Design Document for the "Res Medicinae" Medical Information System

Address of the current main editor:
Christian Heller
Taubachstrasze 23
98714 Stuetzerbach
Thueringen
Germany
christian.heller@tuxtax.de
http://www.resmedicinae.org

$RCSfile: design.txt,v $
@version $Revision: 1.3 $ $Date: 2001/12/21 19:36:35 $ $Author: chrissy $
@author Christian Heller christian.heller@tuxtax.de
@author Karsten Hilbert karsten.hilbert@gmx.net
@author resmedicinae-deutsch@lists.sourceforge.net

Preface

This document and the whole "Res Medicinae" project arise from a common effort of many enthusiasts working in the areas of medicine and informatics, at public institutions as well as in companies.

The initial hints summed up in this paper were given from the participants of the resmedicinae-deutsch@lists.sourceforge.net mailing list. As a first step, an "Analysis Document" was set up and is since then completed step by step. On the basis of that analysis, we could create this "Design Document" which
in turn will be the basis of the final "Implementation".

We need your help! Any kind of participation is very welcome!
The more people help us, the better our free software will be for your use!
If you'd like to contribute, just contact one of the project members or use our mailing lists which can be found together with further project details on our homepage
http://www.resmedicinae.org

Contents

1 Introduction

This design paper covers all aspects that are important for the development of (parts of) the "Res Medicinae" system. However, please keep in mind that it is an ongoing effort and hence will never be complete to the end.

Res Medicinae tries to be not just another dirty, monolithic application which - without a proper architecture - would soon encounter limitations.
We are using latest design technology including patterns, layers and stick to official standards such as those of the (Healthcare) Domain Task Forces
(H)DTF (CORBAmed):
http://www.omg.org/homepages/index.htm
http://www.acl.lanl.gov/OMG/CD/corba.html
and the Good Electronic Health Record (GEHR) standard:
http://www.gehr.org
Also, HL7 is considered to what concerns the domain structure:
http://www.hl7.org

You may also want to inspect our modelling repository that uses UML notation:
http://www.resmedicinae.org/model/rep/doc/index.html
However, it is currently NOT up-to-date and uses old concepts. Please check out again later.

2 Basics

2.1 Programming Language

It is our opinion that the core of software systems of a certain size can only be developed successfully in one of the two comprising languages, C++ or Java. This applies even more, as one of our basic aims is to offer true platform independence. Hence, only Java (or C++ based on ported libraries such as Qt)
can really be considered.

C++ is all Open Source while Java's license model doesn't follow those rules. Still, there is hope that one day in the near future Sun will open up the language.

On the other hand, there are many weaknesses of the C++ language just because it has grown over the years and was extended to modern design possibilities step by step. It always again causes instability. The Java Framework, on the contrary, brings along plenty of features.

Scripting- and related languages such as PERL and Python are free but don't play in one league with C++ and Java, in our opinion.

However, since the layers of the Res Medicinae system will be divided properly, it should be not a big problem to create client applications such as for the Linux KDE/GNOME desktops in other languages, e.g. using the Qt, GTK or the wxWindows libraries, most of which are platform-independent, like Java.

2.2 Directory Structure

The system resides in the project root directory: /opt/resmedicinae or, on Windows operating systems, in: c:\programs\resmedicinae

Following symbolic links will be created on installing the system:
/etc/opt/resmedicinae/ --> /opt/resmedicinae/etc/
/usr/local/resmedicinae/ --> /opt/resmedicinae/local/

Within the system directory, one can find the following sub directories:

- /admin (project administration things)
  - /accessdata
  - /developers
  - /mailinglists
  - /projectplan (todo)
  - /workgroups
    - /administration
    - /documentation
    - /programming
    - /promotion
    - /support
    - /translation
    - /webdesign
- /bin (build/start/stop scripts for all applications and domains)
- /build (compiled .class files)
- /dist (distributable files)
  - binaries.zip
  - documentation.zip
  - all.zip
  - sources.zip
- /doc (documentation)
  - /api
  - /bugreport
  - /demo
  - /description
  - /license
  - /logos
  - /manuals
  - /opinions
  - /portals
  - /presentations
  - /screenshots
  - /support
    - /credits
    - /help
  - /tutorials
- /etc (configuration files)
- /jre (java runtime engine)
- /lib (libraries)
  - database.zip
  - enterprise.jar
  - reform.jar

```
    healthcare.jar
    reslab.jar
    resmedicinae.jar
    record.jar
/local (things which are specific to only one machine)
    /etc (configuration files)
/model
    /analysis (requirements, wishlist)
    /design (model, structure)
    /rep (UML model/diagram repository)
    /standards (includes style guides)
/share (contains default system settings as opposed to user specific settings in
./resmedicinae; stores resources that are accessible from all applications, such as forms of
the "ReForm" application)
    /addons (additional system resources provided and configured by the administrator)
        /reform
            /forms
                /germany
                    MyRezept.xml
    /apps
        /reform
            /dtd
            /forms
                /germany
                    Rezept.xml
            /pics
            /plugins
            /templates
            /tmp
            /toolbar
    /doc
        /record
            /api
            /manual
        /reform
            /api
            /manual
            /screenshots
        /reslab
            /api
            /manual
        /resmedicinae
            /api
            /manual
            /tutorial
        /sample
            /api
            /manual
            /tutorial
    /etc (application configuration files)
        record.xml
        reform.xml
        reslab.xml
```

```
        resmedicinae.xml
    /fonts
    /icons
        /record
        /reform
        /reslab
        /resmedicinae
        /sample
    /local
        /reform
            /dtd
            /forms
            /pics
            /plugins
            /templates
            /tmp
            /toolbar
    /locale
    /sounds
    /templates (doctor's letter)
    /wallpapers
/src (source .java files, implementation)
    /application
        /enterprise
            /controlling
            /humanresources
            /logistics
            /materials
            /organization
            /quality
            /sales
            /treasury
        /healthcare
            /pids
                /org
                    /resmedicinae
                        /application
                            /healthcare
                                /pids
            /record
                /org
                    /resmedicinae
                        /application
                            /healthcare
                                /record
            /reform
                /org
                    /resmedicinae
                        /application
                            /healthcare
                                /reform
                                    /presentation
            /reslab
```

/org
/resmedicinae
/application
/healthcare
/reslab

/datatransfer
/presentation (may contain not only graphical but also textual user interface, e.g. to retrieve laboratory data)
/main
/resmedicinae
/org
/resmedicinae
/application
/main
/resmedicinae
/sample
/simple
/org
/resmedicinae
/application
/sample
/simple
/domain
/enterprise
/controlling
/humanresources
/logistics
/materials
/ordering
/organization
/quality
/sales
/treasury
/healthcare
/billing
/org
/resmedicinae
/domain
/healthcare
/billing
/billing/germany/cooperativesociety
/billing/germany/private
/billing/germany/sickfund
/communication
/communication/conference
/communication/email
/communication/fax
/communication/letter
/communication/web
/configuration
/configuration/color
/configuration/font

