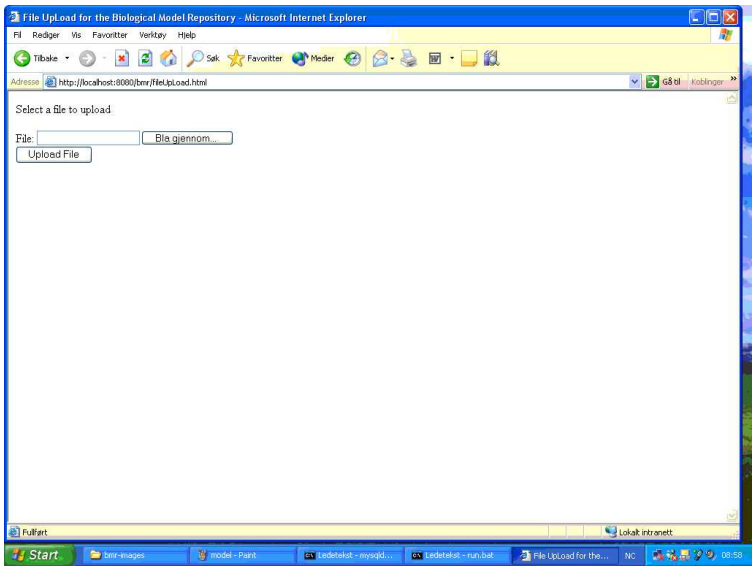

Figure 8: Submiting a file in the BMR

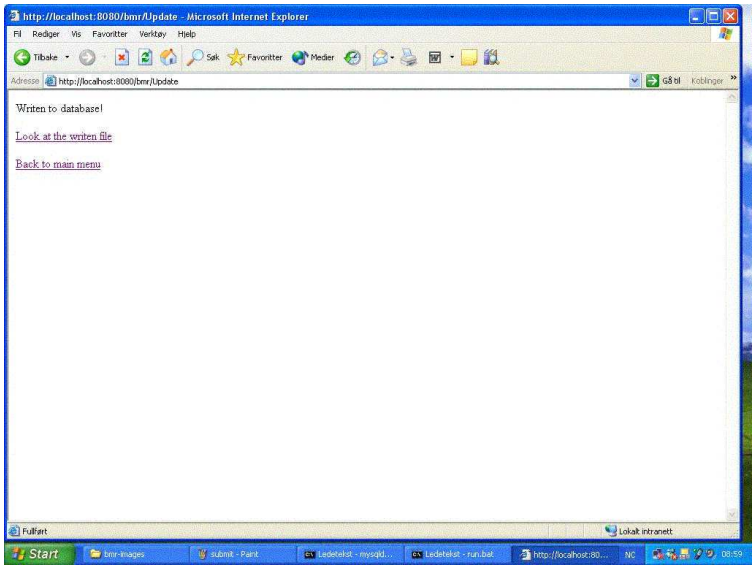

Figure 9: Result of a writen sbml file to the BMR

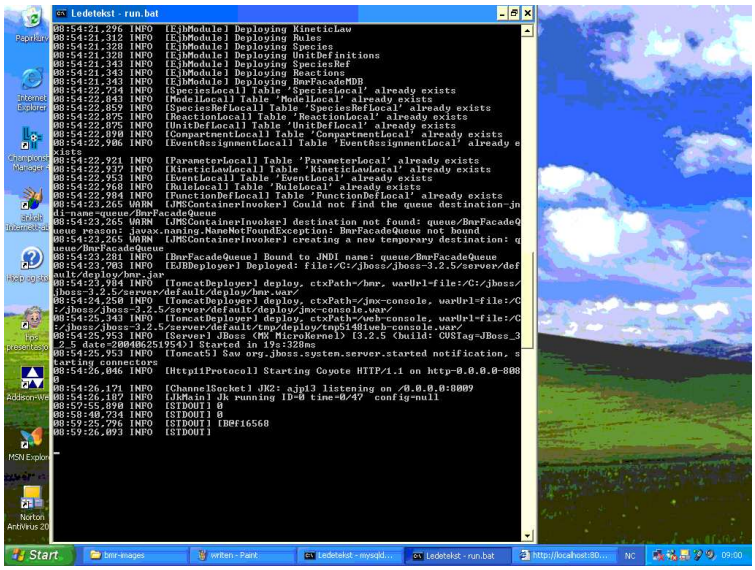

Figure 10: Console reaction on a written sbml file to the BMR

8.5 How to add new functionality

I have tried to design and implement this application so that it would be simple to add new functionality. If someone desires to add something, he pretty much only need to add a few lines into the Session Facade or Message Facade and add the component. The new component can be added in a new package inside the bmr code structured. It is also very important to remember to update any configuration files. In case someone downloads my application and desires to do something with it, I have written a readme file with the distribution that tells how to install and use this package[34]. The distribution contains a binary distribution of the BMR, two SBML files that I have constructed myself (both are discused and available in the appendix), source code and configuration files. The readme file is also available in the appendix.

9 Conclusion

The final result of this thesis was the implementation of the Biological Model Repository. This was partly achieved since only some parts where implemented, although the most important ones. The application is running and seems to be without very big faults or bugs. It can store and retrieve all the current available SBML Level 2 version 1 models, but some information may be lost during parsing. The BMR does not handle different kind of annotations which are allowed according to the SBML specification. If a third party client whishes to use my application he will probably need to rewrite it. The best thing is probably to write a new one from the scrach. I will have done that, but kept some central ideas and code fragments.

9.1 The SBML language

A part of this thesis, mostly in the beginning, was to examine the SBML language. The point of this language seems to be very good. It tries define a standard for exchange of biological and chemical models. It achieves it in a certain way, because the list of implementations and projects that use SBML keep growing from month to month. The success of SMBL can also be measured in the amount of final specifications. There currently exist three SBML specifications, and a fourth one is on the way. During the same period the CellML community had not launched a single new specification nor any additional software or libraries. These two languages are very similar, and can be compared with each other. On the basis of that it is hard to say why SBML became more popular and videly used than the CellML. The answer to this question could be the fact that SBML was a cooperation between institutes from two countries while CellML is probably only used the same place it was developed, at the University of Auckland in New Zealand.
Applications developed by the SBML community seems to be better and give the impression of being written by professionals, while this is not the case of the CellML. I am pretty sure that this is the case, since I have used both SBML and CellML applications over a longer period. They where parts of projects that did not fit into the scope of this thesis. On the other side there probably exist weaknesses with the SBML, but I have not figured out what kind of weaknesses there are. This is mainly due to the fact that I have background in computer science, and not in scientific computing, biology or chemistry.

9.2 The BMR Specification and Application

The BMR specification was developed at the beginning of my work with this thesis, and has not been change much after that. After a while I added some new features, but unfortunately I have not had the time to implement them. The currently implemented features are

The ability to store SBML models

The ability to retrieve SBML models

A server application that can be managed by an administrator

In other words have I just implemented the most basic features. My specification was developed on the foundation of my desire to make something useful for the SBML and CellML community. At the time of development of my specification neither of communities had any kind of database application to store their models. The idea was to try to cooperate with them, but that did not happen. However, I think I have achieved what I wanted by designing and implementing my application. Even if this applications is not as good as it should be, and requires rewriting, it is a good starting point for that kind of applications.

9.3 Choice of technologies

I think I did the right choice when I decided to use the full J2EE specification to implement this application. The alternatives where to use old technologies, current or new ones. It could have been a greate expericence to try a new paradigm, but I do not think that the XML databases are trustable at the current stage. The BMR could also have been implemented using old technology, but I do not think that would have been a good idea.

9.4 The BMR Source Code

I think I have achieved what I wanted by coding this application. I have learned much about Java, coding and different design patterns. It is easiest to look at my coding progression by comparing my old code with the last lines I wrote. The EJB part that handles the insertion into the database was written as the first feature for the BMR, the returning features where the last ones. If someone compares the two source codes they will notice a progression. The last written code is of much higher value, than the first one. Currently the BMR consist of more than a 100 files, which gives about 4000-5000 or more lines of code (including all configuration files written by me).
There are some things in the code that could need a refactoring or a complete rewriting. Below I present some suggestions of what should have bean done in another way.

Organization of Entity Bean I think I used to many Entity Beans to define the database, currently I have one for each part or subpart of the SBML (12 Entity Beans). I belive there is a possibility to compress them further. A good solution could be to try to store some part of the SMBL model as collections inside Entity Beans. For example if we take and look at the bmr.ejb.sbml2.EventBean (Session Bean) and the corresponding bmr.db.EventBean (Entity Bean). This component tries to parse and store 'event' from the SBML model (see chapter 4), which consist of variables, MathML elements and can have several 'eventAssignment's which in fact also consist of MathML. In the current version of BMR is this done by letting each 'event' to have one table row, and each 'eventAssignment' also have one table row. If we have 100 events and each of them have 100 eventAssignments, them we use 100 table rows in the event table (or instances of the EventBean Entity Bean), and 10 000 table rows in the eventAssignment table (or instances of the

EventAssignment Entity Bean). That is huge amount of unnecessary data. My suggestion is to compress it as much as possible, but to still have the ability to do complex queries against those data. I think one instance in the Entity Bean of each model part is enough and all additional subparts of this part should be stored in collections. The ejb-ql gives use the possibility to search in collections that are stored inside an Entity Bean. If we look at the example above I will suggest doing the following. Each event stores a collection of eventAssignments inside its Entity Bean. The eventAssignment in the database would look like a BLOB for the database, but that is ok, since we only use the bean, which by the way knows that this is a collection.

Storing of MathML I have chosen to not parse MathML elements, only to store them as objects in the corresponding Entity Beans. This is not a good solution since there should be a possibility to search up different parts of the MathML elements. A scientist should have the possibility to search for variables and functions in the database, it would help him with his research by reducing the time to look at all models. This can be achieved by putting the MathML elements into a MathML table or putting them into collections in their parents Entity Bean. The second solution is the preferred one, and something similar is described in the paragraph above.

Memory usage in parsing of SBML documents To parse SBML documents I use DOM. Since DOM loads an entire document into memory, that is not a good solution. Imagine loading an 2 MB document into memory an a overloaded server. However, my current solution uses Message Driven Beans which enables parsing and storing of SBML models at minimum memory usage while there are active users on the server. The model is just put into a queue and waits until it can be parsed. While it is parsed, then the users can have a problem accessing the server. The solution is to use some kind of SAX implementation which is fairly easy and non-complex or a combination of both SAX and DOM.

Losing of information during parsing The current implementation of the BMR loses some SBML model information during parsing. This is mostly happening to the annotation field. Since it is impossible to know what kind of annotations a model can have, I have just skipped storing them. In order to store them into the database they can be handled by the same way as I use to store the MathML elements. They can be stored as an object or as a collection inside the parent Entity Bean.

The key generator class The key generator class (bmr.utils.KeyGenerator) is a very good solution, but has one little mistake. Every time the application is accessing it a new instance is create. If we have several models that are parsed at the same time, and they are all creating a new instance at the same time, they all will get the same unique identification. I have used the synchronized key word, but that does not help in this case. Since every model is creating its own instance, they are not accessing the same

method instance concurrently. My suggestion is to make the key generator class a static class, with static synchronized methods. This solution would achieve unique identification for each model even if several models are parsed concurrently. Each model would then have to wait on the other to finish.

Some minor things There are also several other smaller things that should be done in another way. I use key class holders for every Entity Bean, that is unnecessary because I probably only need one key class holder. The user interface is not as good as it should be, and needs to be rewritten. It should take the advantage of style sheets and separate style and presentation and view and controller. Neither is done today.

9.5 What can be done in the future

I think this thesis could be used as a basis of other thesis in the future. A good idea is try to implement my suggestion and remarks that have been presented in this conclusion. It is not a very good idea to let one student do the changes, because it is too much to do. I remember using much time in the beginning to try to learn the different parts of the J2EE and different design patterns. Neither is widely used in any kind of courses at the department of computer science. Much must be learned by reading books and specifications. Although I think this knowledge is good to have, since the J2EE and component based architecture is the leading paradigm at the time. Almost every company that have an it department that does some Java programming use this kind of technology. A good start for students whom wish to continue to develop this applications is the ServerSide page on the Web [75]. If you register, you have then access to several books and articles about the component based architecture and the J2EE specification.

A Explanations

This appendix will gives short explanations of different technologies and words that are encountered in this thesis. I will focus on the technologies that do not play an important part in this thesis and therfor not explained in the man text.

Apache Software Foundation is a community of open-source software projects [25]. Currently they are one of the biggest open-source software foundations in the world, developing many useful Java and XML applications, servers and libraries.

ASCII American Standard Code for Information Interchange, an ISO 8859-1 standard for machine codes.

Backend database A database that is hidden to user of a certain software package. The user, a program or a human, only interacts with a interface and never has any contact with the given backend database.

BioSpice Community Citation from homepage *'Our goal is to create an open source framework and toolset for modeling dynamic cellular network functions and around this to develop a user community committed to using, extending and exploiting these tools to further our knowledge of biologic processes.'*

Broker is a program that passes messages between clients and servers. The most know broker specification is the Common Object Request Broker (CORBA). However it is beyond the scope of this thesis. More information about CORBA can be found on the Object Management Groups web pages [**?**].

Browser A program that reads HTML documents and presents them according to their specification.

CASE applications Computer Aided Software Engineering. Engineering with help from computers.

CERN The largest particle physics laboratory in the world.

cDNA Complementary DNA, a single-stranded DNA synthesized from a mature mRNA template.

DARPA Defense Advanced Research Project Agency, a US military research agency [7].

DNA data banks There currently exist three DNA data banks (sometimes refered to as DNA databases) DNA Data Bank of Japan (DDBJ) [39], European Molecular Biology Laboratory (EMBL) [48] and GenBank [26].

Distributed System is an application that consists of components running on different computers concurrently.

Differential Equations is an equation that describes a relationship between a set of unknowns and its derivatives. There exist ordinary differential equations and partial differencial equations.

ejb-ql Enterprise Java Bean query language. An OQL implementation.

Expression tree is an abstract representation of a tree containing an expression. E.g., a mathematical expression 'a+b' can be expressed as a tree containing both 'a' and 'b' as nodes while '+' is the parent or root element.

Facade An object that is representing a controlling system in an application. It is often an object or a package that handles flow and redirection of data inside a package or serves as controller for several packages. (link til denne pattern)

GPS Global Positioning System is a satelite positioning system desinged by the US millitary at the end of the 1980s (called NAVSTAR by US millitary). GPS is used to determine the precise postion of an object on earth or in earths orbit.

IBM International Business Machines Corporation (IBM). A world vide company that manufactures and sells computer hardware, software and services. It has been in operation since 1888, and is today the largest information technology company in the world.

Imperative programming language are programming languages like Java and C++. It is a paradignm that describes computation in terms of a program state and statements that change the program state.

jar file A packed and compressed Java binary application distribution.