/configuration/praxisaddress
/device
/device/cardreader
/drug
/entity
/entity/legal
/entity/living
/entity/living/patients
/entity/living/patients/address
/entity/living/patients/injections
/entity/living/patients/sickfund
/form
/form/germany
/form/germany/abrechnung
/form/germany/arbeitsunfaehigkeit
/form/germany/attest
/form/germany/berufsgenossenschaft/a13
/form/germany/berufsgenossenschaft/d13
/form/germany/berufsgenossenschaft/h13
/form/germany/berufsgenossenschaft/d9
/form/germany/berufsgenossenschaft/kd10
/form/germany/briefkopf
/form/germany/briefumschlag
/form/germany/einweisung
/form/germany/etiketten
/form/germany/gesundheitsuntersuchung
/form/germany/krankenpflege
/form/germany/krankenhausbehandlung
/form/germany/krebsfrueherkennung
/form/germany/krebsfrueherkennung/frau
/form/germany/krebsfrueherkennung/mann
/form/germany/privat
/form/germany/rezept
/form/germany/rezept/privat
/form/germany/rezept/krankenkasse
/form/germany/therapieplan
/form/germany/transport
/form/germany/ueberweisung
/form/germany/unfallmeldung
/importexport
/importexport/bdt
/inventory
/material
/paraclinical
/paraclinical/arthroscopy
/paraclinical/computertomogram
/paraclinical/bloodpressure
/paraclinical/bloodpressure/longtime
/paraclinical/electrocardiogram
/paraclinical/ecg/ergometry
/paraclinical/ecg/longtime
/paraclinical/ecg/rest
/paraclinical/endoscopy

/paraclinical/gastroscopy
/paraclinical/laboratory
/paraclinical/laryngoscopy
/paraclinical/magnetoresonance
/paraclinical/sonogram
/paraclinical/xray
/record
/record/content
/record/diagnosis
/record/diagnosis/chains
/record/diagnosis/icd
/record/history
/record/laboratory
/record/maintenance (Vorsorge)
/record/note
/record/observation
/record/prescription (Medikamente, Heilmittel wie Massage oder Bestrahlung und Hilfsmittel wie z.B. Gehstock)
/record/vitalsign
/scheduling
/scheduling/birthday
/scheduling/consultation
/scheduling/visit
/scheduling/waitingroom
/security
/specialists
/specialists/general
/specialists/ophthalmology
/specialists/orthopedy
/statistic
/statistic/age
/statistic/billing
/statistic/drugs
/statistic/group
/statistic/laboratory
/statistic/overall
/statistic/privatedaily
/statistic/sickfundbudget
/resmedlib
/org
/resmedicinae
/resmedlib
/application
ApplicationController.java
ApplicationModel.java
ApplicationView.java
/domain
DomainObject.java
/util
/logging
/preference
/printing
/tmp (temporary stuff)

All users working with Res Medicinae will find a directory ./resmedicinae in their home directory, which might be something like: /home/username.
The inner structure of ./resmedicinae will look like that:
./resmedicinae (contains user specific settings as opposed to default system settings in /share)
  /share
    /apps (application data)
      /record
      /reform
        /dtd
        /forms
        /pics
        /plugins
        /templates
        /tmp
        /toolbar
    /etc (application configuration files)
      record.xml
      reform.xml
      resmedicinae.xml

So far to what concerns the project structure.

You won't find a /web directory or the like. We have decided to integrate our web site files smoothly into the project structure, i.e. the project root /resmedicinae contains the root index.html file which may point to any sub directory.
The disadvantage of this solution is that all files belonging to the web site are spread over the whole system. However, there are also two big advantages:

1 All links will point to directories which are at a lower level than the project root. If we had our root html file in a /web directory and had to point to the /doc directory for example, we would leave the actual scope of our web site by using a link ../doc that would go higher in hierarchy than our root web file is.

2 There are no redundant files or resources. If we had an extra /web directory, it might become necessary at times, to copy files of the project (e.g. the analysis.txt document) into that /web directory to provide it also on the web site. In that case, we would have unwanted duplication. Even worse when both, the project as well as the web site would get versionated in CVS.

2.3 Application Communication

- XML files/XML pipe when call per CLI (Java Messaging Service JMS??)
- sysctl, socket when accessing a running deamon

==> the application can run local or on a remote machine

3 Software Architecture

3.1 Logical Layers

What is informatics trying to do?

It tries to model real world aspects which can be manipulated. Let us consider the "real world aspects" to be given, static models. The "manipulation" on the other hand corresponds to dynamic actions.

Figure 1: Informatics
- Application = Dynamics = manipulates domain = Tool = Time
- Domain = Statics = contains domain objects = Material = Space

Now, an "application" is some executable program (a process) that manipulates objects (data) of a special "domain". It is obvious, that a separation of those two kinds into "layers" is highly recommendable.
Nowadays, most applications have a graphical user interface that is often called "presentation" layer (graphical user interface or "view"). The other parts of application can be put into a "controller" and "model" layer. Depending on whether or not the data are stored directly in their domain objects (as it is with use of an object oriented database), an optional "dataaccess" layer becomes necessary (when using a relational database). It provides access to the data which are kept in the "datasource" layer. Finally, a system has mostly a number of supportive (external) libraries.

Figure 2: Layered Architecture

Every modern system consists of a "layered architecture". One can think of those layers as slices in a stack, as displayed in figure 2. It is even better to imagine the layers as skins of a ball that has a core inside.
You might have heard about 2-layered and 3-layered architectures. Above, we have mentioned even five layers! However, when it comes to implementation, then layers are nothing else than directories (sometimes called "packages" in UML and Java jargon).

In Res Medicinae, we have put our layers into a special package structure. The two major ones are:
/application
/domain

An additional one is our general library (framework) which contains functionality that can be used by all other packages:
/resmedlib

Figure 3: Package dependencies

Each major package has several sub packages. The application package contains:
/healthcare
/enterprise
/main
...

And the domain package has a pretty similar structure:
/healthcare

/enterprise
...

The sub packages, again, contain their own sub packages. The healthcare package of application contains for example:
/record
/reform
/reslab
which are single applications (modules) for use in healthcare. And the healthcare package of domain contains domain objects of medicine.

Whilst layers split the system horizontally, one can think of the applications as being a vertical separation within those layers. Each such application provides a special view to a domain area in the core of the system. According to UML, this view is often called "use case". An application can cover one or more use cases. One major use case mostly contains several minor use cases.

Let us now consider the purpose of each layer in detail, starting from the outside of the "ball" which represents our system, to the inside: View-Controller-Model-Domain-DataAccess-DataSource
But before, we need to introduce some more theory: the HMVC design pattern.

3.2 Application

3.2.1 Design Patterns

3.2.1.1 Model View Controller

In the history of software development, many "Design Patterns" were created. One of them has been proven to be especially useful in architecting applications with Graphical User Interface (GUI). It was first introduced in the "Smalltalk" programming language framework and has since then been successfully applied in numerous software projects. That pattern is called "Model View Controller" (MVC).

[Cai] explains it as follows:
"MVC provides clearly defined roles and responsibilities for its three constituent elements -- model, view, and controller. The view manages the screen layout -- that is, what the user interacts with and sees on the screen. The model represents the data underlying the object -- for example, the on-off state of a check box or the text string from a text field. Events cause the data in the model to change. The controller determines how the user interacts with the view in the form of commands."

3.2.1.2 Hierarchical MVC

This chapter is taken in large parts from [Cai]. Please refer to that source for further information on HMVC.

"HMVC provides a powerful yet easy-to-understand layered design methodology
for developing a complete (application) layer. While MVC provides an efficient framework for developing GUI interaction, HMVC scales it to the entire (application layer).
Some key benefits of a responsibility-based, layered architecture include:

- Defined intralayer communication and isolation from higher layers
- Defined interlayer communication with minimal coupling
- Localization of exposure to third-party code

The HMVC pattern decomposes the (application layer) into a hierarchy of parent-child MVC layers. The repetitive application of this pattern allows for a structured (application) architecture, as shown in Figure X.

Figure X: MVC layers [Cai]

The layered MVC approach assembles a fairly complex (application layer). Some of the key benefits of using HMVC reveal the benefits of object orientation. An optimally layered architecture:
- reduces dependencies between disparate parts of the program
- encourages reuse of code, components, and modules
- increases extensibility while easing maintainability
One can effectively manage the development of an application layer by ... using this robust and scalable pattern that can reduce some of the risk and provide a ready-made design foundation on which to build.

There are three key aspects of (application layer) development:
- GUI layout code: Widget layout and screen look and feel
- GUI feature code: Validations and user-event capture
- Application logic code: App flows, navigation, and server interaction

The HMVC design pattern encourages the decomposition of the (application layer) into developed, distinct layers for implementing GUI and application services. A pattern-based architecture results in standardization; the HMVC pattern standardizes the (application) (user-service) layer of Web applications.
Standardization in the (application) layer helps contribute to:
- UI consistency:
The framework divides a visual entity (view) into panes with specific, consistent responsibilities and functionalities.
- Standardized interaction:
The interaction between the various subcomponents within the (application) layer is clearly defined, providing customizable base classes.
- Maintainable code:
Using a pattern results in maintainable code that provides a flexible and extensible code base for developing applications.
- Application flow support:
The framework structures the presentation (application) service into distinct layers and provides for inter- and intralayer communication. Such a structure offers a strong, orderly way to implement application logic and flow.

The HMVC pattern provides clear delineation of responsibility among the different components and layers. Standard design patterns (Abstract Factories, Composite, Chain of Responsibility, Facade etc.) can be used to provide a stable design.

Figure X: HMVC layers and components [Cai]

Figure X illustrates some layers and key components of the HMVC pattern.
The horizontal layers specify the hierarchy within the application; the vertical slices refer to the components of the MVC triad. Within a layer, the controller has the overall responsibility of

managing the model and view components. For example, the GUIFrame Controller controls the GUIFrame Model and the GUIFrame (the view). The dashed lines between model, controller, and view within a layer signify clearly defined interfaces for communication. This interaction is achieved through AppEvents. For intralayer communication, a parent-child controller hierarchy exists, and all intralayer communication can
only be routed through this path. Controllers interact by means of AppEvents."

3.2.1.3 Extended HMVC

...for runtime GUI switching

We must differentiate between the presentation layer (or client tier) and the GUI layer. The GUI layer deals with a small subset of the whole presentation layer, namely the UI widgets and the immediate effects of user actions -- a JTextField and its ActionListener, for example. The presentation layer needs to deal with application flows and server interaction in addition to providing GUI services.
The terms presentation layer and client tier are used interchangeably.

3.2.2 View

It provides a way for the user to communicate with the system. Nowadays, mostly graphical user interfaces are used, i.e. windows with menus, toolbars and several forms serve as the interface. A very common user requirement is to be offered a way to control an application also by keyboard only, without the need for a mouse or trackball. Both will be possible in Res Medicinae.

There are other user interfaces such as Braille devices for the blind. Also, vocal control of applications is increasingly used. Both are planned for future versions of this system.

The following description comes from [Cai] again. It explains the role of views in the HMVC pattern.

"A user interacts with the view, the visible portion of the application. HMVC abstracts views at different levels to provide a clean method for designing the GUI. At the highest level is a GUIContainer, with its associated controller. The container essentially holds potentially multiple views, called GUIFrame(s); each GUIFrame is a visual entity with which a user interacts. The framework defines a GUIFrame as composed of multiple subparts -- that is, a Menu GUIPane, a Navigation GUIPane, Status GUIPane, and a central Content GUIPane (see Figure 3). In most common Web applications, developers usually expect multiple GUIFrames to be unlikely; primarily, it is the Content GUIPane that needs to change. The Content GUIPane area is considered to be the most important part of the GUIFrame; that's where most of the user interaction occurs. The framework assumes that the efficient control of multiple Content GUIPanes will suffice to deliver a very large part of the user experience.

Figure 3. GUI components [Cai]

Figure 3 illustrates a typical GUI frontend. It breaks into several parts (i.e., GUIPanes). We can apply the MVC triad to each of the composing panes and establish a hierarchy, with the GUIFrame being composed of the Menu, Status, Nav, and Content GUIPanes. Depending on the complexity of the code within each component, we may or may not assign an independent controller and model to a GUIPane. For example, because of its simplicity and lack of any real

need for sophisticated control, it is not necessary for the Status GUIPane to have its own controller; we may choose to have the GUIFrame controller run the Status GUIPane instead. However, as the Content GUIPane is an important activity area, we might assign it a separate controller and model. Based on the MVC triad, a GUIFrame has its associated controller and data-holder model, as does the Content GUIPane. The GUIFrame layer has the GUIContainer as its parent triad. The GUIContainer is an invisible part of the architecture; it can potentially hold multiple GUIFrames.

A crucial aspect of the design is the isolation of Swing-specific code -- i.e., the Swing components and their listeners (refer back to Figure 2) -- within the lowest rung of the hierarchy. As an illustration, Swing widgets primarily compose the Content GUIPane. This is not a design limitation; a Nav GUIPane could also have a Swing component like, for example, a JTree. Therefore, the Content GUIPane is also responsible for catering to Swing events like ActionEvents. Similarly, an ActionEvent generated by clicking a JMenuItem within the Menu GUIPane is heard by the Menu GUIPane itself. Thus, a GUIPane acts as a listener for Swing events. The affected GUIPane can subsequently request further service from its controller by using application- level events. This allows for the localization of Swing-specific code."

In Java, there are two basic ways of implementing a user interface which are described in short in the following chapters:
- a standalone rich client application using the "Swing" library
- a servlet based web client using "Java Server Pages" (JSP)

3.2.2.1 Swing

That is the Java Swing Graphical User Interface (GUI) library. It can be used for standalone (rich) clients that have a sophisticated user interface (as opposed to most HTML web interfaces). Swing is somewhat slow at times but platform-independent and well designed.

If the layers of a system are divided properly, other user interfaces such as one for the Linux KDE desktop using the Qt library or another one using the wxWindows library can be created on top of the application layer.

3.2.2.2 Servlets/Java Server Pages

The use of servlets is becoming more and more popular. Java provides the "Java Server Pages" (JSP) which is Java code integrated into common HTML. At runtime, the HTML pages are parsed by a web server and transformed into pure HTML code which can then be displayed in any web browser.

In future versions of Res Medicinae, we may provide both, Swing and JSP interfaces.