JDOM A simplyfied Java based DOM accessing specification and implementation.

JAXP Java API for XML Processing. A Sun specification and implementation of a simplyfied SAX specification.

jpg A picture format which enables compression of data.

Layer is a kind of stack offering services to layers above, shielding its details and sub layers.

Lynx A text based web browser.

Mathematica is a technical computing system with high-quality mathematical typesetting and editing [57]. Currently widely used at Universities and different

Mosaic The first Web browser in the world.

Mozilla/Firefox is an open source web browser. The project started in 1998 when Netscape Communication Corporation published source code for its web browser, Netscape Communicator [61]. It is based on the Gecko engine. Currently the Mozilla community supports a additional browser (Firefox) and a mail program (Thunderbird).

NCSA A US research facility.

Netscape is web browser developed by Netscape Communication Corporation. It is based on the Gecko engine.

OQL Object Query Language - a query language for objects. Similar to SQL.

Oracle Company A company that is best known for making and implementing databases [16]

Robot A program that indexes the Web, resulting in matches in a search engine (e.g. www.google.com)

Pattern Is a set of rules that can be applied to make a certain thing. E.g. Model-View-Controller is a pattern for creating server side applications.

TeX A typing format developt by Donald Knuth in the beginning of the 1980s [47].

Tomcat server Tomcat server can be both used as a standalone server, and integrated with an other server. If integrated (for example with a jboss server) it is started from inside the parent server and uses the same context as the parent server.

Web Interface A graphical user interface that can be viewed through a Web browser

W3C A World Wide Web Consortium that standardizes the Web

W3C Recommendation A W3C specification of a protocol that is a standard and is recommended for use in applications. Since W3C only makes guidelines not actual specifications, these are called recommendations.

World Wide Web (WWW) World Wide Web is an acronym for internet. Often used for HTTP and FTP protocols.

XML4J An IBM parser that was rated as the best XML parser in ADTmag in february 1999 [40]. This parser project was later transfered to Apaches XML project.

B SBML model

This appendix shows the SBML version of the Beeleur-Reuter Mammalian Ventricular model from 1977 [32]. This SBML model is reconstructed from the original CellML model found on the CellML page [60].

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" level="2" version="1"
xmlns:math="http://www.w3.org/1998/Math/MathML" xmlns:p="http://www.w3.org/1999/xhtml">
  <model id="BeelerReuter" name="BeelerReuter">
   <notes>
      <p xmlns="http://www.w3c.org/1999/xhtml">
        Beeler-Reuter Mammalian Ventricular Model 1977
        This model is reconstructed from the original CellML model
        found on the www.cellml.org site. This is a version with
        no reactions, just mathematical rules.
      </p>
   </notes>
<!-- ========================== -->
<!-- List of FunctionDefinitions -->
<!-- ========================== -->
   <listOfFunctionDefinitions>
     <functionDefinition id="alpha_m">
       <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
         <bvar><ci>V</ci></bvar>
         <apply><divide/>
         <apply><times/>
         <apply><minus/>
            <cn>1</cn>
          </apply>
          <apply><plus/>
            <ci>V</ci>
            <cn>47</cn>
          </apply>
          </apply>
          <apply><minus/>
          <apply><exp/>
          <apply><times/>
          <apply><minus/>
            <cn>0.1</cn>
          </apply>
          <apply><plus/>
             <ci>V</ci>
             <cn>47</cn>
          </apply>
          </apply>
          <apply>
             <cn>1</cn>
          </apply>
          </apply>
          </apply>
          </apply>
        </lambda></math>
      </functionDefinition>
      <functionDefinition id="beta_m">
        <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
          <bvar><ci>V</ci></bvar>
          <apply><times/>
          <apply>
            <cn>40</cn>
          </apply>
          <apply><exp/>
          <apply><times/>
          <apply><minus/>
            <cn>0.056</cn>
          </apply>
          <apply><plus/>
            <ci>V</ci>
            <cn>72</cn>
          </apply>
          </apply>
          </apply>
          </apply>
```

```
  </lambda></math>
</functionDefinition>
<functionDefinition id="alpha_h">
  <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
    <bvar><ci>V</ci></bvar>
    <apply><times/>
    <apply>
      <cn>0.126</cn>
    </apply>
    <apply><exp/>
    <apply><times/>
    <apply><minus/>
      <cn>0.25</cn>
    </apply>
    <apply><plus/>
      <ci>V</ci>
      <cn>77</cn>
    </apply>
    </apply>
    </apply>
    </apply>
  </lambda></math>
</functionDefinition>
<functionDefinition id="beta_h">
  <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
    <bvar><ci>V</ci></bvar>
    <apply><divide/>
    <apply>
      <cn>1.7</cn>
    </apply>
    <apply><plus/>
    <apply><exp/>
    <apply><times/>
    <apply><minus/>
      <cn>0.082</cn>
    </apply>
    <apply><plus/>
      <cn>22.5</cn>
      <ci>V</ci>
    </apply>
    </apply>
    </apply>
      <cn>1.0</cn>
    </apply>
    </apply>
  </lambda></math>
</functionDefinition>
<functionDefinition id="alpha_j">
  <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
    <bvar><ci>V</ci></bvar>
    <apply><divide/>
    <apply><times/>
      <cn>0.055</cn>
    <apply><exp/>
    <apply><times/>
    <apply><minus/>
      <cn>0.25</cn>
    </apply>
    <apply><plus/>
      <ci>V</ci>
      <cn>78</cn>
    </apply>
    </apply>
    </apply>
    </apply>
    <apply><plus/>
    <apply><exp/>
    <apply><times/>
    <apply><minus/>
      <cn>0.2</cn>
    </apply>
    <apply><plus/>
      <cn>78</cn>
      <ci>V</ci>
```

```
     </apply>
     </apply>
     </apply>
       <cn>1</cn>
     </apply>
     </apply>
   </lambda></math>
 </functionDefinition>
 <functionDefinition id="beta_j">
  <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
    <bvar><ci>V</ci></bvar>
    <apply><divide/>
    <apply>
      <cn>0.3</cn>
    </apply>
    <apply><plus/>
    <apply><exp/>
    <apply><times/>
    <apply><minus/>
      <cn>0.1</cn>
    </apply>
    <apply><plus/>
      <ci>V</ci>
      <cn>32</cn>
    </apply>
    </apply>
    </apply>
      <cn>1</cn>
    </apply>
    </apply>
  </lambda></math>
</functionDefinition>
<functionDefinition id="alpha_d">
  <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
    <bvar><ci>V</ci></bvar>
    <apply><divide/>
    <apply><times/>
      <cn>0.095</cn>
   <apply> <exp/>
   <apply><minus/>
   <apply><divide/>
   <apply><plus/>
     <ci>V</ci>
   <apply><minus/>
     <cn>5</cn>
   </apply>
   </apply>
   <apply>
     <cn>100</cn>
   </apply>
   </apply>
   </apply>
   </apply>
   </apply>
   <apply><plus/>
     <cn>1</cn>
   <apply><exp/>
   <apply><minus/>
   <apply><divide/>
   <apply><plus/>
     <ci>V</ci>
   <apply><minus/>
     <cn>5</cn>
   </apply>
   </apply>
   <apply>
     <cn>13.89</cn>
   </apply>
   </apply>
   </apply>
   </apply>
   </apply>
   </apply>
 </lambda></math>
```

```
</functionDefinition>
<functionDefinition id="beta_d">
 <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
   <bvar><ci>V</ci></bvar>
   <apply><divide/>
   <apply><times/>
     <cn>0.07</cn>
   <apply><exp/>
   <apply><minus/>
   <apply><divide/>
   <apply><plus/>
     <ci>V</ci>
     <cn>44</cn>
   </apply>
   <apply>
     <cn>59</cn>
   </apply>
   </apply>
   </apply>
   </apply>
   </apply>
   <apply><plus/>
     <cn>1</cn>
   <apply><exp/>
   <apply><minus/>
   <apply><divide/>
   <apply><plus/>
     <ci>V</ci>
     <cn>44</cn>
   </apply>
   <apply>
     <cn>20</cn>
   </apply>
   </apply>
   </apply>
   </apply>
   </apply>
   </apply>
  </lambda></math>
</functionDefinition>
<functionDefinition id="alpha_f">
 <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
   <bvar><ci>V</ci></bvar>
   <apply><divide/>
   <apply><times/>
     <cn>0.012</cn>
   <apply><exp/>
   <apply><minus/>
   <apply><divide/>
   <apply><plus/>
     <ci>V</ci>
     <cn>28</cn>
   </apply>
   <apply>
     <cn>125</cn>
   </apply>
   </apply>
   </apply>
   </apply>
   </apply>
   <apply><plus/>
     <cn>1</cn>
   <apply><exp/>
   <apply><divide/>
   <apply><plus/>
     <ci>V</ci>
     <cn>28</cn>
   </apply>
   <apply>
     <cn>6.67</cn>
   </apply>
   </apply>
   </apply>
   </apply>
```

```
    </apply>
  </lambda></math>
 </functionDefinition>
 <functionDefinition id="beta_f">
  <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
    <bvar><ci>V</ci></bvar>
    <apply><divide/>
    <apply><times/>
    <cn>0.0065</cn>
    <apply><exp/>
    <apply><minus/>
    <apply><divide/>
    <apply><plus/>
      <ci>V</ci>
      <cn>30</cn>
    </apply>
    <apply>
      <cn>50</cn>
    </apply>
    </apply>
    </apply>
    </apply>
    </apply>
    <apply><plus/>
      <cn>1</cn>
    <apply><exp/>
    <apply><minus/>
    <apply><divide/>
    <apply><plus/>
      <ci>V</ci>
      <cn>30</cn>
    </apply>
    <apply>
      <cn>5</cn>
    </apply>
    </apply>
    </apply>
    </apply>
    </apply>
    </apply>
   </lambda></math>
 </functionDefinition>
 <functionDefinition id="alpha_x1">
  <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
    <bvar><ci>V</ci></bvar>
    <apply><minus/>
      <cn>5E</cn>
    <apply><times/>
      <cn>4</cn>
    <apply><divide/>
    <apply><exp/>
    <apply><divide/>
    <apply><plus/>
      <ci>V</ci>
      <cn>50</cn>
    </apply>
    <apply>
      <cn>12.1</cn>
    </apply>
    </apply>
    </apply>
    <apply><plus/>
      <cn>1</cn>
    <apply><exp/>
    <apply><divide/>
    <apply><plus/>
      <ci>V</ci>
      <cn>50</cn>
    </apply>
    <apply>
      <cn>17.5</cn>
    </apply>
    </apply>
    </apply>
```

```
  </apply>
  </apply>
  </apply>
  </apply>
 </lambda></math>
</functionDefinition>
<functionDefinition id="beta_x1">
 <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
  <bvar><ci>V</ci></bvar>
  <apply><times/>
  <cn>0.0013</cn>
  <apply><divide/>
  <apply><exp/>
  <apply><minus/>
  <apply><divide/>
  <apply><plus/>
    <ci>V</ci>
    <cn>20</cn>
  </apply>
  <apply>
    <cn>16.67</cn>
  </apply>
  </apply>
  </apply>
  </apply>
  <apply><plus/>
    <cn>1</cn>
  <apply><exp/>
  <apply><minus/>
  <apply><divide/>
  <apply><plus/>
    <ci>V</ci>
    <cn>20</cn>
  </apply>
  <apply>
    <cn>25</cn>
  </apply>
  </apply>
  </apply>
  </apply>
  </apply>
  </apply>
  </apply>
 </lambda></math>
</functionDefinition>
<functionDefinition id="i_K1">
  <math xmlns="http://www.w3.org/1998/Math/MathML"><lambda>
    <bvar><ci>V</ci></bvar>
    <apply><times/>
      <cn>0.35</cn>
      <cn>4</cn>
  <apply><plus/>
  <apply><divide/>
  <apply><minus/>
  <apply><exp/>
  <apply><times/>
    <cn>0.04</cn>
  <apply><plus/>
    <ci>V</ci>
    <cn>85</cn>
  </apply>
  </apply>
  </apply>
    <cn>1</cn>
  </apply>
  <apply><plus/>
  <apply><exp/>
  <apply><times/>
    <cn>0.08</cn>
  <apply><plus/>
    <ci>V</ci>
    <cn>53</cn>
  </apply>
  </apply>
```

```
        </apply>
        <apply><exp/>
        <apply><times/>
          <cn>0.04</cn>
        <apply><plus/>
          <ci>V</ci>
          <cn>53</cn>
        </apply>
        </apply>
        </apply>
        </apply>
        </apply>
          <cn>0.2</cn>
        </apply>
        <apply><divide/>
        <apply><plus/>
          <ci>V</ci>
          <cn>23</cn>
        </apply>
        <apply><minus/>
          <cn>1</cn>
        <apply><exp/>
        <apply><minus/>
        <apply><times/>
          <cn>0.04</cn>
        <apply><plus/>
          <ci>V</ci>
          <cn>23</cn>
        </apply>
        </apply>
        </apply>
        </apply>
        </apply>
        </apply>
        </apply>
       </lambda></math>
      </functionDefinition>
    </listOfFunctionDefinitions>
<!-- ======================== -->
<!-- List of Unit Definitions -->
<!-- ======================== -->
    <listOfUnitDefinitions>
        <unitDefinition id="millisecond" name="millisecond">
            <listOfUnits><unit kind="second" scale="-3"/></listOfUnits>
        </unitDefinition>
        <unitDefinition id="per_millisecond" name="per_millisecond">
            <listOfUnits><unit kind="second" scale="-3" exponent="-1"/></listOfUnits>
        </unitDefinition>
        <unitDefinition id="millivolt" name="millivolt">
            <listOfUnits><unit kind="volt" scale="-3"/></listOfUnits>
        </unitDefinition>
        <unitDefinition id="per_millivolt" name="per_millivolt">
            <listOfUnits><unit kind="volt" scale="-3" exponent="-1"/></listOfUnits>
        </unitDefinition>
        <unitDefinition id="per_millivolt_millisecond" name="per_millivolt_millisecond">
            <listOfUnits><unit kind="volt" scale="-3" exponent="-1"/>
            <unit kind="second" scale="-3" exponent="-1"/></listOfUnits>
        </unitDefinition>
        <unitDefinition id="milliS_per_cm2" name="milliS_per_cm2">
            <listOfUnits><unit kind="siemens" scale="-3"/>
            <unit kind="metre" scale="-2" exponent="-2"/></listOfUnits>
        </unitDefinition>
        <unitDefinition id="microF_per_cm2" name="microF_per_cm2">
            <listOfUnits><unit kind="farad" scale="-4"/>
            <unit kind="metre" scale="-2" exponent="-2"/></listOfUnits>
        </unitDefinition>
        <unitDefinition id="microA_per_cm2" name="microA_per_cm2">
            <listOfUnits><unit kind="ampere" scale="-4"/>
            <unit kind="metre" scale="-2" exponent="-2"/></listOfUnits>
        </unitDefinition>
        <unitDefinition id="concentration_units" name="concentration_units">
            <listOfUnits><unit kind="mole" scale="-3"/>
            <unit kind="litre" exponent="-1"/></listOfUnits>
        </unitDefinition>
```