3.2.2.3 UIML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uiml SYSTEM "UIML2_0a.dtd">
<uiml xmlns="http://uiml.org/dtds/UIML2_0a.dtd">
 <head>
 metadata
 </head>
```

```
<interface>
<structure>
hierarchical enumeration of all gui elements
        definition of different hierarchies for different devices are possible
        every element can get a unique name and be assigned a class
        presentation layer
        <part class="Frame">
          <part class="Container">
            <part class="InputButton" name="SubmitButton"/>
          </part>
        </part>
      </structure>
      <style>
        sets the properties like font, color, layout
        presentation layer
        <property part-class="Frame" name="rendering">
          Html
        </property>
        <property part-class="Container" name="rendering">
          Body
        </property>
        <property part-name="SubmitButton" name="rendering">
        </property>
        <property part-name="SubmitButton" name="type">
        </property>
        <property part-name="SubmitButton" name="value">
          <reference constant-name=" SubmitLabel "/>
        </property>
        <property event-class="ButtonClicked" name="rendering">
          onClick
        </property>
        <property call-class="submit" name="rendering">
          location.submit
        </property>
      </style>
      <content name="English">
        may be text, sound, images
        makes internationalization possible
        guis with different content are possible, e.g. for beginners and professionals
        presentation layer
        <constant name="SubmitLabel">
          Submit
        </constant>
      </content>
      <content name="German">
        <constant name="SubmitLabel">
          Uebertrage
        </constant>
      </content>
      <behaviour>
        dialog management layer
        <rule>
          <condition>
```

```
            <event-class="ButtonClicked" part-name="SubmitButton"/>
          </condition>
          <action>
            <call class=" submit"/>
          </action>
        </rule>
      </behaviour>
    </interface>
    <peers>
      mapping of elements to specific gui toolkit/platform
      <presentation name="html">
        <component name="Html" maps-to="html:html"/>
        <component name="Body" maps-to="html:body"/>
        <component name="ButtonSubmit" maps-to="html:input"/>
          <attribute name="value" maps-to="value.value"/>
          <attribute name="type" maps-to="type.value"/>
        <component name="onClick" maps-to="onClick"/>
      </presentation>
      <logic name="JavaScript">
        <component name="location" maps-to="">
          <method name="submit" maps-to="submit">
            <script>
              <![CDATA[ function submit () (...) ]]>
            </script>
          </method>
        </component>
      </logic>
    </peers>
    <template>
      definition of reusable components
    </template>
</uiml>
```

3.2.3 Controller

This layer contains the actual functionality of an application. It is often referred to as "business logic" or "workflow" layer.

The application layer is mostly implemented in standard Java. Classes most often used here are containers, facades, get/set methods.

Following UML diagrams are frequently used for modelling an application:
- Use Case Diagram (UCD)
- Activity Diagram (AD)
- Sequence Diagram (SD)
- State Chart Diagram (ScD)

[Cai] continues its description by explaining the role of controllers in HMVC:

"The controller uses the model to coordinate the effects of user events on the view with the model; it also caters to logic flow. HMVC defines layers within the (application) and provides for distributed control of events through a parent-child hierarchy of controllers. Within a layer,

the controller is the supreme commander, orchestrating the application flows and user-event responses. The 'Chain of Responsibility' design pattern implements the controllers, wherein they pass on events that they can't cater to.

For example, if, as a result of clicking a button within a Content GUIPane, the Menu GUIPane needs to change, then the ActionEvent would be intercepted by the Content GUIPane itself (as it is the listener for Swing/AWT events). The ContentGUIPane would subsequently make a navigation request to the
ContentGUIPane controller, which would, in turn, pass it on to its parent controller, the GUIFrame controller. This results because the change in the Menu GUIPane can be effected only at a higher level, as the Content GUIPane and Menu GUIPane are at the same level in the hierarchy (they are both
contained within a GUIFrame).

An absolute and clearly defined parent-child relationship is established between a GUIContainer controller at the topmost, or parent, level and its child, the GUIFrame controller. Similarly, there is a parent-child
relationship between a GUIFrame controller and a GUIContent Pane controller. The controller within each layer is only responsible for actions limited to its sphere of influence -- that is, the model and view at that level. For all other services, the controller needs to pass actions on to its parent.

If a controller can't handle its event, the 'Chain of Responsibility' pattern signals the controller to pass the event to its parent. Controllers communicate with each other via AppEvents -- which typically can be navigation events, data-request events, or status events. Navigation events are typically those that change the look and feel of the view. For example, if you click the JMenuItem within the Menu GUIPane -- which replaces the active Content GUIPane -- the navigation event would make the change. The application developer would need to identify these events and create some basic stereotypes.

Controllers can also communicate via data events. If a Content GUIPane needs to display data in some JTextField objects, then the Content GUIPane would create a data event. The Content GUIPane would then pass it to its controller, which, upon determining that it is a data event, would delegate it to the
associated model. The model subsequently would pass a refresh request to the Content GUIPane, which would deliver a clean and well-defined communication path.

The controller has multiple responsibilities; it must respond to application- level navigation events and data-request events, for instance. In response to navigation events, a controller provides application-flow
logic -- changing screens or disabling/enabling options, for example. For data-request events, the controller delegates the request to an associated model object."

3.2.3.1 Application

3.2.3.2 Main Application

3.2.4 Model

According to [Cai], the model, finally, plays the following role in HMVC:

"View entities like GUIContainer, GUIFrame(s), and GUIContent Pane(s) have associated

models. HMVC makes a provision for models at every layer of the hierarchy, but it is up to the application designer to actually implement them. The GUIContainer model typically contains data or information that
affects the whole application, while the GUIFrame model contains information related only to the state of a GUIFrame. The model contains or holds the data objects that are to be displayed or worked upon in a view. Typically, the model receives a delegated data-service request from the controller, fetches
the data, and notifies the associated view of the availability of fresh data.

The model encapsulates key server-interaction functionality to retrieve data. To accomplish this task, key issues -- ... transport data formats etc. -- need to be redressed. Design issues like data-persistence mechanisms in the model need to be ironed out. However, in most thin-client architectures, the role of the model tends to be that of a data conduit that hides the complexity of data fetching and decouples the rest of the (application) from server-side (domain layer) data formats."

When modelling distributed objects or client-server architectures (Remote Facades), the following UML diagram might be helpful:
- Distribution Diagram (DD)

3.2.4.1 Local Facade

3.2.4.2 Remote Facade

3.3 Domain

This layer is the actual heart of the system. It contains "domain objects" and their relationships among each other.

An application may get objects from the domain layer and alter them in some way. Afterwards, the objects are left in a new state.

A popular and very useful technology for this layer in conjunction with a relational database management system (RDBMS) are "Enterprise Java Beans" (EJB).

However, the domain layer objects can also be stored in an "Object Oriented Database Management System" (OODBMS) that would be integrated into this layer. In that case, the two layers "Data Access" and "Data Source" described below, would not be needed. Examples of OODBMS are FastObjects (Poet), Objectivity/DB and ObjectStore.

Useful UML diagrams most often applied in domain modelling are:
- Component Diagram (CmD)
- Class Diagram (CsD)

3.3.1 Data Access

This intermediate layer is only needed when using a Relational Database Management System (RDBMS). It is then responsible for translating the object oriented structure of the domain layer into an entity relationship structure for the data source layer (database).

When using Enterprise Java Beans in the domain layer, one can skip this data access (interface) layer because the EJBs generate the necessary database access code automatically.

Some UML tools offer a special diagram to do the OOM - ERM mapping:
- Entity Relationship Diagram (ERD)

3.3.2 Data Source

The place where all data are stored. This is the Relational Database Management System (RDBMS).

One doesn't need this layer when an Object Oriented Database Management System (OODBMS) is used in the domain layer.

A useful database to start with for Res Medicinae could be PostgreSQL. Another open source database is MySQL. Other, commercial databases are for example: ORACLE, DB2 (Informix), InterBase, Ingres and last, SQL Server.

4 Applications

4.1 Main

4.1.1 Res Medicinae

The class "ResMedicinae" inherits from "MainApplication".

It adds no big additional features and hence just provides a portal where other, also external, applications can be started from.

Figure X: ResMedicinae

4.2 Standard

4.2.1 Record

Handhabung Patientendatenbank im Praxisbetrieb:

Strikte Trennung von Patientenstammsatz (= einmalig) und Quartalsfallsatz (kann mehrmalig sein, z.B. 2x zum Notfalldienst dagewesen und einmal in der normalen Sprechstunde = 3 Fälle des gleichen Patienten).

Beim Quartalswechsel wird die alte Quartalstabelle archiviert und eine leere angelegt. Kommt ein Patient, wird von ihm ein Fall in der Quartalsliste angelegt. Dort sammelt man die im Quartal anfallenden
Ziffern und das letzte KVK-Einlesedatum für diesen Fall.

Nach jeder Abrechnung werden die dabei nicht abgerechneten, aber eigentlich abzurechnenden Patienten (z.B.: Ziffern eingetragen, aber keine Fallart angegeben) in die Nachzüglerliste übertragen. Bei Abrechnung des neuen Quartals wird die Liste der Nachzügler mit berücksichtigt.

Beim Layout von Patientenstammdaten sollte man darauf achten, die eigentlichen persönlichen Daten (Name, Geburtsdatum, Adresse, Telefon) von den landesspezifischen (KVK-Einlesedatum, Social Security Number, Krankenkassenzugehörigkeit, Fallart, ...) zu trennen.

record
record.content
record.diagnosis
record.diagnosis.chains
record.diagnosis.icd
record.history
record.laboratory
record.maintenance
record.note
record.observation
record.prescription
record.vitalsign

4.2.2 ReForm

form
form.germany
form.germany.abrechnung
form.germany.arbeitsunfaehigkeit
form.germany.attest
form.germany.berufsgenossenschaft.a13
form.germany.berufsgenossenschaft.d13
form.germany.berufsgenossenschaft.h13
form.germany.berufsgenossenschaft.d9