```
    </listOfUnitDefinitions>
<!-- ==================== -->
<!-- List of Compartments -->
<!-- ==================== -->
    <listOfCompartments>
       <compartment id="cell" name="cell"/>
    </listOfCompartments>
<!-- =============== -->
<!-- List of Species -->
<!-- =============== -->
    <listOfSpecies>
     <species id="V" name="V" compartment="cell" initialAmount="-84.624"/>
     <species id="m" name="m" compartment="cell"  initialAmount="0.011"/>
     <species id="h" name="h" compartment="cell"  initialAmount="0.988"/>
     <species id="j" name="j" compartment="cell"  initialAmount="0.975"/>
     <species id="Cai" name="Cai" compartment="cell" />
     <species id="d" name="d" compartment="cell" initialAmount="0.003"/>
     <species id="f" name="f" compartment="cell"  initialAmount="0.994"/>
     <species id="x1" name="x1" compartment="cell" />
    </listOfSpecies>
<!-- ================= -->
<!-- List of Parameters -->
<!-- ================= -->
    <listOfParameters>
      <parameter id="C" value="1.0" units="microF_per_cm2"/>
      <parameter id="i_Na" units="microA_per_cm2"/>
      <parameter id="g_Na" value="4.0" units="milliS_per_cm2"/>
      <parameter id="E_Na" value="50.0" units="millivolt"/>
      <parameter id="g_Nac" value="0.003" units="milliS_per_cm2"/>
      <parameter id="m" value="0.011" units="dimensionless"/>
      <parameter id="alpha_m" units="per_millisecond"/>
      <parameter id="beta_m" units="per_millisecond"/>
      <parameter id="h" value="0.988" units="dimensionless"/>
      <parameter id="alpha_h" units="per_millisecond"/>
      <parameter id="beta_h" units="per_millisecond"/>
      <parameter id="j" value="0.975" units="dimensionless"/>
      <parameter id="alpha_j" units="per_millisecond"/>
      <parameter id="beta_j" units="per_millisecond"/>
      <parameter id="i_s" units="microA_per_cm2"/>
      <parameter id="g_s" value="0.09" units="milliS_per_cm2"/>
      <parameter id="E_s" units="millivolt"/>
      <parameter id="Cai" units="concentration_units"/>
      <parameter id="d" value="0.003" units="dimensionless"/>
      <parameter id="alpha_d" units="per_millisecond"/>
      <parameter id="beta_d" units="per_millisecond"/>
      <parameter id="f" value="0.994" units="dimensionless"/>
      <parameter id="alpha_f" units="per_millisecond"/>
      <parameter id="beta_f" units="per_millisecond"/>
      <parameter id="i_x1" units="microA_per_cm2"/>
      <parameter id="x1" units="dimensionless"/>
      <parameter id="alpha_x1" units="per_millisecond"/>
      <parameter id="beta_x1" units="per_millisecond"/>
      <parameter id="i_K1" units="microA_per_cm2"/>
      <!-- External parameter -->
      <parameter id="i_external" value="0.0" units="dimensionless"/>
    </listOfParameters>
<!-- ============= -->
<!-- List of Rules -->
<!-- ============= -->
    <listOfRules>
     <!-- RateRules -->
     <rateRule variable="V">
       <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply><times/>
        <apply><minus/>
        <apply><divide/>
        <apply>
          <cn>1</cn>
        </apply>
        <apply>
          <ci>C</ci>
        </apply>
        </apply>
        </apply>
```

```
    <apply><plus/>
      <ci>i_Na</ci>
      <ci>i_s</ci>
      <ci>i_x1</ci>
      <ci>i_K1</ci>
      <minus/>
      <ci>i_external</ci>
    </apply>
    </apply>
   </math>
</rateRule>
<rateRule variable="m">
   <math xmlns="http://www.w3.org/1998/Math/MathML">
    <apply><minus/>
    <apply><times/>
    <apply>
      <ci>alpha_m</ci>
    </apply>
    <apply><minus/>
    <apply>
      <cn>1</cn>
    </apply>
    <apply>
      <ci>m</ci>
    </apply>
    </apply>
    </apply>
    <apply><times/>
    <apply>
      <ci>beta_m</ci>
    </apply>
    <apply>
      <ci>m</ci>
    </apply>
    </apply>
    </apply>
  </math>
</rateRule>
<rateRule variable="h">
 <math xmlns="http://www.w3.org/1998/Math/MathML">
    <apply><minus/>
    <apply><times/>
    <apply>
      <ci>alpha_h</ci>
    </apply>
    <apply><minus/>
    <apply>
      <cn>1</cn>
    </apply>
    <apply>
      <ci>h</ci>
    </apply>
    </apply>
    </apply>
    <apply><times/>
    <apply>
      <ci>beta_h</ci>
    </apply>
    <apply>
      <ci>h</ci>
    </apply>
    </apply>
    </apply>
   </math>
  </rateRule>
  <rateRule variable="j">
   <math xmlns="http://www.w3.org/1998/Math/MathML">
    <apply><minus/>
    <apply><times/>
    <apply>
      <ci>alpha_j</ci>
    </apply>
    <apply><minus/>
    <apply>
```

```
    <cn>1</cn>
   </apply>
   <apply>
    <ci>j</ci>
   </apply>
   </apply>
   </apply>
   <apply><times/>
   <apply>
    <ci>beta_j</ci>
   </apply>
   <apply>
    <ci>j</ci>
   </apply>
   </apply>
   </apply>
  </math>
 </rateRule>
 <rateRule variable="Cai">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
   <apply><times/>
   <apply><minus/>
   <apply>
    <ci>0.001</ci>
   </apply>
   </apply>
   <apply><plus/>
   <apply>
    <cn>i_s</cn>
    <ci>0.007</ci>
   </apply>
   </apply>
   <apply><minus/>
   <apply>
    <ci>0.001</ci>
   </apply>
   <apply>
    <cn>Cai</cn>
   </apply>
   </apply>
   </apply>
  </math>
 </rateRule>
 <rateRule variable="d">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
   <apply><minus/>
   <apply><times/>
   <apply>
    <ci>alpha_d</ci>
   </apply>
   <apply><minus/>
   <apply>
    <cn>1</cn>
   </apply>
   <apply>
    <ci>d</ci>
   </apply>
   </apply>
   </apply>
   <apply><times/>
   <apply>
    <ci>beta_d</ci>
   </apply>
   <apply>
    <ci>d</ci>
   </apply>
   </apply>
   </apply>
  </math>
 </rateRule>
 <rateRule variable="f">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
   <apply><minus/>
   <apply><times/>
```

```
  <apply>
    <ci>alpha_f</ci>
  </apply>
  <apply><minus/>
  <apply>
    <cn>1</cn>
  </apply>
  <apply>
    <ci>f</ci>
  </apply>
  </apply>
  </apply>
  <apply><times/>
  <apply>
    <ci>beta_f</ci>
  </apply>
  <apply>
    <ci>f</ci>
  </apply>
  </apply>
  </apply>
 </math>
</rateRule>
<rateRule variable="x1">
 <math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply><minus/>
  <apply><times/>
  <apply>
    <ci>alpha_x1</ci>
  </apply>
  <apply><minus/>
  <apply>
    <cn>1</cn>
  </apply>
  <apply>
    <ci>x1</ci>
  </apply>
  </apply>
  </apply>
  <apply><times/>
  <apply>
    <ci>beta_x1</ci>
  </apply>
  <apply>
    <ci>x1</ci>
  </apply>
  </apply>
  </apply>
 </math>
</rateRule>
<!-- Assignment Rules -->
<assignmentRule variable="i_Na">
 <math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply><times/>
  <apply><plus/>
  <apply><times/>
    <ci>g_Na</ci>
  <apply><power/>
    <ci>m</ci>
    <cn>3</cn>
  </apply>
    <ci>h</ci>
    <ci>j</ci>
  </apply>
  <apply><plus/>
    <ci>g_Nac</ci>
  </apply>
  </apply>
  <apply><minus/>
  <apply>
    <ci>V</ci>
  </apply>
  <apply>
    <ci>E_Na</ci>
```

```
        </apply>
        </apply>
        </apply>
       </math>
      </assignmentRule>
      <assignmentRule variable="i_x1">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply><times/>
          <ci>x1</ci>
          <cn>0.8</cn>
        <apply><divide/>
        <apply><minus/>
        <apply><exp/>
        <apply><times/>
          <cn>0.04</cn>
        <apply><plus/>
          <ci>V</ci>
          <cn>77</cn>
        </apply>
        </apply>
        </apply>
          <cn>1</cn>
        </apply>
        <apply><exp/>
        <apply><times/>
          <cn>0.04</cn>
        <apply><plus/>
          <ci>V</ci>
          <cn>35</cn>
        </apply>
        </apply>
        </apply>
        </apply>
        </apply>
       </math>
      </assignmentRule>
      <assignmentRule variable="E_s">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
         <apply><minus/>
           <cn>82.3</cn>
         <apply><times/>
           <cn>13.0287</cn>
        <apply><ln/>
          <ci>Cai</ci>
        </apply>
        </apply>
        </apply>
      </math>
      </assignmentRule>
      <assignmentRule variable="i_s">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply><times/>
          <ci>g_s</ci>
          <ci>d</ci>
          <ci>f</ci>
        <apply><minus/>
          <ci>V</ci>
          <ci>E_s</ci>
        </apply>
        </apply>
      </math>
      </assignmentRule>
     <!-- Algebraic Rules -->
     </listOfRules>
   </model>
</sbml>
```

C SBML model 2

This appendix shows a selfconstructed SBML model. All reactions are ficital. This model is only used to explain SBML parts not covered by the SBML model over Beeler-Reuter model.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" level="2" version="1">
  <model id="ReactionAndEventExample" name="ReactionAndEventExample">
    <notes>
      <p xmlns="http://www.w3.org/1999/xhtml">
        A small example involving reactions and events
      </p>
    </notes>
    <listOfCompartments>
        <compartment id="one" size="1"/>
    </listOfCompartments>
    <listOfSpecies>
        <species id="X" compartment="one"/>
        <species id="S" compartment="one"/>
    </listOfSpecies>
    <listOfParameters>
        <parameter id="k" value="0"/>
        <parameter id="j" value="0"/>
    </listOfParameters>
    <listOfReactions>
        <reaction id="reaction1" reversible="false">
          <listOfReactants>
            <speciesReference species="X"/>
          </listOfReactants>
          <listOfProducts>
            <speciesReference species="S"/>
          </listOfProducts>
          <kineticLaw>
            <math xmlns="http://www.w3.org/1998/Math/MathML">
              <apply>
                <times/>
                <ci>k</ci>
                <ci>j</ci>
              </apply>
            </math>
          </kineticLaw>
        </reaction>
    </listOfReactions>
    <listOfEvents>
      <event>
        <trigger>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <leq/>
              <ci>X</ci>
              <ci>k</ci>
            </apply>
          </math>
        </trigger>
        <listOfEventAssignments>
          <eventAssignment variable="S">
            <math xmlns="http://www.w3.org/1998/Math/MathML">
              <cn>0</cn>
            </math>
          </eventAssignment>
        </listOfEventAssignments>
      </event>
    </listOfEvents>
  </model>
</sbml>
```

D BMR source

This appendix shows some examples of central BMR files that are described in this thesis.

D.1 Utility classes

This sub appendix shows an example of utility classes described in the thesis and used in the bmr application.

bmr.utils.KeyGenerator This is the key generator class described in chapter..

```
package bmr.utils;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.security.SecureRandom;
/**
*
*  Universally unique identifier (UUID)
*  Used modified example from
*  'EJB Design Patterns' by
*  Floyd Marinescu. His solution is
*  based on the implementation by Steve
*  Woodcock. The idea behind this solution
*  can be found at
*  http://casl.csa.iisc.ernet.in/Standards/internet-draft/draft-
*  leach-uuids-guide-01.txt
*
**/
public class KeyGenerator {

public KeyGenerator(){}
/**
 * A synchronized method that Generates and returns a model UUID
 * @return a UUID
 */
public synchronized String getUUID(){
String uuID = null;
try{
InetAddress inet = InetAddress.getLocalHost();
String ipAdr = inet.getHostAddress();
int idHash = System.identityHashCode(this);
String thisHash = String.valueOf(idHash);
SecureRandom seeder = new SecureRandom();
int random = seeder.nextInt();
String secureRandom = String.valueOf(random);
long timeNow = System.currentTimeMillis();
String currentTime = String.valueOf(timeNow);
uuID = currentTime + ipAdr + idHash + secureRandom;
}catch (UnknownHostException unKnown){
System.err.println(unKnown);
}
return uuID;
}
}
```

bmr.utils.dbkeys.EventPK This is a serializable class that holds on a Entity Bean key.

```
package bmr.utils.dbkeys;
import java.io.Serializable;
/**
 * A Serializable object containing a key for EventBean
 */
public class EventPK implements Serializable {
public String eventID;
```

```
public EventPK(String eventID){
  this.eventID = eventID;
}
public EventPK(){}

public String toString(){
return eventID.toString();
}
public int hashCode(){
return eventID.hashCode();
}
public boolean equals(Object o){
return ((EventPK)o).eventID.equals(eventID);
 }
}
```

bmr.utils.sbml2.SbmlMath This is an SBML2 specific class that extracts MathML from an SBML2 node.

```
package bmr.utils.sbml2;
import org.w3c.dom.Node;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.Source;
import javax.xml.transform.Result;
import javax.xml.transform.stream.StreamResult;
import java.io.StringWriter;
/**
 * A helper class that is used to get a mathml object from
 * an sbml file
 */
public class SbmlMath {
/**
 *
 * @param node a parser node
 * @return object containing a mathml string
 */
public synchronized Object getMathML(Node node){
StringWriter stringWriter;
String math = null;
for(int i=0;i<node.getChildNodes().getLength();i++){
  if(Node.ELEMENT_NODE == 1
  && node.getChildNodes().item(i).getNodeValue() == null){
try{
  TransformerFactory xformFactory = TransformerFactory.newInstance();
  Transformer transformer = xformFactory.newTransformer();
  Source input = new DOMSource(node.getChildNodes().item(i));
  stringWriter = new StringWriter();
  Result output = new StreamResult(stringWriter);
  transformer.clearParameters();
  transformer.transform(input,output);
  math = stringWriter.getBuffer().toString();
} catch (Exception e){
  System.out.println(e);
}
  }//end if
}//end for
return (Object)math.substring(38,math.length());
}//end get MathML
}
```