form.germany.berufsgenossenschaft.kd10
form.germany.briefkopf
form.germany.briefumschlag
form.germany.einweisung
form.germany.etiketten
form.germany.gesundheitsuntersuchung
form.germany.krankenpflege
form.germany.krankenhausbehandlung
form.germany.krebsfrueherkennung
form.germany.krebsfrueherkennung.frau
form.germany.krebsfrueherkennung.mann
form.germany.privat
form.germany.rezept
form.germany.rezept.privat
form.germany.rezept.krankenkasse
form.germany.therapieplan
form.germany.transport
form.germany.ueberweisung
form.germany.unfallmeldung

4.2.3 ResLab

4.2.4 Billing

Die Überprüfung gegen das Regelwerk zur Abrechnung sollte (vorerst) ein Modul eines Fremdanbieters (APW ?) vornehmen (da großer, dauernder Pflegeaufwand). Dieses kann binär sein und sollte etwa folgende API haben:

- Übergabe von
  - Patientenstammdaten
  - Patientendiagnosen
  - abzurechnende Ziffern
- Rückgabe von
  - Fehler: mehr als ein Quartal
  - Fehler: Ziffer falsch abgerechnet (mehrfach, Kollision, ...)
  - Fehler: Ziffernzusatz fehlt (Wegezone, Begündung, Uhrzeit, ...)
  - Fehler: ...
  - Abrechnung ist OK

Es sollte in der Lage sein, einen oder mehrere Patienten zu verarbeiten. Es sollte nach erfolgreicher Prüfung der übergebenen Daten eine konforme Abrechnungsdatei erzeugen können (dadurch brauchen wir vorerst keine eigene Zertifizierung von der KBV). Das Modul braucht sich nicht um Persistenz zu kümmern. Zugriff auf die API siehe oben.

billing
billing.germany.cooperativesociety
billing.germany.private
billing.germany.sickfund

4.2.5 Paraclinical

paraclinical
paraclinical.arthroscopy
paraclinical.computertomogram
paraclinical.bloodpressure
paraclinical.bloodpressure.longtime
paraclinical.electrocardiogram
paraclinical.ecg.ergometry
paraclinical.ecg.longtime
paraclinical.ecg.rest
paraclinical.endoscopy
paraclinical.gastroscopy
paraclinical.laboratory
paraclinical.laryngoscopy
paraclinical.magnetoresonance
paraclinical.sonogram
paraclinical.xray

4.2.6 Statistics

statistic
statistic.age
statistic.billing
statistic.drugs
statistic.group
statistic.laboratory
statistic.overall
statistic.privatedaily
statistic.sickfundbudget

4.2.7 Scheduling

scheduling
scheduling.birthday
scheduling.consultation
scheduling.visit
scheduling.waitingroom

4.3 Specialist

4.3.1 ResDentum

4.3.2 ResOcularum

4.4 HDTF Services

4.4.1 Patient Centred Services

4.4.1.1 PIDS

4.4.1.2 SLIMS

4.4.1.3 COAS

Laboratory results
Vital signs
Subjective and Objective observations and assessments
Observations and measurements provided by a specialist such as radiologist or pathologist who interprets images and other multi-media data
[should leverage existing standards such as HL7 and DICOM]
CIAS - Clinical Image Access Service (retrieve and manage images)
MTM - Medical Transcription Management (distribute transcription of clinical information

about the patient over integrated networks)
HILS - Health Information Locator Service (locate the segments of a distributed patient record)
PDS - Person Demographics Service (provide demographic information about persons, e.g. date of birth, name, address, relationships to other people etc.)
PRM - Patient Record Management (provide evidentiality, metadata, review of records by differing views and needs, medical quality assurance)

4.4.2 Provider Centred Services

HBPMS - Health Benefit Plan Management Service (set parameters for Health Benefit Plan, e.g. cost to member of o/p visit, cost to member of prescription, lifetime dollar maximums, drug/procedure exclusions etc.; includes complex enterprise rules)
ES - Eligibility/Benefit Plan Usage Service (verify that a person is enrolled in a Health Benefit Plan and the enrolement is active; query on the Enrolement Management Service)
WMS - Worklist Management Service (utilize information about planned activities to develop worklists for specific agents)

4.4.3 Enterprise Information Services

Knowledge and Decision Support Services
TQS - Terminology Query Service (access terminological information for use in mediation, presentation and dynamic discovery)
TSS - Terminology Storage Service (update terminology used by TQS)
HDIF - Health Data Interpretation Facility/Decision Support (aid decision support activities by analysing health information retrieved from heterogeneous sources)
CPAS - Clinical Protocol Access Service (provide complete or part clinical protocols for implementation as care plans)
LS - Location Service (provide information on physical locations, e.g. postal addresses, map references, wards etc.)
RMS - Resource Management Service (locate, allocate and dispose resources)
SS - Supplies Service (locate, allocate and dispose consumable resources)
?? - Organization Record Service
CAMS - Care Authorization Management Service (get authorization for the provision of a service or resource that is not automatically covered by this Health Benefit Plan)
RAD - Resource Access Decision
PIF - Pharmacy Interaction Facility (communicate prescription information between pharmacy prescribers and pharmacy dispensers)

4.4.4 Administration Centred Services

Financing
BS - Billing Service (accommodate variable pricing based on sequence of procedures performed)
CMS - Claims Management Service (request payment from a HBP for services provided by claimant or by third party)
TBC - Transaction Based Costing (calculate health service cost based on healthcare service transactions)
CRPS - Claims Re-Pricing Service (provide variable pricing based on sequence of procedures performed)
CCMS - Charge Capture Management Service (collect information needed to enable

effective charging for procedures, drugs administered, supplies used, services etc.)
OE/TS - Order Entry/ Tracking Service (allow electronical clinical orders made between departments and facilities)
OMS - Management of Clinical Orders (ensure that the correct sequence is followed and orders are forwarded to the correct provider/department)
CDCS - Clinical Documentation Capture Service (expedite capture and storage of clinical information)
SF - Scheduling Facility (deal with scheduling what, where and when, related to workflow)
PMF - Party Management Facility
(manage relationships between parties, e.g. rights, roles, responsibilities, authority; Finance DTF)
HRS - Healthcare Relationship Service (provide generic relationships and their types; may be included in PMF)
EMS - Enrolement Management Service (establish a relationship between a person and a Health Benefit Plan, e.g. between an employee's family and a Primary Care Physician)
RMS - Remittance Management Service (interface between CMS - Claims Management Service and accounts payable for payer and accounts receivable for provider)
RMS - Referral Management Service (request a Healthcare Specialist to perform services; this act is performed by the physician who presently has responsibility for the patient)
RUM - Resource Utilisation Management (assess resource utilisation)

4.5 Enterprise

(Contents)
SAP R/3:
Sales & Distribution
Materials Management
Production Planning
Financial Accounting
Controlling
Anlagenmanagement
Quality Management
Instandhaltung/Service
Personalmanagement
Projekte
SAPOffice
Branchenloesungen
Menu:
Buero
Logistik
Rechnungswesen
Personal
Infosysteme
Werkzeuge
System Hilfe
Mindmanager:
Organisation
Beschaffung
Customer Service
--

CA (Cross-Application Functions):

Document Management System
Classification System
CAD Integration
SAP Office
Plant Data Collection
General Task Functions
Documentation Tools
Distribution (ALE)
Eletronic Data Interchange
ArchiveLink
Message Control
Translations

FI (Financial Accounting):
Global Settings
General Ledger Accounting
Accounts Receivable
Accounts Payable
Legal Consolidation
Consolidation Preparation
Asset Accounting
Special Purpose Ledger
Eletronic Bank Statment
Financial Information System

TR (Treasury):
Cash Management
Cash Budget Management and Financial Budgeting
Commitment Accounting

CO (Controlling):
Controlling General
Overhead Cost Controlling
Product Cost Controlling
Profitability Analysis

IM (Investment Management):
Investment Programs
Investment Orders
Investment Projects

EC (Enterprise Controlling):
Profit Center Accounting
Executive Information System

LO (Logistics General):
Managing Material Master Data
Business Partners
Environment Data
Variant Configuration
Engineering Change Management
Logistics Information System

SD (Sales & Distribution):
Schedule Agreement Processing
Availability Check and Requirements
Pricing and Conditions
Sales
Shipping
Transportation
Foreign Trade
Billing
Sales Support
Credit Management
Information and Analysis

MM (Materials Management):
Consumption-Based Planning
Purchasing Guide
Inventory Management
Valuation and Account Assignment
Invoice Verification
Material Evaluation
Warehouse Management
Vendor Evaluation
Material Ledger

QM (Quality Management):
Quality Plannning
Quality Inspection
Quality Certificates
Quality Notifications

PM (Plant Maintenance):
Equipment and Technical Objects
Preventive Maintenance
Maintenance Order Management
Maintenance History
PP (Production Planning):
Bills of Material
Demand Management
Routings
Sales & Operations planning
Master Plannning
Capacity Planning
Material Requirements Planning
Production Orders
Kanban
Repetitive Manufacturing
Work Centers

PS (Project System):
Task Management
Reference Guide

PD (Personnel Planning and Development):

Organizational Management
Training and Event Management
Personnel Development
Workforce Planning
Personnel Cost Planning
Room Reservations Planning
Structural Graphics

PA (Personnel Administration and Payroll Accounting):
Personnel Administration
Benefits
Recruitment
Time Management
Incentive Wages
Travel Expenses
Payroll: Country Specifications

IN (International Development):
Africa (South Africa)
Asian Pacific Area (Australia, China, Japan, Singapore)
Europe (Austria, Belgium, Switzerland, Czech Republic, Germany, Denmark, Spain, Finland, France, Hungary, Italy, The Netherlands, Norway, Portugal, Russia, Sweden, United Kingdom)
North America (Canada, Mexico, USA)
South America (Argentina, Brazil)

BC (Basis Components):
Workflow Management
Frontend Services
Report Tree
Computer Center Management System
ABAP/4 Dictionary
ABAP/4 Workbench
ABAP/4 Query
SAP Graphics
SAP Communication
Style & Layout set Maintenance
Modification and Enhancements
Authorization Administration
Computer Aided Test Tool
Application Data Archiving and Reorganization

5 Domains

5.1 Healthcare

5.1.1 Doctor's Practice

Possible domain objects for Res Medicinae are:
- Patient/Person
- Legal Entity

- Address
- Observation/Measurement
- Blood Pressure
- Form
- Calculation

5.1.2 Hospital Information System

Personnel administration
  Patients: PIDS service
  Guardians/guaranteer
  Physicians
  Nurses
  Technologists
  Therapists
  Pharmacists
  Clerical Personnel
  Administrative personnel
  Maintenance personnel
  etc.
Institutions
  Hospital
  Clinic
  Office practice
  Laboratory
  Pharmacy
  etc.
Ordering
  Clinical Orders (medications, diagnostic procedures, therapeutic procedures): Pharmacy
  Event orders (ADT)
Tracking
  Enterprise (patient tracking)
  Departmental (workflow tracking): CORBA workflow
Scheduling
  Enterprise
  Departmental
Delivery of goods/services (order fulfillment)
  Clinical Observations/Results Reporting: CORBAlex (vocabulary service)
  Clinical Decision Support
Billing
  Healthcare Financial Services
Inventory
Common services (security, timekeeping, persistence, vocabulary etc.)

HL7

Finances
Patient
Organization
Scheduling
Clinical Documents

5.2 Enterprise

(Contents)
SAP R/3:
Sales & Distribution
Materials Management
Production Planning
Financial Accounting
Controlling
Anlagenmanagement
Quality Management
Instandhaltung/Service
Personalmanagement
Projekte
SAPOffice
Branchenloesungen
Menu:
Buero
Logistik
Rechnungswesen
Personal
Infosysteme
Werkzeuge
System Hilfe
Mindmanager:
Organisation
Beschaffung
Customer Service

6 Libraries

6.1 ResMedLib

Application View

http://www.nerosworld.com/ping/kiwi/
(TreeTable gui component: table whose first column is a tree)

http://sourceforge.net/projects/javacurses/
(java textual ui in console windows on Linux and Windows/DOS)

http://sourceforge.net/projects/gvf/
(Graphics Visualization)

http://sourceforge.net/projects/jfreechart/
http://jrefinery.com/jfreechart/samples.html
(2D diagrams; LGPL!!)

http://sourceforge.net/projects/jcharts/
(Charting lib; LGPL!!)

http://www.incors.org/
(Kunststoff Look&Feel; LGPL!)

http://www.L2FProd.com/
(Skins and Look&Feel; BSD!)

Application Controller

http://sourceforge.net/projects/jasperreports/
(Reporting tool, for print-and-on-screen reports --> Forms?)
"Danciu Teodor" <teodord@hotmail.com>
>Do you think your project would be capable by then to print any
>medical forms which would be specified in XML files?

I don't know how exactly are you planning to implement those forms, but I can tell you that JasperReports can generate reports using data retrieved from a database through JDBC. If those forms are more special and you keep your form data in customized Java objects (like JavaBeans or so), then JasperReports would be also fit for the job since you can pass your objects to the report filling operations as parameters and display their content on your reports. JasperReports allows you to use complex Java expression based
on the report parameters and the reports fields (fields from the database), when defining the text fields
that will appear on the output document.

http://sourceforge.net/projects/geheimnis/
(Encryption Wrapping)

http://sourceforge.net/projects/somnifugi/
(JMS for communication between processes/Threads in the same JVM)

http://sourceforge.net/projects/openjmail/
(eMail and Messaging; LGPL!!)

http://sourceforge.net/projects/grouppac/
(CORBA implementation)

http://sourceforge.net/projects/javadc/
(File sharing)

http://sourceforge.net/projects/kontor/
(ERP system)
http://www.geocrawler.com/archives/3/4838/2001/11/0/7037319/
(Message: Kontor and ResMedicinae?; Castor is not GPL!)

Application Model

Database

http://sourceforge.net/projects/squirrel-sql/
(SQL client to view DB tables)
(homepage contains also several useful DB tool links!)
http://sourceforge.net/projects/jxdbcon/
(contains a featureful PostgreSQL driver)

Tools

http://sourceforge.net/projects/astyle/
(Indenter and Formatter)

http://sourceforge.net/projects/jode/
(Java Decompiler)

http://sourceforge.net/projects/dom4j/
(XML, XPath, XSLT; BSD!)

Miscellaneous

http://www.gnu.org/software/java/java-software.html
(many other gnu java software projects)

6.2 External

6.2.1 Scope

7 Hardware Architecture

A common ambiguity arises when using the terms "Client" and "Server". Both can mean either a running application (process) or a computer.

As the name says, a server application serves its clients with data. The client applications, on the other hand, can access a server process to receive specific data. Actually, client and server applications can have
a pretty similar architecture, only that one provides and the other retrieves data. It is even possible, to mix them into just one application that would be client and server at the same time.

We have already described everything you need to know about client and server applications and their logical layers in the "Software Architecture" chapter. The next question is how to map those layers onto Hardware. Whilst the term "Layer" is used to describe logical separation of a system, a common word to
describe the separation on the Hardware side is "Tier".

7.1 Physical Tiers

Now, when talking about client and server in means of hardware, one should think about a computer (machine).

[Sanderson] writes:
"... a client (application) is a 'facade' for business objects. In some manner -- there are many legitimate methods -- a client (application) will 'populate' itself with data retrieved from business objects in the application's 'domain'. The domain (layer) might be:
- in the same process (application is its own server with domain access),
- on the same machine (client and server application run on one machine,

- on an application server (server application on a special server computer),
- or indirectly accessed through a web server (application).
It matters not."

The web server application, again, can run on its own machine or on the one of the server application.

7.2 Scenarios

7.2.1 Simple Application

Figure X: Simple all-in-one Application

7.2.2 Client and Server Application on the same Machine

Figure X: Client and Server Application on the same Machine

7.2.3 Server Application on a special Server Machine

Figure X: Server Application on a special Server Machine

7.2.4 Web Server accessing a Server Application

Figure X: Web Server accessing a Server Application

7.2.5 Server Application spread to many Machines

Figure X: Server Application spread to many Machines

8 Development

8.1 Activities

8.1.1 Prepare

All resources necessary for developing on the Res Medicinae system should come with the downloaded file.

However, to specify them in detail, here is a list of what a developer needs:

1 An operating system for which Java is available, e.g.:
- Linux
  http://www.linux.org
- Solaris
- Mac OS
- Windows

2 Java Development Kit (JDK) in version 1.4 or higher:
http://www.javasoft.com

3 Libraries:
- Ant
  http://jakarta.apache.org/ant/index.html
- Log4J
  http://jakarta.apache.org/log4j/docs/index.html

Most of those libraries are part of the Jakarta project:
- Jakarta
  http://jakarta.apache.org/

- XML 4 Java
  http://www.xml4j.org

4 Res Medicinae sources:
http://www.resmedicinae.org

8.1.2 Build

We are using the "Ant" utility of the Jakarta project for project building. Apache Ant is a Java based build tool. In theory it is kind of like make without make's wrinkles. It uses configuration files in XML format.

Our configuration file lies in the project's bin directory /resmedicinae/bin. Its name is "build.xml". For simplification of the build process, we have included some shell scripts/batch files into the /bin directory. To start a build process, just follow these steps:

1 Open up a terminal/console dos window

2 Change into the /bin directory of Res Medicinae
cd /opt/resmedicinae/bin

3 Run the build file you need, e.g.:
./build.sh

There are several other build.xml files in sub directories of /bin. The one on top builds the

whole system, including API documentation etc. and finally creates distributable archives. The ones in the sub directories may be used to build just their part of the system.

8.1.3 Run

For reasons of convenience, we have included some shell scripts/batch files into the /bin directory. To run an application, just follow these steps:

1 Open up a terminal/console dos window

2 Change into the /bin directory of Res Medicinae
cd /opt/resmedicinae/bin

3 Run the application you want, e.g.:
./application/main/resmedicinae/start.sh
./application/healthcare/record/start.sh

8.2 Create an Application

It is easy!

1 Create your new application (controller) class, inheriting from:
org.resmedicinae.resmedlib.application.AbstractApplicationController

2 Create a new view class for the application (if you wish the application to have a graphical user interface), inheriting from:
org.resmedicinae.resmedlib.application.AbstractApplicationView

3 Create a new model class for your application, inheriting from:
org.resmedicinae.resmedlib.application.AbstractApplicationModel

For further description and examples, see our sample applications in package:
org.resmedicinae.application.sample

8.3 Create a Domain

There is really nothing difficult in creating a new domain.

As a domain mostly corresponds to a package, just create a new directory and give it the name of your desired domain, e.g.: /resmedicinae/src/domain/mydomain