bmr.utils.sbml2.SbmlParameters This is an SBML2 specific class that represents an SBML2 model.

```
package bmr.utils.sbml2;
/**
 * A static class containing specific Sbml parameters
 */
public class SbmlParameters {
public static final String XML_FILE_INFO =
```

```
"<?xml version=\"1.0\" encoding=\"UTF-8\"?>";
public static final String SBML_FILE_BEGIN =
"<sbml xmlns=\"http://www.sbml.org/sbml/level2\" level=\"2\"
version=\"1\"  xmlns:math=\"http://www.w3.org/1998/Math/MathML\"
xmlns:p=\"http://www.w3.org/1999/xhtml\">";
public static final String SBML_FILE_END = "</model></sbml>";
public static final String LIST_OF_FUNCTION_DEF_BEGIN = "<listOfFunctionDefinitions> \n";
public static final String LIST_OF_FUNCTION_DEF_END = "</listOfFunctionDefinitions> \n";
public static final String FUNCTION_DEF_BEGIN = "<functionDefinition id=\"";
public static final String FUNCTION_DEF_END = "</functionDefinition>";
public static final String LIST_OF_COMPARTMENTS_DEF_BEGIN = "<listOfCompartments> \n";
public static final String LIST_OF_COMPARTMENTS_DEF_END = "</listOfCompartments> \n";
public static final String COMPARTMENTS_DEF_BEGIN = "<compartment id=\"";
public static final String COMPARTMENTS_DEF_END = "/>";
public static final String LIST_OF_SPECIES_DEF_BEGIN = "<listOfSpecies> \n";
public static final String LIST_OF_SPECIES_DEF_END = "</listOfSpecies> \n";
public static final String SPECIES_DEF_BEGIN = "<species id=\"";
public static final String SPECIES_DEF_END = "/>";
public static final String LIST_OF_PARAMETERS_DEF_BEGIN = "<listOfParameters> \n";
public static final String LIST_OF_PARAMETERS_DEF_END = "</listOfParameters> \n";
public static final String PARAMETERS_DEF_BEGIN = "<parameter id=\"";
public static final String PARAMETERS_DEF_END = "/>";
public static final String LIST_OF_EVENTASSIGNMENT_DEF_BEGIN = "<listOfEventAssignments> \n";
public static final String LIST_OF_EVENTASSIGNMENT_DEF_END = "</listOfEventAssignments> \n";
public static final String EVENTASSIGNMENT_DEF_BEGIN = "<eventAssignment variable=\"";
public static final String EVENTASSIGNMENT_DEF_END = "\">";
public static final String LIST_OF_EVENTS_DEF_BEGIN = "<listOfEvents> \n";
public static final String LIST_OF_EVENTS_DEF_END = "</listOfEvents> \n";
public static final String EVENT_DEF_BEGIN = "<event";
public static final String EVENT_DEF_END = "</event>";
public static final String LIST_OF_UNITDEFINITIONS_DEF_BEGIN = "<listOfUnitDefinitions> \n";
public static final String LIST_OF_UNITDEFINITIONS_DEF_END = "</listOfUnitDefinitions> \n";
public static final String UNITDEFINITION_DEF_BEGIN = "<unitDefinition id=\"";
public static final String UNITDEFINITION_DEF_END = "</unitDefinition> \n";
public static final String LIST_OF_UNITS_DEF_BEGIN = "<listOfUnits> \n";
public static final String LIST_OF_UNITS_DEF_END = "</listOfUnits> \n";
public static final String LIST_OF_RULES_DEF_BEGIN = "<listOfRules> \n";
public static final String LIST_OF_RULES_DEF_END = "</listOfRules> \n";
public static final String RULE_ALGEBRAIC_DEF_BEGIN = "<algebraicRule ";
public static final String RULE_ALGEBRAIC_DEF_END = "</algebraicRule>";
public static final String RULE_ASSIGNMENT_DEF_BEGIN = "<assignmentRule ";
public static final String RULE_ASSIGNMENT_DEF_END = "</assignmentRule>";
public static final String RULE_RATE_DEF_BEGIN = "<rateRule ";
public static final String RULE_RATE_DEF_END = "</rateRule>";
public static final String KINETICLAW_DEF_BEGIN = "<kineticLaw";
public static final String KINETICLAW_DEF_END = "</kineticLaw>";
public static final String LIST_OF_REACTANTS_DEF_BEGIN = "<listOfReactants> \n";
public static final String LIST_OF_REACTANTS_DEF_END = "</listOfReactants> \n";
public static final String SPECIESREF_DEF_BEGIN = "<speciesReference species=\"";
public static final String LIST_OF_PRODUCTS_DEF_BEGIN = "<listOfProducts> \n";
public static final String LIST_OF_PRODUCTS_DEF_END = "</listOfProducts> \n";
public static final String LIST_OF_MODIFIERS_DEF_BEGIN = "<listOfModifiers> \n";
public static final String LIST_OF_MODIFIERS_DEF_END = "</listOfModifiers> \n";
public static final String MODIFIER_SPECIESREF_DEF_BEGIN = "<modifierSpeciesReference species=\"";
public static final String STOICHIOMETRY_MATH_DEF_BEGIN = "<stoichiometryMath> \n";
public static final String STOICHIOMETRY_MATH_DEF_END = "</stoichiometryMath> \n";
public static final String LIST_OF_REACTIONS_DEF_BEGIN = "<listOfReactions> \n";
public static final String LIST_OF_REACTIONS_DEF_END = "</listOfReactions> \n";
public static final String RECTION_DEF_BEGIN = "<reaction id=\"";
}
```

D.2 Database - Entity Beans

This appendix shows an example of an entity bean used in the BMR. Since this bean consist of three parts, all parts are show.

EventLocalHome interface This is the bmr.db.EventLocalHome.java interface

```
package bmr.db;
import javax.ejb.EJBLocalHome;
```

```
import javax.ejb.CreateException;
import javax.ejb.FinderException;
import java.util.Collection;
import bmr.utils.dbkeys.EventPK;

public interface EventLocalHome extends EJBLocalHome {
EventLocal create(String eventID,String eventName,Object trigger,
Object delay,String timeUnits,String modelID) throws CreateException;
public EventLocal findByPrimaryKey(EventPK key) throws FinderException;
public Collection findEvents(String modelID) throws FinderException;
}
```

EventLocal interface This is the bmr.db.EventLocal.java interface.

```
package bmr.db;
import javax.ejb.EJBLocalObject;

public interface EventLocal extends EJBLocalObject {
public abstract String getEventID();
public abstract String getEventName();
public abstract void setEventName(String eventName);
public abstract Object getTrigger();
public abstract void setTrigger(Object trigger);
public abstract Object getDelay();
public abstract void setDelay(Object delay);
public abstract String getTimeUnits();
public abstract void setTimeUnits(String timeUnits);
public abstract String getModelID();
public abstract void setModelID(String modelID);
}
```

EventBean This is the bmr.db.EventBean.java class.

```
package bmr.db;
import java.rmi.RemoteException;
import javax.ejb.EJBException;
import javax.ejb.EntityBean;
import javax.ejb.EntityContext;
import javax.ejb.RemoveException;
import javax.ejb.CreateException;
import bmr.utils.dbkeys.EventPK;
/**
 * A bean that takes care of inserting events into a database
 */
public abstract class EventBean implements EntityBean {
private EntityContext context;
public EventBean() {
}
public abstract String getEventID();
public abstract void setEventID(String eventID);
public abstract String getEventName();
public abstract void setEventName(String eventName);
public abstract Object getTrigger();
public abstract void setTrigger(Object trigger);
public abstract Object getDelay();
public abstract void setDelay(Object delay);
public abstract String getTimeUnits();
public abstract void setTimeUnits(String timeUnits);
public abstract String getModelID();
public abstract void setModelID(String modelID);

public void ejbPostCreate(String eventID,String eventName,Object
trigger,Object delay,String timeUnits,String modelID){}

public EventPK ejbCreate(String eventID,String eventName,
Object trigger,Object delay,String timeUnits,String modelID)
throws CreateException{
  setEventID(eventID);
  setEventName(eventName);
  setTrigger(trigger);
  setDelay(delay);
  setTimeUnits(timeUnits);
```

```
  setModelID(modelID);

  return new EventPK(eventID);
}
public void ejbActivate() throws EJBException, RemoteException {
}
public void ejbLoad() throws EJBException, RemoteException {
}
public void ejbPassivate() throws EJBException, RemoteException {
}
public void ejbRemove()
throws RemoveException, EJBException, RemoteException {
}
public void ejbStore() throws EJBException, RemoteException {

}
public void setEntityContext(EntityContext context)
throws EJBException, RemoteException {
this.context = context;
}
public void unsetEntityContext() throws EJBException, RemoteException {
this.context = null;
}
}
```

D.3 Facades

This appendix shows the Session Facade bean that is used in the BMR. Since this bean consist of three parts, all parts are show.

Session Facade - BmrFacade.java

```
package bmr.ejb.facade;
import javax.ejb.EJBObject;
import javax.ejb.EJBException;
import java.rmi.RemoteException;
import java.util.Collection;

public interface BmrFacade extends EJBObject{
public String getModel(String modelID) throws RemoteException,EJBException;
public Collection browseDatabase() throws RemoteException,EJBException;
public String getMathAsC(String modelID) throws RemoteException,EJBException;
public String getMathAsXML(String modelID) throws RemoteException,EJBException;
}
```

Session Facade - BmrFacadeHome.java

```
package bmr.ejb.facade;
import javax.ejb.EJBHome;

public interface BmrFacadeHome extends EJBHome {
public BmrFacade create() throws java.rmi.RemoteException, javax.ejb.CreateException;
}
```

Session Facade - BmrFacadeBean.java

```
package bmr.ejb.facade;

import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import javax.ejb.FinderException;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import java.util.Collection;
```

```
import bmr.db.ModelLocalHome;
import bmr.ejb.service.*;
/**
 * Main Bean controling the bmr application
 */
public class BmrFacadeBean implements SessionBean {
private SessionContext context;
private ModelLocalHome modelLocalHome;
/**
 * A method that returns an sbml model as a string
 * @param modelID an id for the current model
 * @return an sbml file represented as a string
 * @throws RemoteException
 * @throws EJBException
 */
public String getModel(String modelID) throws RemoteException,EJBException{
String model = null;
try{
SbmlFileLocalHome sbmlfileLocalHome = lookUpSbmlFileLocalHome();
SbmlFile sbmlFile = sbmlfileLocalHome.create();
model = sbmlFile.getSbmlModel(modelID);
}catch(CreateException c){
c.printStackTrace();
}
return model;
}
/**
 * A method that returns a collections containing all models
 * in the database
 * @return a collection of sbml models
 * @throws RemoteException
 * @throws EJBException
 */
public Collection browseDatabase() throws RemoteException,EJBException{
Collection co = null;
try{
modelLocalHome = lookUpModelLocalHome();
co = modelLocalHome.findAllModels();
}catch (FinderException f){
f.printStackTrace();
}

return co;
}
/**
 * A method that returns math in a smbl model as a
 * C/C++ string
 * @param modelID an id for the current model
 * @return math in the sbml model as a C/C++ string
 * @throws RemoteException
 * @throws EJBException
 */
public String getMathAsC(String modelID) throws RemoteException,EJBException{
return modelID;
}
/**
 * A method that returns math in a sbml model as
 * a XML string
 * @param modelID an id for the current model
 * @return math in the sbml model as a XML string
 * @throws RemoteException
 * @throws EJBException
 */
public String getMathAsXML(String modelID) throws RemoteException,EJBException{
return modelID;
}

public void ejbActivate() throws EJBException, RemoteException {}
public void ejbPassivate() throws EJBException, RemoteException {}
public void ejbRemove() throws EJBException, RemoteException {}
public void ejbCreate() throws EJBException,RemoteException{}

public void setSessionContext(SessionContext context)
throws EJBException, RemoteException {
```

```
this.context = context;
}

private ModelLocalHome lookUpModelLocalHome(){
Object objref = null;
try{
Context initial = new InitialContext();
objref = initial.lookup("java:comp/env/ejb/bmr/db/ModelLocalHome");
}catch (NamingException n){
throw new EJBException(n.getMessage());
}
return (ModelLocalHome)objref;
}

private SbmlFileLocalHome lookUpSbmlFileLocalHome(){
Object objref = null;
try{
Context initial = new InitialContext();
objref = initial.lookup("local/SbmlFile");
//objref = initial.lookup("java:comp/env/ejb/bmr/ejb/service/SbmlFileLocalHome");
}catch (NamingException n){
throw new EJBException(n.getMessage());
}

return (SbmlFileLocalHome)objref;
}
}
```

Message Facade - BmrFacadeMDB.java

```
package bmr.ejb.facade;
import java.rmi.RemoteException;
import javax.ejb.EJBException;
import javax.ejb.MessageDrivenBean;
import javax.ejb.MessageDrivenContext;
import javax.ejb.CreateException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.StreamMessage;
import javax.jms.JMSException;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import bmr.ejb.service.InsertIntoDbLocalHome;
import bmr.ejb.service.InsertIntoDb;
/**
 * A bean that recives and sends an sbml file
 */
public class BmrFacadeMDB implements MessageDrivenBean, MessageListener {
private MessageDrivenContext context;
private InsertIntoDbLocalHome insertIntoDbLocalHome;
public void ejbRemove() throws EJBException {
}
public void ejbCreate() throws EJBException{
}
public void setMessageDrivenContext(MessageDrivenContext context)
throws EJBException {
this.context = context;
}
/**
 * A method that processes a message send to this MDB
 */
public void onMessage(Message msg) {
  if (msg instanceof StreamMessage){
try{
StreamMessage bm = (StreamMessage)msg;
Object o = bm.readObject();
System.out.println(o);
insertIntoDbLocalHome = lookUpInsertIntoDb();
InsertIntoDb insertIntoDb = insertIntoDbLocalHome.create();
insertIntoDb.upDateDb(o);
}catch (JMSException jms){
jms.printStackTrace();
```

```
}catch (CreateException c){
c.printStackTrace();
}catch (RemoteException r){
  r.printStackTrace();
}
}
}
private InsertIntoDbLocalHome lookUpInsertIntoDb(){
Object objref = null;
try{
Context initial = new InitialContext();
objref = initial.lookup("java:comp/env/ejb/bmr/ejb/service/InsertIntoDbLocalHome");
}catch (NamingException n){
throw new EJBException(n.getMessage());
}
return (InsertIntoDbLocalHome)objref;
}
}
```