When you start to create domain objects for your new domain, then place them into that new directory. Also, those domain objects should belong to the corresponding package, e.g.:
org.resmedicinae.domain.mydomain.MyDomainObject

To sum up, a new class "MyDomainObject.java" would reside in directory:
/resmedicinae/src/domain/mydomain/org/resmedicinae/domain/mydomain/

9 Indexes

9.1 Abbreviations

?? - Organization Record Service of the HDTF
BS - Billing Service (accommodate variable pricing based on sequence of procedures performed) of the HDTF
CAMS - Care Authorization Management Service (get authorization for the provision of a service or resource that is not automatically covered by this Health Benefit Plan) of the HDTF
CCMS - Charge Capture Management Service (collect information needed to enable effective charging for procedures, drugs administered, supplies used, services etc.) of the HDTF
CDCS - Clinical Documentation Capture Service (expedite capture and storage of clinical information) of the HDTF
CIAS - Clinical Image Access Service (retrieve and manage images) of the HDTF
CMS - Claims Management Service (request payment from a HBP for services provided by claimant or by third party) of the HDTF
COAS - Clinical Observation Access Service (retrieve clinical information about a person, e.g. text, measurements, wave forms etc.) of the HDTF
CORBA - Common Object Request Broker Architecture. A CORBA "module" corresponds to a "package" in UML or Java and to a "namespace" in C++. On system level, a "package" usually is a "directory". Packages can also contain "modules" (as directories contain "files"). Hence, UML or Java "modules" usually correspond to "files" on system level.
CPAS - Clinical Protocol Access Service (provide complete or part clinical protocols for implementation as care plans) of the HDTF
CRPS - Claims Re-Pricing Service (provide variable pricing based on sequence of procedures performed) of the HDTF
EMS - Enrolement Management Service (establish a relationship between a person and a Health Benefit Plan, e.g. between an employee's family and a Primary Care Physician) of the HDTF
ES - Eligibility/Benefit Plan Usage Service (verify that a person is enrolled in a Health Benefit Plan and the enrolement is active; query on the Enrolement Management Service) of the HDTF
GNU - "GNU's Not UNIX" Rekursives Akronym, welches als Name fuer ein freies UNIX, basierend auf dem Linux Kernel, gewaehlt wurde.
HBPMS - Health Benefit Plan Management Service (set parameters for Health Benefit Plan, e.g. cost to member of o/p visit, cost to member of prescription, lifetime dollar maximums, drug/procedure exclusions etc.; includes complex enterprise rules) of the HDTF
HDIF - Health Data Interpretation Facility/Decision Support (aid decision support activities by analysing health information retrieved from heterogeneous sources) of the HDTF
HDTF - Healthcare Domain Task Force (CORBAmed) of the OMG
HILS - Health Information Locator Service (locate the segments of a distributed patient record) of the HDTF
HRS - Healthcare Relationship Service (provide generic relationships and their types; may be included in PMF) of the HDTF
JSP - "Java Server Pages" HTML Seiten, die Java Programmcode enthalten. Vor ihrer Anzeige werden die JSP Seiten durch ein Interpreter Programm gelesen und in reines HTML umgewandelt. Mit Hilfe von JSP koennen Applikationen eine plattformunabhaengige Web Oberflaeche erhalten und sind damit ueber WebBrowser bedienbar. Als Alternative zu diesen Web basierten Oberflaechen gibt es herkoemmliche stand alone Applikationen, also Programme, die eigenstaendig gestartet und ausgefuehrt werden und ihre Oberflaeche mitbringen. Siehe hierzu "Swing".

LS - Location Service (provide information on physical locations, e.g. postal addresses, map references, wards etc.) of the HDTF
MTM - Medical Transcription Management (distribute transcription of clinical information about the patient over integrated networks) of the HDTF
OE/TS - Order Entry/ Tracking Service (allow electronical clinical orders made between departments and facilities) of the HDTF
OMG - Object Management Group http://www.omg.org
OMS - Management of Clinical Orders (ensure that the correct sequence is followed and orders are forwarded to the correct provider/department) of the HDTF
PDS - Person Demographics Service (provide demographic information about persons, e.g. date of birth, name, address, relationships to other people etc.) of the HDTF
PIDS - Person Identification Service (unique identification of persons by matching person traits and identifiers) of the HDTF
PIF - Pharmacy Interaction Facility (communicate prescription information between pharmacy prescribers and pharmacy dispensers) of the HDTF
PMF - Party Management Facility (manage relationships between parties, e.g. rights, roles, responsibilities, authority; Finance DTF) of the HDTF
PRM - Patient Record Management (provide evidentiality, metadata, review of records by differing views and needs, medical quality assurance) of the HDTF
RAD - Resource Access Decision Service of the HDTF
RMS - Referral Management Service (request a Healthcare Specialist to perform services; this act is performed by the physician who presently has responsibility for the patient) of the HDTF
RMS - Remittance Management Service (interface between CMS
- Claims Management Service and accounts payable for payer and accounts receivable for provider) of the HDTF
RMS - Resource Management Service (locate, allocate and dispose resources) of the HDTF
RUM - Resource Utilisation Management (assess resource utilisation) of the HDTF
SF - Scheduling Facility (deal with scheduling what, where and when, related to workflow) of the HDTF
SLIMS - Summary List Management Service (compile and manage medical summary lists) of the HDTF
SS - Supplies Service (locate, allocate and dispose consumable resources) of the HDTF
Swing - Bezeichnung einer Quelltextbibliothek der Sprache Java fuer die Oberflaechenprogrammierung. Dient dem Erstellen von Oberflaechen fuer stand alone Applikationen.
TBC - Transaction Based Costing (calculate health service cost based on healthcare service transactions) of the HDTF
TQS - Terminology Query Service (access terminological information for use in mediation, presentation and dynamic discovery) of the HDTF
TSS - Terminology Storage Service (update terminology used by TQS) of the HDTF
W3C - World Wide Web Consortium http://www.w3.org
WMS - Worklist Management Service (utilize information about planned activities to develop worklists for specific agents) of the HDTF

9.2 Figures

9.3 Tables

9.4 Sources

[Apache] The Apache project. Open Source under the Apache License.
http://www.apache.org

[Blueprints] The Java Blueprints.
http://www.javasoft.com/...

[Buschmann] Buschmann, Frank. ...

[Cai] Cai, Jason; Ranjit Kapila; Gaurav Pal.
HMVC: The layered pattern for developing strong client tiers.
This hierarchical model eases the development of a Java-based client tier.
Java World, July 2000.
http://www.javaworld.com/javaworld/jw-07-2000/jw-0721-hmvc_p.html

[Fowler] Fowler, Martin. Analysis Patterns.
Addison-Wesley, 1997.
http://www.martinfowler.com

[Gamma] Gamma, Erich; Richard Helm; Ralph Johnson; John Vlissides (Gang Of Four).
Design Patterns.
Addison-Wesley, 1995.
ISBN: 0-201-63361-2.

[ResMedicinae] The Res Medicinae project. Free Software. GPL license.
Founded in April 2000.
Administrators: Christian Heller <christian.heller@tuxtax.de>,
Karsten Hilbert <karsten.hilbert@gmx.net>
http://www.resmedicinae.org

[Sanderson] Sanderson, Rick. <ricks@fourbit.com>
The Fourbit Group.
http://www.fourbit.com

[Scope] The Scope project. Open Source Software. BSD license.
Founded in April 2000.
Administrator: Steve Meyfroidt <smeyfroi@users.sourceforge.net>
http://scope.sourceforge.net

10 Attachments

10.1 GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or

Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version
of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five). C. State on the Title page the name of the publisher of the Modified Version, as the publisher. D. Preserve all the copyright notices of the Document. E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. H. Include an unaltered copy of this License. I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version. N. Do not retitle any existing section as "Endorsements" or

to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

10.2 History

Please version your changes following this recommendation:
(major).(minor).(<99 = preX, 99 = final).(bug fix)

0.0.1.0 May 2001
Christian Heller
Initial working draft.

0.0.1.1 June 2001
Christian Heller
Added UIML code example.

0.0.1.2 June 2001
Christian Heller
Added miscellaneous hints from Karsten (currently in German).

0.0.1.3 June 2001
Christian Heller
Added some aspects concerning the division of the system into layers.

0.0.1.4 July 2001
Christian
Heller
Edited for first source code release/prototype.

0.0.2.0 29.10.2001
Christian Heller
Completely restructured document keeping the "analysis.txt" document in mind.

0.0.2.1 30.10.2001
Christian Heller
Included sample domains of HDTF (CORBAmed), HL7, SAP R/3, Mindmanager.

0.0.3.0 15.11.2001
Christian Heller
Extended and properly formulated chapters "Preface", "Introduction" and parts of "Basics".
Restructured and finished first iteration of chapter "Software Architecture".
Started to fill chapter "Applications".
Inserted first things into chapter "Hardware Architecture".
Adapted chapter "Development" to new project directory structure.
Extended Sources in chapter "Indexes".

0.0.4.0 21.12.2001
Christian Heller
- moved the project directory structure from /resmedicinae/directories.txt
into chapter "2.2 Directory Structure" of this document
- included new chapter "6 Libraries"
- added some links of other free/opensource software projects which might be
useful for various features of Res Medicinae