D.4 Update Database

This appendix shows the bmr.ejb.service.InsertIntoDbBean.

```
package bmr.ejb.service;
import java.rmi.RemoteException;
import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import javax.ejb.CreateException;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import java.io.*;
import org.w3c.dom.NodeList;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.FactoryConfigurationError;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;
import org.xml.sax.InputSource;
import bmr.db.ModelLocal;
import bmr.db.ModelLocalHome;
import bmr.utils.KeyGenerator;
import bmr.ejb.sbml2.interfaces.FunctionDefinitions;
import bmr.ejb.sbml2.interfaces.FunctionDefinitionsHome;
import bmr.ejb.sbml2.interfaces.UnitDefinitions;
import bmr.ejb.sbml2.interfaces.UnitDefinitionsHome;
import bmr.ejb.sbml2.interfaces.Compartments;
import bmr.ejb.sbml2.interfaces.CompartmentsHome;
import bmr.ejb.sbml2.interfaces.Species;
import bmr.ejb.sbml2.interfaces.SpeciesHome;
import bmr.ejb.sbml2.interfaces.Parameters;
import bmr.ejb.sbml2.interfaces.ParametersHome;
import bmr.ejb.sbml2.interfaces.Rules;
import bmr.ejb.sbml2.interfaces.RulesHome;
import bmr.ejb.sbml2.interfaces.Reactions;
import bmr.ejb.sbml2.interfaces.ReactionsHome;
import bmr.ejb.sbml2.interfaces.Events;
import bmr.ejb.sbml2.interfaces.EventsHome;
/**
 * Bean that takes care of database insertion
 */
public class InsertIntoDbBean implements SessionBean {
private SessionContext context;
private ModelLocalHome modelLocalHome;
private FunctionDefinitionsHome functionDefinitionsHome;
private UnitDefinitionsHome unitDefinitionsHome;
private CompartmentsHome compartmentsHome;
private SpeciesHome speciesHome;
private ParametersHome parametersHome;
private RulesHome rulesHome;
private ReactionsHome reactionsHome;
private EventsHome eventsHome;
private File f;
/**
 * A method that updates the database
 * @param o an object containing a sbml file
 * @throws EJBException
 * @throws RemoteException
 */
public void upDateDb(Object o)throws EJBException,RemoteException{
byte []bytes = (byte[])o;
ByteArrayInputStream byteInStream = new ByteArrayInputStream(bytes);
Document doc = getDocument(byteInStream);
parseModel(doc);
System.gc();
}
private void parseModel(Document doc){
String modelName = null;
String modelID = null;
```

```
KeyGenerator keyGenerator;
for(int i=0;i<doc.getChildNodes().getLength();i++){
if(doc.getChildNodes().item(i).getNodeName().equalsIgnoreCase("sbml") == true){
for(int j=0;j<doc.getChildNodes().item(i).getChildNodes().getLength();j++){
if(doc.getChildNodes().item(i).getChildNodes().
item(j).getNodeName().equalsIgnoreCase("model") == true){
 NamedNodeMap modelAttributes = doc.getChildNodes().item(i).
 getChildNodes().item(j).getAttributes();
 keyGenerator = new KeyGenerator();
 if(modelAttributes.getNamedItem("id") != null)
    modelID = modelAttributes.getNamedItem("id").getNodeValue();
 else
modelID = f.getName();
if(modelAttributes.getNamedItem("name") != null)
    modelName = modelAttributes.getNamedItem("name").getNodeValue();
String Id = keyGenerator.getUUID();
insert(Id,modelID,modelName);
parseCompartments(doc.getChildNodes().item(i).getChildNodes().item(j).getChildNodes(),Id+modelID);
}//end model
}//end for int=j
}//end if == sbml
}//end for
}//end parseModel
private void parseCompartments(NodeList compartments, String id){
String ruleType;
System.out.println(compartments.item(2).getNodeValue());
for(int i=0;i<compartments.getLength();i++){
if(compartments.item(i).getNodeName().
   equalsIgnoreCase("listOfFunctionDefinitions") == true){
    for(int j=0;j<compartments.item(i).getChildNodes().getLength();j++){
    if(compartments.item(i).getChildNodes().item(j).getNodeName()
    .equalsIgnoreCase("functionDefinition") == true){
    try{
    if(functionDefinitionsHome == null)
         functionDefinitionsHome =(FunctionDefinitionsHome)lookUpGeneral
         ("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/FunctionDefinitionsHome");
    FunctionDefinitions fd = functionDefinitionsHome.create();
fd.insertIntoDb(compartments.item(i).getChildNodes().item(j),id);
    }catch (NamingException n){
    n.printStackTrace();
    }catch (CreateException c){
    c.printStackTrace();
    }
    }//end if
    }//end for
}else if(compartments.item(i).getNodeName().
   equalsIgnoreCase("listOfUnitDefinitions") == true){
    for(int j=0;j<compartments.item(i).getChildNodes().getLength();j++){
    if(compartments.item(i).getChildNodes().item(j).getNodeName()
.equalsIgnoreCase("unitDefinition") == true){
try{
if(unitDefinitionsHome == null)
unitDefinitionsHome =(UnitDefinitionsHome) lookUpGeneral
("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/UnitDefinitionsHome");
UnitDefinitions unit = unitDefinitionsHome.create();
unit.insertIntoDb(compartments.item(i).getChildNodes().item(j),id);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException c){
c.printStackTrace();
}
}//end if
    }//end for
}else if(compartments.item(i).getNodeName().
   equalsIgnoreCase("listOfCompartments") == true){
    for(int j=0;j<compartments.item(i).getChildNodes().getLength();j++){
    if(compartments.item(i).getChildNodes().item(j).getNodeName()
.equalsIgnoreCase("compartment") == true){

try{
if(compartmentsHome == null)
compartmentsHome =(CompartmentsHome) lookUpGeneral
("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/CompartmentsHome");
Compartments comp = compartmentsHome.create();
```

```
comp.insertIntoDb(compartments.item(i).getChildNodes().item(j),id);
    }catch (NamingException n){
n.printStackTrace();
}catch (CreateException c){
c.printStackTrace();
}//end try-catch
}//end if
    }//end for
}else if(compartments.item(i).getNodeName().
   equalsIgnoreCase("listOfSpecies") == true){
    for(int j=0;j<compartments.item(i).getChildNodes().getLength();j++){
if(compartments.item(i).getChildNodes().item(j).getNodeName()
  .equalsIgnoreCase("species") == true){
try{
if(speciesHome == null)
speciesHome =(SpeciesHome) lookUpGeneral
("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/SpeciesHome");
Species species = speciesHome.create();
species.insertIntoDb(compartments.item(i).getChildNodes().item(j),id);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException c){
c.printStackTrace();
}//end try-catch
}//end if
    }//End for
   /** Noe feil i denne delen**/
}else if(compartments.item(i).getNodeName().
   equalsIgnoreCase("listOfParameters") == true){
for(int j=0;j<compartments.item(i).getChildNodes().getLength();j++){
if(compartments.item(i).getChildNodes().item(j).getNodeName()
  .equalsIgnoreCase("parameter") == true){
try{
if(parametersHome == null)
parametersHome =(ParametersHome) lookUpGeneral
("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/ParametersHome");
Parameters parameters = parametersHome.create();
parameters.insertIntoDb(compartments.item(i).getChildNodes().item(j),id);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException c){
c.printStackTrace();
}//end try-catch
  }//end if
}//end for
}else if(compartments.item(i).getNodeName().
   equalsIgnoreCase("listOfRules") == true){
    ruleType = null;
    Node node = null;
    Rules rules = null;
    for(int j=0;j<compartments.item(i).getChildNodes().getLength();j++){
    if(compartments.item(i).getChildNodes().item(j).getNodeName().
    equalsIgnoreCase("AlgebraicRule") == true) {
    ruleType = compartments.item(i).getChildNodes().item(j).getNodeName();
    node = compartments.item(i).getChildNodes().item(j);
    }else if(compartments.item(i).getChildNodes().item(j).getNodeName().
equalsIgnoreCase("AssignmentRule") == true) {
ruleType = compartments.item(i).getChildNodes().item(j).getNodeName();
node = compartments.item(i).getChildNodes().item(j);
} else if(compartments.item(i).getChildNodes().item(j).getNodeName().
equalsIgnoreCase("RateRule") == true) {
ruleType = compartments.item(i).getChildNodes().item(j).getNodeName();
node = compartments.item(i).getChildNodes().item(j);
}
    //}//end for /** liten bug**/

    if(ruleType != null){
try{
if(rulesHome == null)
rulesHome =(RulesHome) lookUpGeneral
("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/RulesHome");
rules = rulesHome.create();

rules.insertIntoDb(node,id,ruleType);
```

```
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException c){
c.printStackTrace();
}
    }//end if
    }//end for
}else if(compartments.item(i).getNodeName().
   equalsIgnoreCase("listOfReactions") == true){
for(int j=0;j<compartments.item(i).getChildNodes().getLength();j++){
if(compartments.item(i).getChildNodes().item(j).getNodeName()
.equalsIgnoreCase("reaction") == true){
try{
if(reactionsHome == null)
reactionsHome =(ReactionsHome) lookUpGeneral
("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/ReactionsHome");
Reactions reaction = reactionsHome.create();
reaction.insertIntoDb(compartments.item(i).getChildNodes().item(j),id);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException c){
c.printStackTrace();
}
}//end if
}//end for

}else if(compartments.item(i).getNodeName().
   equalsIgnoreCase("listOfEvents") == true){
for(int j=0;j<compartments.item(i).getChildNodes().getLength();j++){
if(compartments.item(i).getChildNodes().item(j).getNodeName()
.equalsIgnoreCase("event") == true){
try{
if(eventsHome == null)
eventsHome =(EventsHome) lookUpGeneral
("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/EventsHome");
Events event = eventsHome.create();
event.insertIntoDb(compartments.item(i).getChildNodes().item(j),id);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException c){
c.printStackTrace();
}
}//end if
}//end for

}
}//end for
}//end parseCompartments

private void insert(String id,String modelID,String modelName){
try{
  ModelLocal modelLocal = modelLocalHome.create(id+modelID,modelName);
}catch (Exception e){
  throw new EJBException(e.getMessage());
}
}//end

//private Document getDocument(Reader model){
  private Document getDocument(InputStream model){
   Document doc = null;
   InputSource inputSource = new InputSource(model);
  try{
 DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
 DocumentBuilder builder = factory.newDocumentBuilder();
 doc = builder.parse(inputSource);

  }
catch (FactoryConfigurationError f) {
  System.err.print(f);
}
catch (ParserConfigurationException p) {
  System.err.print(p);
}
catch (SAXException s) {
```

```
  System.err.print(s);
}
catch (IOException i) {
}

  return doc;
}//end parseModel


public void ejbActivate() throws EJBException,RemoteException{
}
public void ejbPassivate() throws EJBException,RemoteException {
}
public void ejbRemove() throws EJBException,RemoteException{
}
public void ejbCreate() throws EJBException,RemoteException {
try{
   modelLocalHome = lookupDb();
   functionDefinitionsHome = (FunctionDefinitionsHome)
lookUpGeneral("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/FunctionDefinitionsHome");
}catch (NamingException n){
n.printStackTrace();
}
}
public void setSessionContext(SessionContext context)
throws EJBException,RemoteException{
this.context = context;
}

private ModelLocalHome lookupDb() throws NamingException{

  Context initial = new InitialContext();
  Object objref = initial.lookup("java:comp/env/ejb/bmr/db/ModelLocalHome");

  return (ModelLocalHome) objref;
   }
private Object lookUpGeneral(String typeLookUp) throws NamingException{

Context initial = new InitialContext();
Object objref = initial.lookup(typeLookUp);
  return objref;
}//end Object lookUpGeneral()
}
```

D.5 Get Model from Database

This appendix shows the bmr.ejb.service.SbmlFileBean

```
package bmr.ejb.service;
import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import javax.ejb.FinderException;
import javax.ejb.CreateException;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import java.io.File;
import bmr.ejb.sbml2.interfaces.*;
import bmr.db.*;
import bmr.utils.sbml2.*;
import bmr.utils.dbkeys.ModelPK;
/**
 * Bean class to construct a sbml model
 */
public class SbmlFileBean implements SessionBean {
private SessionContext context;
private String sbmlString;
private ModelLocalHome modelLocalHome;
private FunctionDefinitionsHome functionDefinitionsHome;
private UnitDefinitionsHome unitDefinitionsHome;
private CompartmentsHome compartmentsHome;
```

```
private SpeciesHome speciesHome;
private ParametersHome parametersHome;
private RulesHome rulesHome;
private ReactionsHome reactionsHome;
private EventsHome eventsHome;
/**
 * Method to get a model from the database
 * @param modelID id for a model stored in the database
 * @return
 * @throws EJBException
 */
public String getSbmlModel(String modelID) throws EJBException {
File f = null;
sbmlString = SbmlParameters.XML_FILE_INFO;
sbmlString += "\n";
sbmlString += SbmlParameters.SBML_FILE_BEGIN+"\n";
sbmlString += "<model id=\""+modelID+"\" ";
if(getModelName(modelID) != null)
sbmlString += "name=\""+getModelName(modelID)+"\"> \n";
else
sbmlString += "> \n";
/** listOfFunctionDefinitions **/
sbmlString += getFunctionDefinitions(modelID);
/** listOfUnitDefinitions **/
sbmlString += getUnitDefinitions(modelID);
/** Compartments **/
sbmlString += getCompartments(modelID);
/** Species **/
sbmlString += getSpecies(modelID);
/** Parameters **/
sbmlString += getParameters(modelID);
/** Rules **/
sbmlString += getRules(modelID);
/** Reactions **/
sbmlString += getReactions(modelID);
/** Events **/
sbmlString += getEvents(modelID);
sbmlString += SbmlParameters.SBML_FILE_END;

return sbmlString;
}
/**
 *
 * @param modelID
 * @return
 */
private String getEvents(String modelID){
String events = null;
try{
eventsHome = (EventsHome)
lookUpGeneral("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/EventsHome");
Events ev = eventsHome.create();
events = ev.getEvents(modelID);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException f){
f.printStackTrace();
}

return events;
}
/**
 *
 * @param modelID
 * @return
 */
private String getReactions(String modelID){
String reactions = null;
try{
reactionsHome = (ReactionsHome)
lookUpGeneral("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/ReactionsHome");
Reactions rea = reactionsHome.create();
reactions = rea.getReactions(modelID);
}catch (NamingException n){
```

```
n.printStackTrace();
}catch (CreateException f){
f.printStackTrace();
}

return reactions;
}
/**
 *
 * @param modelID
 * @return
 */
private String getRules(String modelID){
String rules = null;
try{
rulesHome = (RulesHome)
lookUpGeneral("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/RulesHome");
Rules ru = rulesHome.create();
rules = ru.getRules(modelID);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException f){
f.printStackTrace();
}

return rules;
}
/**
 *
 * @param modelID
 * @return
 */
private String getParameters(String modelID){
String parameters = null;
try{
parametersHome = (ParametersHome)
lookUpGeneral("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/ParametersHome");
Parameters param = parametersHome.create();
parameters = param.getParameters(modelID);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException f){
f.printStackTrace();
}

return parameters;
}
/**
 *
 * @param modelID
 * @return
 */
private String getSpecies(String modelID){
String species = null;
try{
speciesHome = (SpeciesHome)
lookUpGeneral("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/SpeciesHome");
Species spe = speciesHome.create();
species = spe.getSpecies(modelID);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException f){
f.printStackTrace();
}
return species;
}
/**
 *
 * @param modelID
 * @return
 */
private String getCompartments(String modelID){
String compartments = null;
try{
```

```
compartmentsHome  = (CompartmentsHome)
lookUpGeneral("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/CompartmentsHome");
Compartments comp = compartmentsHome.create();
compartments = comp.getCompartments(modelID);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException f){
f.printStackTrace();
}


return compartments;
}
/**
 *
 * @param modelID
 * @return
 */
private String getUnitDefinitions(String modelID){
String unitDefinitions = null;
try{
unitDefinitionsHome = (UnitDefinitionsHome)
lookUpGeneral("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/UnitDefinitionsHome");
UnitDefinitions unitDef = unitDefinitionsHome.create();
unitDefinitions = unitDef.getUnitDefinitons(modelID);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException f){
f.printStackTrace();
}

return unitDefinitions;
}
/**
 *
 * @param modelID id for the current model
 * @return Returns functionDefinitionns
 */
private String getFunctionDefinitions(String modelID){
String functionDefinitions = null;
try{
functionDefinitionsHome = (FunctionDefinitionsHome)
lookUpGeneral("java:comp/env/ejb/bmr/ejb/sbml2/interfaces/FunctionDefinitionsHome");
FunctionDefinitions functionDef = functionDefinitionsHome.create();
functionDefinitions = functionDef.getFunctionDefinitions(modelID);
}catch (NamingException n){
n.printStackTrace();
}catch (CreateException f){
f.printStackTrace();
}

return functionDefinitions;
}
/**
 *
 * @param modelID id for the current model
 * @return Returns model name
 */
private String getModelName(String modelID){
String name = null;
try{
if(modelLocalHome == null)
modelLocalHome = (ModelLocalHome)
lookUpGeneral("java:comp/env/ejb/bmr/db/ModelLocalHome");
ModelLocal modelLocal = modelLocalHome.findByPrimaryKey(new ModelPK(modelID));
name = modelLocal.getModelName();
}catch (NamingException n){
n.printStackTrace();
}catch (FinderException f){
f.printStackTrace();
}

return name;
}
```

```
public void ejbCreate() throws EJBException{
}
public void ejbActivate() throws EJBException {
}
public void ejbPassivate() throws EJBException{
}
public void ejbRemove() throws EJBException {
}
public void setSessionContext(SessionContext context)
throws EJBException {
this.context = context;
}
private Object lookUpGeneral(String typeLookUp) throws NamingException{
Context initial = new InitialContext();
Object objref = initial.lookup(typeLookUp);
return objref;
}//end Object lookUpGeneral()
}
```

E The ejb-jar.xml configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems,
Inc.//DTD Enterprise JavaBeans 2.0//EN" "
http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
<description>Biological Model Repository</description>
<display-name>Biological Model Repository</display-name>
  <enterprise-beans>

<!-- ============ -->
<!-- Entity beans -->
<!-- ============ -->

 <entity>
<ejb-name>ModelLocal</ejb-name>
<local-home>bmr.db.ModelLocalHome</local-home>
<local>bmr.db.ModelLocal</local>
<ejb-class>bmr.db.ModelBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.ModelPK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>ModelBean</abstract-schema-name>
        <cmp-field>
<field-name>modelID</field-name>
</cmp-field>
<cmp-field>
<field-name>modelName</field-name>
</cmp-field>
<query>
   <query-method>
    <method-name>findAllModels</method-name>
        <method-params></method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM ModelBean AS o]]>
   </ejb-ql>
  </query>
   </entity>

   <entity>
<ejb-name>FunctionDefLocal</ejb-name>
<local-home>bmr.db.FunctionDefLocalHome</local-home>
<local>bmr.db.FunctionDefLocal</local>
<ejb-class>bmr.db.FunctionDefBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.FunctionDefPK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>FunctionDefBean</abstract-schema-name>
        <cmp-field>
<field-name>functionID</field-name>
</cmp-field>
<cmp-field>
<field-name>functionName</field-name>
</cmp-field>
    <cmp-field>
<field-name>modelID</field-name>
</cmp-field>
<cmp-field>
<field-name>mathFunction</field-name>
</cmp-field>
    <query>
   <query-method>
    <method-name>findFunctionDef</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT DISTINCT OBJECT(o) FROM FunctionDefBean AS o WHERE o.modelID = ?1]]>
   </ejb-ql>
```

```
  </query>
   </entity>

   <entity>
<ejb-name>UnitDefLocal</ejb-name>
<local-home>bmr.db.UnitDefHome</local-home>
<local>bmr.db.UnitDefLocal</local>
<ejb-class>bmr.db.UnitDefBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.UnitDefPK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>UnitDefBean</abstract-schema-name>
        <cmp-field>
<field-name>unitID</field-name>
</cmp-field>
<cmp-field>
<field-name>unitName</field-name>
</cmp-field>
    <cmp-field>
<field-name>modelID</field-name>
</cmp-field>
<cmp-field>
<field-name>units</field-name>
</cmp-field>
    <query>
   <query-method>
    <method-name>findUnitDef</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM UnitDefBean AS o WHERE o.modelID LIKE ?1]]>
   </ejb-ql>
  </query>

   </entity>

      <entity>
<ejb-name>CompartmentLocal</ejb-name>
<local-home>bmr.db.CompartmentLocalHome</local-home>
<local>bmr.db.CompartmentLocal</local>
<ejb-class>bmr.db.CompartmentBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.CompartmentPK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>CompartmentBean</abstract-schema-name>
        <cmp-field>
<field-name>compartmentID</field-name>
</cmp-field>
<cmp-field>
<field-name>compartmentName</field-name>
</cmp-field>
    <cmp-field>
<field-name>spatialDimensions</field-name>
</cmp-field>
<cmp-field>
<field-name>size</field-name>
</cmp-field>
<cmp-field>
<field-name>units</field-name>
</cmp-field>
<cmp-field>
<field-name>outSide</field-name>
</cmp-field>
<cmp-field>
<field-name>constant</field-name>
</cmp-field>
<cmp-field>
<field-name>modelID</field-name>
</cmp-field>
    <query>
```

```
   <query-method>
    <method-name>findCompartments</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM CompartmentBean AS o WHERE o.modelID LIKE ?1]]>
   </ejb-ql>
  </query>
   </entity>

   <entity>
<ejb-name>SpeciesLocal</ejb-name>
<local-home>bmr.db.SpeciesLocalHome</local-home>
<local>bmr.db.SpeciesLocal</local>
<ejb-class>bmr.db.SpeciesBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.SpeciesPK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>SpeciesBean</abstract-schema-name>
        <cmp-field>
<field-name>speciesID</field-name>
</cmp-field>
<cmp-field>
<field-name>speciesName</field-name>
</cmp-field>
    <cmp-field>
<field-name>compartment</field-name>
</cmp-field>
<cmp-field>
<field-name>initialAmount</field-name>
</cmp-field>
<cmp-field>
<field-name>initialConcentration</field-name>
</cmp-field>
<cmp-field>
<field-name>substanceUnits</field-name>
</cmp-field>
<cmp-field>
<field-name>spatialSizeUnits</field-name>
</cmp-field>
<cmp-field>
<field-name>hasOnlySubstanceUnits</field-name>
</cmp-field>
<cmp-field>
<field-name>boundaryCondition</field-name>
</cmp-field>
<cmp-field>
<field-name>charge</field-name>
</cmp-field>
<cmp-field>
<field-name>constant</field-name>
</cmp-field>
<cmp-field>
<field-name>modelID</field-name>
</cmp-field>
    <query>
   <query-method>
    <method-name>findSpecies</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM SpeciesBean AS o where o.modelID LIKE ?1]]>
   </ejb-ql>
  </query>

  </entity>
  <entity>
<ejb-name>ParameterLocal</ejb-name>
<local-home>bmr.db.ParameterLocalHome</local-home>
```

```
<local>bmr.db.ParameterLocal</local>
<ejb-class>bmr.db.ParameterBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.ParameterPK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>ParameterBean</abstract-schema-name>
        <cmp-field>
<field-name>parameterID</field-name>
</cmp-field>
<cmp-field>
<field-name>parameterName</field-name>
</cmp-field>
    <cmp-field>
<field-name>value</field-name>
</cmp-field>
<cmp-field>
<field-name>units</field-name>
</cmp-field>
<cmp-field>
<field-name>constant</field-name>
</cmp-field>
<cmp-field>
<field-name>modelID</field-name>
</cmp-field>
    <query>
   <query-method>
    <method-name>findParameters</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM ParameterBean AS o where o.modelID LIKE ?1]]>
   </ejb-ql>
  </query>
  </entity>
  <entity>
<ejb-name>RuleLocal</ejb-name>
<local-home>bmr.db.RuleLocalHome</local-home>
<local>bmr.db.RuleLocal</local>
<ejb-class>bmr.db.RuleBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.RulePK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>RuleBean</abstract-schema-name>
        <cmp-field>
<field-name>ruleID</field-name>
</cmp-field>
<cmp-field>
<field-name>ruleType</field-name>
</cmp-field>
    <cmp-field>
<field-name>math</field-name>
</cmp-field>
<cmp-field>
<field-name>modelID</field-name>
</cmp-field>
    <query>
   <query-method>
    <method-name>findRules</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM RuleBean AS o where o.modelID LIKE ?1]]>
   </ejb-ql>
  </query>

   </entity>
   <entity>
<ejb-name>KineticLawLocal</ejb-name>
```

```
<local-home>bmr.db.KineticLawLocalHome</local-home>
<local>bmr.db.KineticLawLocal</local>
<ejb-class>bmr.db.KineticLawBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.KineticLawPK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>KineticLawBean</abstract-schema-name>
        <cmp-field>
<field-name>kineticLawID</field-name>
</cmp-field>
<cmp-field>
<field-name>math</field-name>
</cmp-field>
    <cmp-field>
<field-name>timeUnits</field-name>
</cmp-field>
<cmp-field>
<field-name>substanceUnits</field-name>
</cmp-field>
<cmp-field>
<field-name>reactionID</field-name>
</cmp-field>
    <query>
   <query-method>
    <method-name>findKineticLaw</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM KineticLawBean AS o where o.reactionID LIKE ?1]]>
   </ejb-ql>
  </query>

   </entity>
   <entity>
<ejb-name>SpeciesRefLocal</ejb-name>
<local-home>bmr.db.SpeciesRefLocalHome</local-home>
<local>bmr.db.SpeciesRefLocal</local>
<ejb-class>bmr.db.SpeciesRefBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.SpeciesRefPK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>SpeciesRefBean</abstract-schema-name>
        <cmp-field>
<field-name>speciesRefID</field-name>
</cmp-field>
<cmp-field>
<field-name>stoichiometry</field-name>
</cmp-field>
    <cmp-field>
<field-name>stoichiometryMath</field-name>
</cmp-field>
<cmp-field>
<field-name>reaction</field-name>
</cmp-field>
<cmp-field>
<field-name>type</field-name>
</cmp-field>
    <query>
   <query-method>
    <method-name>findSpeciesRef</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM SpeciesRefBean AS o where o.reaction LIKE ?1]]>
   </ejb-ql>
  </query>

   </entity>
```

```
   <entity>
<ejb-name>ReactionLocal</ejb-name>
<local-home>bmr.db.ReactionLocalHome</local-home>
<local>bmr.db.ReactionLocal</local>
<ejb-class>bmr.db.ReactionBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.ReactionPK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>ReactionBean</abstract-schema-name>
        <cmp-field>
<field-name>reactionID</field-name>
</cmp-field>
<cmp-field>
<field-name>name</field-name>
</cmp-field>
    <cmp-field>
<field-name>reversible</field-name>
</cmp-field>
<cmp-field>
<field-name>fast</field-name>
</cmp-field>
<cmp-field>
<field-name>modelID</field-name>
</cmp-field>
    <query>
   <query-method>
    <method-name>findReactions</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM ReactionBean AS o where o.modelID LIKE ?1]]>
   </ejb-ql>
  </query>

   </entity>
   <entity>
<ejb-name>EventLocal</ejb-name>
<local-home>bmr.db.EventLocalHome</local-home>
<local>bmr.db.EventLocal</local>
<ejb-class>bmr.db.EventBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.EventPK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>EventBean</abstract-schema-name>
        <cmp-field>
<field-name>eventID</field-name>
</cmp-field>
<cmp-field>
<field-name>eventName</field-name>
</cmp-field>
    <cmp-field>
<field-name>trigger</field-name>
</cmp-field>
<cmp-field>
<field-name>delay</field-name>
</cmp-field>
<cmp-field>
<field-name>timeUnits</field-name>
</cmp-field>
<cmp-field>
<field-name>modelID</field-name>
</cmp-field>
    <query>
   <query-method>
    <method-name>findEvents</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
```

```
     <![CDATA[SELECT OBJECT(o) FROM EventBean AS o where o.modelID LIKE ?1]]>
   </ejb-ql>
  </query>
   </entity>

   <entity>
<ejb-name>EventAssignmentLocal</ejb-name>
<local-home>bmr.db.EventAssignmentLocalHome</local-home>
<local>bmr.db.EventAssignmentLocal</local>
<ejb-class>bmr.db.EventAssignmentBean</ejb-class>
<persistence-type>Container</persistence-type>
<prim-key-class>bmr.utils.dbkeys.EventAssignmentPK</prim-key-class>
<reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
<abstract-schema-name>EventAssignmentBean</abstract-schema-name>
        <cmp-field>
<field-name>eventAssignmentID</field-name>
</cmp-field>
<cmp-field>
<field-name>math</field-name>
</cmp-field>
    <cmp-field>
<field-name>eventID</field-name>
</cmp-field>
    <query>
   <query-method>
    <method-name>findEventAssignments</method-name>
        <method-params>
        <method-param>java.lang.String</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
     <![CDATA[SELECT OBJECT(o) FROM EventAssignmentBean AS o where o.eventID LIKE ?1]]>
   </ejb-ql>
  </query>

   </entity>

<!-- ============= -->
<!-- Session beans -->
<!-- ============= -->

   <session>
    <ejb-name>BmrFacade</ejb-name>
    <home>bmr.ejb.facade.BmrFacadeHome</home>
    <remote>bmr.ejb.facade.BmrFacade</remote>
    <ejb-class>bmr.ejb.facade.BmrFacadeBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
     <ejb-local-ref>
     <description>Reference to SbmlFile</description>
     <ejb-ref-name>ebj/bmr/ejb/service/SbmlFileLocalHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
     <local-home>bmr.ejb.service.SbmlFileLocalHome</local-home>
     <local>bmr.ejb.service.SbmlFile</local>
     <ejb-link>SbmlFile</ejb-link>
     </ejb-local-ref>
        <ejb-local-ref>
           <description>Reference to database</description>
   <ejb-ref-name>ejb/bmr/db/ModelLocalHome</ejb-ref-name>
   <ejb-ref-type>Entity</ejb-ref-type>
      <local-home>bmr.db.ModelLocalHome</local-home>
   <local>bmr.db.ModelLocal</local>
   <ejb-link>ModelLocal</ejb-link>
    </ejb-local-ref>
   </session>

   <session>
     <ejb-name>SbmlFile</ejb-name>
     <local-home>bmr.ejb.service.SbmlFileLocalHome</local-home>
     <local>bmr.ejb.service.SbmlFile</local>
     <ejb-class>bmr.ejb.service.SbmlFileBean</ejb-class>
     <session-type>Stateless</session-type>
     <transaction-type>Bean</transaction-type>
```

```
       <ejb-local-ref>
            <description>Reference to database</description>
   <ejb-ref-name>ejb/bmr/db/ModelLocalHome</ejb-ref-name>
   <ejb-ref-type>Entity</ejb-ref-type>
      <local-home>bmr.db.ModelLocalHome</local-home>
   <local>bmr.db.ModelLocal</local>
   <ejb-link>ModelLocal</ejb-link>
   </ejb-local-ref>
<ejb-local-ref>
         <description>Reference to EventAssignments</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/EventAssignmentsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.EventAssignmentsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.EventAssignments</local>
  <ejb-link>EventAssignments</ejb-link>
</ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to SpeciesRef</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/SpeciesRefHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.SpeciesRefHome</local-home>
<local>bmr.ejb.sbml2.interfaces.SpeciesRef</local>
<ejb-link>SpeciesRef</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to KineticLaw</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/KineticLawHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.KineticLawHome</local-home>
<local>bmr.ejb.sbml2.interfaces.KineticLaw</local>
<ejb-link>KineticLaw</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Species</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/SpeciesHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.SpeciesHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Species</local>
<ejb-link>Species</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Compartments</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/CompartmentsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.CompartmentsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Compartments</local>
<ejb-link>Compartments</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Event</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/EventsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.EventsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Events</local>
<ejb-link>Events</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to FunctionDefinitions</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/FunctionDefinitionsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.FunctionDefinitionsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.FunctionDefinitions</local>
<ejb-link>FunctionDefinitions</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Parameters</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/ParametersHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.ParametersHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Parameters</local>
<ejb-link>Parameters</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Reactions</description>
```

```
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/ReactionsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.ReactionsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Reactions</local>
<ejb-link>Reactions</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Rules</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/RulesHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.RulesHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Rules</local>
<ejb-link>Rules</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Species</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/SpeciesHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.SpeciesHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Species</local>
<ejb-link>Species</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to UnitDefinitions</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/UnitDefinitionsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.UnitDefinitionsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.UnitDefinitions</local>
<ejb-link>UnitDefinitions</ejb-link>
    </ejb-local-ref>

   </session>

   <session>
    <ejb-name>InsertIntoDb</ejb-name>
    <home>bmr.ejb.service.InsertIntoDbLocalHome</home>
    <remote>bmr.ejb.service.InsertIntoDb</remote>
    <ejb-class>bmr.ejb.service.InsertIntoDbBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
        <ejb-local-ref>
           <description>Reference to database</description>
   <ejb-ref-name>ejb/bmr/db/ModelLocalHome</ejb-ref-name>
   <ejb-ref-type>Entity</ejb-ref-type>
      <local-home>bmr.db.ModelLocalHome</local-home>
   <local>bmr.db.ModelLocal</local>
   <ejb-link>ModelLocal</ejb-link>
</ejb-local-ref>
<ejb-local-ref>
         <description>Reference to EventAssignments</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/EventAssignmentsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.EventAssignmentsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.EventAssignments</local>
  <ejb-link>EventAssignments</ejb-link>
</ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to SpeciesRef</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/SpeciesRefHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.SpeciesRefHome</local-home>
<local>bmr.ejb.sbml2.interfaces.SpeciesRef</local>
<ejb-link>SpeciesRef</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to KineticLaw</description>
         <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/KineticLawHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.KineticLawHome</local-home>
<local>bmr.ejb.sbml2.interfaces.KineticLaw</local>
<ejb-link>KineticLaw</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Species</description>
```

```
        <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/SpeciesHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.SpeciesHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Species</local>
<ejb-link>Species</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Compartments</description>
        <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/CompartmentsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.CompartmentsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Compartments</local>
<ejb-link>Compartments</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Event</description>
        <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/EventsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.EventsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Events</local>
<ejb-link>Events</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to FunctionDefinitions</description>
        <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/FunctionDefinitionsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.FunctionDefinitionsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.FunctionDefinitions</local>
<ejb-link>FunctionDefinitions</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Parameters</description>
        <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/ParametersHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.ParametersHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Parameters</local>
<ejb-link>Parameters</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Reactions</description>
        <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/ReactionsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.ReactionsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Reactions</local>
<ejb-link>Reactions</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Rules</description>
        <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/RulesHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.RulesHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Rules</local>
<ejb-link>Rules</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to Species</description>
        <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/SpeciesHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.SpeciesHome</local-home>
<local>bmr.ejb.sbml2.interfaces.Species</local>
<ejb-link>Species</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to UnitDefinitions</description>
        <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/UnitDefinitionsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.UnitDefinitionsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.UnitDefinitions</local>
<ejb-link>UnitDefinitions</ejb-link>
    </ejb-local-ref>
 </session>
   <session>
    <ejb-name>Compartments</ejb-name>
    <local-home>bmr.ejb.sbml2.interfaces.CompartmentsHome</local-home>
```

```
    <local>bmr.ejb.sbml2.interfaces.Compartments</local>
    <ejb-class>bmr.ejb.sbml2.CompartmentBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
    <ejb-local-ref>
      <description>Reference to database</description>
        <ejb-ref-name>ejb/bmr/db/CompartmentLocalHome</ejb-ref-name>
     <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>bmr.db.CompartmentLocalHome</local-home>
<local>bmr.db.CompartmentLocal</local>
<ejb-link>CompartmentLocal</ejb-link>
    </ejb-local-ref>
   </session>
   <session>
    <ejb-name>Events</ejb-name>
    <local-home>bmr.ejb.sbml2.interfaces.EventsHome</local-home>
    <local>bmr.ejb.sbml2.interfaces.Events</local>
    <ejb-class>bmr.ejb.sbml2.EventBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
    <ejb-local-ref>
      <description>Reference to database</description>
        <ejb-ref-name>ejb/bmr/db/EventLocalHome</ejb-ref-name>
     <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>bmr.db.EventLocalHome</local-home>
<local>bmr.db.EventLocal</local>
<ejb-link>EventLocal</ejb-link>
    </ejb-local-ref>
        <ejb-local-ref>
      <description>Reference to EventAssignments</description>
        <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/EventAssignmentsHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.EventAssignmentsHome</local-home>
<local>bmr.ejb.sbml2.interfaces.EventAssignments</local>
<ejb-link>EventAssignments</ejb-link>
    </ejb-local-ref>
   </session>
   <session>
    <ejb-name>EventAssignments</ejb-name>
    <local-home>bmr.ejb.sbml2.interfaces.EventAssignmentsHome</local-home>
    <local>bmr.ejb.sbml2.interfaces.EventAssignments</local>
    <ejb-class>bmr.ejb.sbml2.EventAssignmentBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
    <ejb-local-ref>
      <description>Reference to database</description>
        <ejb-ref-name>ejb/bmr/db/EventAssignmentHome</ejb-ref-name>
     <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>bmr.db.EventAssignmentLocalHome</local-home>
<local>bmr.db.EventAssignmentLocal</local>
<ejb-link>EventAssignmentLocal</ejb-link>
    </ejb-local-ref>
    </session>
    <session>
   <ejb-name>FunctionDefinitions</ejb-name>
    <local-home>bmr.ejb.sbml2.interfaces.FunctionDefinitionsHome</local-home>
    <local>bmr.ejb.sbml2.interfaces.FunctionDefinitions</local>
    <ejb-class>bmr.ejb.sbml2.FunctionDefinitionBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
    <ejb-local-ref>
      <description>Reference to database</description>
        <ejb-ref-name>ejb/bmr/db/FunctionDefLocalHome</ejb-ref-name>
     <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>bmr.db.FunctionDefLocalHome</local-home>
<local>bmr.db.FunctionDefLocal</local>
<ejb-link>FunctionDefLocal</ejb-link>
    </ejb-local-ref>
   </session>
   <session>
    <ejb-name>Parameters</ejb-name>
    <local-home>bmr.ejb.sbml2.interfaces.ParametersHome</local-home>
    <local>bmr.ejb.sbml2.interfaces.Parameters</local>
    <ejb-class>bmr.ejb.sbml2.ParameterBean</ejb-class>
```

```
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
    <ejb-local-ref>
      <description>Reference to database</description>
        <ejb-ref-name>ejb/bmr/db/ParameterLocalHome</ejb-ref-name>
     <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>bmr.db.ParameterLocalHome</local-home>
<local>bmr.db.ParameterLocal</local>
<ejb-link>ParameterLocal</ejb-link>
    </ejb-local-ref>
   </session>
   <session>
    <ejb-name>KineticLaw</ejb-name>
    <local-home>bmr.ejb.sbml2.interfaces.KineticLawHome</local-home>
    <local>bmr.ejb.sbml2.interfaces.KineticLaw</local>
    <ejb-class>bmr.ejb.sbml2.KineticLawBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
    <ejb-local-ref>
      <description>Reference to database</description>
        <ejb-ref-name>ejb/bmr/db/KineticLawLocalHome</ejb-ref-name>
     <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>bmr.db.KineticLawLocalHome</local-home>
<local>bmr.db.KineticLawLocal</local>
<ejb-link>KineticLawLocal</ejb-link>
    </ejb-local-ref>
   </session>
   <session>
    <ejb-name>Rules</ejb-name>
    <local-home>bmr.ejb.sbml2.interfaces.RulesHome</local-home>
    <local>bmr.ejb.sbml2.interfaces.Rules</local>
    <ejb-class>bmr.ejb.sbml2.RuleBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
    <ejb-local-ref>
      <description>Reference to database</description>
        <ejb-ref-name>ejb/bmr/db/RuleLocalHome</ejb-ref-name>
     <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>bmr.db.RuleLocalHome</local-home>
<local>bmr.db.RuleLocal</local>
<ejb-link>RuleLocal</ejb-link>
    </ejb-local-ref>
   </session>
   <session>
    <ejb-name>Species</ejb-name>
    <local-home>bmr.ejb.sbml2.interfaces.SpeciesHome</local-home>
    <local>bmr.ejb.sbml2.interfaces.Species</local>
    <ejb-class>bmr.ejb.sbml2.SpecieBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
    <ejb-local-ref>
      <description>Reference to database</description>
        <ejb-ref-name>ejb/bmr/db/SpeciesLocalHome</ejb-ref-name>
     <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>bmr.db.SpeciesLocalHome</local-home>
<local>bmr.db.SpeciesLocal</local>
<ejb-link>SpeciesLocal</ejb-link>
    </ejb-local-ref>
   </session>
   <session>
    <ejb-name>UnitDefinitions</ejb-name>
    <local-home>bmr.ejb.sbml2.interfaces.UnitDefinitionsHome</local-home>
    <local>bmr.ejb.sbml2.interfaces.UnitDefinitions</local>
    <ejb-class>bmr.ejb.sbml2.UnitDefinitionBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
    <ejb-local-ref>
      <description>Reference to database</description>
        <ejb-ref-name>ejb/bmr/db/UnitDefHome</ejb-ref-name>
     <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>bmr.db.UnitDefHome</local-home>
<local>bmr.db.UnitDefLocal</local>
<ejb-link>UnitDefLocal</ejb-link>
    </ejb-local-ref>
```

```
   </session>
   <session>
    <ejb-name>SpeciesRef</ejb-name>
    <local-home>bmr.ejb.sbml2.interfaces.SpeciesRefHome</local-home>
    <local>bmr.ejb.sbml2.interfaces.SpeciesRef</local>
    <ejb-class>bmr.ejb.sbml2.SpeciesRefBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
    <ejb-local-ref>
      <description>Reference to database</description>
        <ejb-ref-name>ejb/bmr/db/SpeciesRefLocalHome</ejb-ref-name>
     <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>bmr.db.SpeciesRefLocalHome</local-home>
<local>bmr.db.SpeciesRefLocal</local>
<ejb-link>SpeciesRefLocal</ejb-link>
    </ejb-local-ref>
   </session>
   <session>
    <ejb-name>Reactions</ejb-name>
    <local-home>bmr.ejb.sbml2.interfaces.ReactionsHome</local-home>
    <local>bmr.ejb.sbml2.interfaces.Reactions</local>
    <ejb-class>bmr.ejb.sbml2.ReactionBean</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Bean</transaction-type>
    <ejb-local-ref>
      <description>Reference to database</description>
        <ejb-ref-name>ejb/bmr/db/ReactionLocalHome</ejb-ref-name>
     <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>bmr.db.ReactionLocalHome</local-home>
<local>bmr.db.ReactionLocal</local>
<ejb-link>ReactionLocal</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to SpeciesRef</description>
        <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/SpeciesRefHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.SpeciesRefHome</local-home>
<local>bmr.ejb.sbml2.interfaces.SpeciesRef</local>
<ejb-link>SpeciesRef</ejb-link>
    </ejb-local-ref>
    <ejb-local-ref>
      <description>Reference to KineticLaw</description>
        <ejb-ref-name>ejb/bmr/ejb/sbml2/interfaces/KineticLawHome</ejb-ref-name>
     <ejb-ref-type>Session</ejb-ref-type>
   <local-home>bmr.ejb.sbml2.interfaces.KineticLawHome</local-home>
<local>bmr.ejb.sbml2.interfaces.KineticLaw</local>
<ejb-link>KineticLaw</ejb-link>
    </ejb-local-ref>

   </session>


<!-- ================== -->
<!-- Message Driven Beans -->
<!-- ================== -->
   <message-driven>
    <ejb-name>BmrFacadeMDB</ejb-name>
<ejb-class>bmr.ejb.facade.BmrFacadeMDB</ejb-class>
<transaction-type>Container</transaction-type>
<message-driven-destination>
  <destination-type>javax.jms.Queue</destination-type>
</message-driven-destination>
  <ejb-ref>
        <description> Reference to InsertIntoDbBean </description>
        <ejb-ref-name>ejb/bmr/ejb/service/InsertIntoDbLocalHome</ejb-ref-name>
        <ejb-ref-type>Session</ejb-ref-type>
        <home>bmr.ejb.service.InsertIntoDbLocalHome</home>
        <remote>bmr.ejb.service.InsertIntoDb</remote>
        <ejb-link>InsertIntoDb</ejb-link>
  </ejb-ref>
   </message-driven>

  </enterprise-beans>
</ejb-jar>
```

F The jboss.xml configuration file

This config file is only used for the purpose of the Message Driven Bean. It configurates the queue for this bean.

```
<?xml version="1.0" encoding="UTF-8" ?>
<jboss>
  <enterprise-beans>
    <message-driven>
      <ejb-name>BmrFacadeMDB</ejb-name>
      <configuration-name>Standard Message Driven Bean</configuration-name>
      <destination-jndi-name>queue/BmrFacadeQueue</destination-jndi-name>
    </message-driven>
  </enterprise-beans>
</jboss>
```

G Overview of MySQL database tables

This appendix shows create statements for MySQL database. These are generated from Entity Bean by the jboss server during booting.

```
CREATE TABLE ModelLocal (
  modelID varchar(250) binary NOT NULL default '',
  modelName varchar(250) binary default NULL,
  PRIMARY KEY  (modelID)
) TYPE=MyISAM;

CREATE TABLE FunctionDefLocal (
  functionID varchar(250) binary NOT NULL default '',
  functionName varchar(250) binary default NULL,
  modelID varchar(250) binary default NULL,
  mathFunction longblob,
  PRIMARY KEY  (functionID)
) TYPE=MyISAM;

CREATE TABLE UnitDefLocal (
  unitID varchar(250) binary NOT NULL default '',
  unitName varchar(250) binary default NULL,
  modelID varchar(250) binary default NULL,
  units longblob,
  PRIMARY KEY  (unitID)
) TYPE=MyISAM;

CREATE TABLE CompartmentLocal (
  compartmentID varchar(250) binary NOT NULL default '',
  compartmentName varchar(250) binary default NULL,
  spatialDimensions int(11) default NULL,
  size double default NULL,
  units varchar(250) binary default NULL,
  outSide varchar(250) binary default NULL,
  constant tinyint(4) default NULL,
  modelID varchar(250) binary default NULL,
  PRIMARY KEY  (compartmentID)
) TYPE=MyISAM;

CREATE TABLE SpeciesLocal (
  speciesID varchar(250) binary NOT NULL default '',
  speciesName varchar(250) binary default NULL,
  compartment varchar(250) binary default NULL,
  initialAmount double default NULL,
  initialConcentration double default NULL,
  substanceUnits varchar(250) binary default NULL,
  spatialSizeUnits varchar(250) binary default NULL,
  hasOnlySubstanceUnits tinyint(4) default NULL,
  boundaryCondition tinyint(4) default NULL,
  charge int(11) default NULL,
  constant tinyint(4) default NULL,
  modelID varchar(250) binary default NULL,
  PRIMARY KEY  (speciesID)
) TYPE=MyISAM;

CREATE TABLE SpeciesRefLocal (
  speciesRefID varchar(250) binary NOT NULL default '',
  stoichiometry double default NULL,
  stoichiometryMath longblob,
  reaction varchar(250) binary default NULL,
  type varchar(250) binary default NULL,
  PRIMARY KEY  (speciesRefID)
) TYPE=MyISAM;

CREATE TABLE ParameterLocal (
  parameterID varchar(250) binary NOT NULL default '',
  parameterName varchar(250) binary default NULL,
  value double default NULL,
  units varchar(250) binary default NULL,
  constant tinyint(4) default NULL,
  modelID varchar(250) binary default NULL,
  PRIMARY KEY  (parameterID)
) TYPE=MyISAM;
```

```
CREATE TABLE RuleLocal (
  ruleID varchar(250) binary NOT NULL default '',
  ruleType varchar(250) binary default NULL,
  math longblob,
  modelID varchar(250) binary default NULL,
  PRIMARY KEY  (ruleID)
) TYPE=MyISAM;


CREATE TABLE ReactionLocal (
  reactionID varchar(250) binary NOT NULL default '',
  name varchar(250) binary default NULL,
  reversible tinyint(4) default NULL,
  fast tinyint(4) default NULL,
  modelID varchar(250) binary default NULL,
  PRIMARY KEY  (reactionID)
) TYPE=MyISAM;

CREATE TABLE KineticLawLocal (
  kineticLawID varchar(250) binary NOT NULL default '',
  math longblob,
  timeUnits varchar(250) binary default NULL,
  substanceUnits varchar(250) binary default NULL,
  reactionID varchar(250) binary default NULL,
  PRIMARY KEY  (kineticLawID)
) TYPE=MyISAM;

CREATE TABLE EventLocal (
  eventID varchar(250) binary NOT NULL default '',
  eventName varchar(250) binary default NULL,
  trigger longblob,
  delay longblob,
  timeUnits varchar(250) binary default NULL,
  modelID varchar(250) binary default NULL,
  PRIMARY KEY  (eventID)
) TYPE=MyISAM;

CREATE TABLE EventAssignmentLocal (
  eventAssignmentID varchar(250) binary NOT NULL default '',
  math longblob,
  eventID varchar(250) binary default NULL,
  PRIMARY KEY  (eventAssignmentID)
) TYPE=MyISAM;
```

H Readme file from the BMR implementation

This appendix shows the contents of the Readme file from the BMR implementation.

```
Biological Model Repository - BMR - Oslo, Norway 27. October 2004
=================================================================

This is the Biological Model Repository a part of a master
thesis at University of Oslo department of computer science.

1. What is this?
2. Catalog structure
3. Compilation & external libraries
4. Installation
5. Future & comments
6. References

1. What is this?

This is a database for storing sbml files. It is an alpha version where
not every features are implemented. The BMR is coded using EJB, JSP,
HTML and Java Servlets. For a full list of implemented futures, I
suggest looking at the BMRFacadeBean.java class.

2. Catalog structure

bmr - src catalog
doc - auto generated documentation
dist - a binary distribution jar file
bmr.war - a web application structure for bmr which can
be used in a jboss server
models - two example models in sbml
build.xml - ant file I used to make and copy the jar file
ejb-jar.xml - a ejb-jar file for jboss
readme - this file

3. Compilation & external libraries

The probably easiest thing to do when compiling this
application is to import all source files to a IDE.
If you are using something never and more fancy than
Emacs or WordPad this would be fairly easy. The BMR
uses apart from J2EE libraries in jboss and Tomcat a
fileupload package from apache (look after commons
FileUpLoad package on the jakarta apache web site).

4. Installation

In order to install and run this application you need a
jboss server, a database,a servlet server, a jwm and
a web browser. The BMR was only tested and developed
on jboss-3.2.5 integrated with Tomcat 5.0 server, using
a MySQL database (version 4.0) and Java 1.4.2. To install
without compilation just copy the jar file and bmr.war
catalog to the deploy calatog of the jboss server.
The next step is to configure a database and the jms provider.
The database is configured in the conf/standardjaws.xml and
conf/standardjbosscmp-jdbc.xml files. Additionaly one have to
add a mysql-ds.xml file to the deploy directory and a
mysql-jdbc2-service.xml file to the deploy/jms directory.
These files are used with MySQL database, if you are a using
another database you will find the other config files in the
docs/example directory.



5. Future & comments
This application won't probably be developt or maintained
in the future. However, if you have any comments or
suggestions they are welcome on the following email
adress - michalst@ifi.uio.no
```

6. References

SBML format - http://www.sbml.org
jboss server - http://www.jboss.org
Tomcat server - http://jakarta.apache.org
Common FileUpLoad - http://jakarta.apache.org
MySQL database - htpp://www.mysql.com
Java programming language - http://java.sun.com
Eclipse IDE - http://www.eclipse.org

References

[1] Systems Biology Markup Language Level 1 Version 1. http://sbml.org/specifications/sbml-level-1/version-1/html/sbml-level-1.html.

[2] Systems Biology Markup Language Level 2 Version 1. http://sbml.org/specifications/sbml-level-2/version-1/html/sbml-level-2.html.

[3] Web Content Accessibility Guidelines 1.0. http://www.w3.org/tr/wai_webcontent.

[4] Systems Biology Markup Language Level 1 Version 2. http://sbml.org/specifications/sbml-level-1/version-2/html/sbml-level-1.html.

[5] ActiveMath. http://www.activemath.org.

[6] W3C Internationalization Activity. http://www.w3.org/international/.

[7] Defense Advanced Research Project Agency. http://www.darpa.mil.

[8] Association of American Publishers. *Association of American Publishers electronic manuscript series standard for electronic manuscript preparation and markup: an SGML application conforming to International Standard ISO 8879–Standard Generalized Markup Language. Version 2.0 Dublin, Ohio.* Association of American Publishers, Dublin, OH, USA, 1987. Available from the Electronic Publishing Special Interest Group. ISBN ????

[9] Java Transaction API. http://java.sun.com.

[10] Java Applets. http://java.sun.com/applets/index.html.

[11] Enterprise Java Beans. http://java.sun.com/products/ejb/index.jsp.

[12] CellML. http://www.cellml.org.

[13] E. F. Codd. A relational model for large shared data banks. *Communications of the ACM*, 13(6):377–387, juni 1970.

[14] BioSpice Community. http://www.biospice.org.

[15] World Wide Web Consortium. http://www.w3c.org.

[16] The Oracle Cooperation. http://www.oracle.com.

[17] CoreTrek. http://www.coretrek.no.

[18] Versant Corporation. http://www.versant.com.

[19] Jasmine Object Database. http://www3.ca.com/solutions/product.asp?id=3008.

[20] Java 2 Enterprise Edition. http://java.sun.com/j2ee/index.jsp.

[21] Java 2 Standard Edition. http://java.sun.com/j2se/index.jsp.

[22] Mathematical Markup Language (MathML) Version 2.0 (Second Edition). http://www.w3.org/tr/mathml2/.

[23] Jakarta Apache Commons FileUpLoad. http://jakarta.apache.org/commons/index.html.

[24] Flash. http://www.macromedia.com/.

[25] Apache Software Foundation. http://www.apache.org.

[26] GenBank. http://www.ncbi.nih.gov.

[27] Apache Geronimo. http://geronimo.apache.org.

[28] Goods. http://www.garret.ru/ knizhnik/goods.html.

[29] Scalable Vector Graphics. http://www.w3.org/tr/svg/.

[30] Object Data Management Group. http://www.odmg.org/.

[31] Object Management Group. http://www.omg.org.

[32] Beeler H. og Reuter G.W. Recontruction of the action potential of ventricular myocardial fibres. *J. of Phys.*, 268:177–210, 1977.

[33] W3C Math Home. http://www.w3.org/math/.

[34] My homepage. http://folk.uio.no/michalst/bmr-dist/.

[35] Device Independence. http://www.w3.org/2001/di/.

[36] Web Accessibility Initiative. http://www.w3.org/wai/.

[37] DFKI German Research Center for Artificial Intelligence. http://www.dfki.de.

[38] Remote Method Invocation. http://java.sun.com.

[39] DNA Data Bank of Japan. http://www.ddbj.nig.ac.jp.

[40] ADTmag Java og XML product review. http://www.javareport.com/java/articleold.asp?id=1249&mon=2&yr=1999.

[41] JavaScript. http://wp.netscape.com/eng/javascript/.

[42] Jboss. http://www.jboss.com.

[43] JDBC. http://java.sun.com/products/jdbc/index.jsp.

[44] Jetty. http://jetty.mortbay.org/jetty/.

[45] Jonas. http://jonas.objectweb.org.

[46] Hiroaki Kitano. Systems Biology: A Brief Overview. *Nature*, 295:1662, 2002.

[47] Donald Knuth. *The TeXbook*. Addison-Wesley, first utgave, 1984. ISBN ISBN 0-201-13447-0.

[48] European Molecular Biology Laboratory. http://www.ebi.ac.uk.

[49] Chemical Markup Language. http://www.xml-cml.org/.

[50] Extensible Hyper Text Markup Language. http://www.w3c.org/markup.

[51] Systems Biology Markup Language. http://www.sbml.org.

[52] Unified Modeling Language. http://www.uml.org.

[53] XML XPath Language. http://www.w3c.org/tr/xpath.

[54] XQuery 1.0: An XML Query Language. http://www.w3.org/tr/xquery.

[55] libSBML. http://sbml.org/software/libsbml/.

[56] Floyd Marinescu. *EJB Design Patterns*. John Wiley and Sons, 2002. ISBN ISBN 0-471-20831-0.

[57] Mathematica. http://www.wolfram.com.

[58] The ServerSide Application Server Matrix. http://wwww.theserverside.com/reviews/matrix.tss.

[59] Sun Microsystems. http://www.sun.com/.

[60] CellML model of the Beeler-Reuter model. http://www.cellml.org/examples/repository/br_model_1977_doc.html.

[61] Mozilla. http://www.mozilla.org.

[62] MySQL. http://www.mysql.com.

[63] Java Naming og Directory Interface. http://java.sun.com/products/jndi/index.jsp.

[64] .NET. http://www.microsoft.com/net/.

[65] ObjectStore. http://www.objectstore.net.

[66] OpenMath. http://www.openmath.org.

[67] ozone. http://www.ozone-db.org/frames/home/what.html.

[68] Java Server Pages. http://java.sun.com/products/jsp/index.jsp.

[69] PostgreSQL. http://www.postgresql.org.

[70] Apache HTTP Server Project. http://httpd.apache.org.

[71] Ed Roman. *Mastering EJB*. John Wiley and Sons, second utgave, 2002. ISBN ISBM 0-471-41711-4.

[72] Japan Science og Technology Corporation. http://www.symbio.jst.go.jp/symbio/index.htm.

[73] Microsoft Internet Information Server. http://www.microsoft.com/windowsserver2003/iis/default.mspx.

[74] Sun Java System Application Server. http://wwws.sun.com/software/products/appsrvr/home_appsrvr.html.

[75] The ServerSide. http://wwww.theserverside.com.

[76] Java Message Service. http://java.sun.com/products/jms/.

[77] Java Servlets. http://java.sun.com/products/servlet/index.jsp.

[78] Cascading Style Sheets. http://www.w3.org/style/css/.

[79] The Systems Biology Group Stuttgart. lhttp://www.sysbio.de/projects/glossary/systems_biology.shtml.

[80] Different Styles. http://www.456bereastreet.com/archive/200410/styling_even_more_form_controls/.

[81] Jakarta Tomcat. http://jakarta.apache.org/tomcat/.

[82] The Saarland University. http://www.uni-saarland.de/en.

[83] VBScript. http://msdn.microsoft.com.

[84] The Apache Jakarta Project Velocity. http://jakarta.apache.org/velocity.

[85] WebLogic. http://www.bea.com.

[86] WebMacro. http://www.webmacro.org.

[87] Systems Biology Workbench. http://sbw.sourceforge.net.

[88] The Newsmagazine Of The Chemical World. http://pubs.acs.org/cen/coverstory/8120/8120biology.html.

[89] Apache Xindice. http://xml.apache.org/xindice.

[90] Extensible Markup Language (XML). http://www.w3.org/xml.

[91] MaXML: mouse annotation XML. http://www.bioinfo.de/isb/2003/04/0003/.

[92] Simple API for XML. http://www.saxproject.org.

[93] eXist an Open Source nativle XML database. http://exist.sourceforge.net.

[94] The Xerces XML Parser. http://xml.apache.org.